RICE UNIVERSITY

Towards Personalized Human Learning At Scale: A Machine Learning Approach

By

Zichao Wang

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE

[illegible]
Richard Baraniuk (Apr 12, 2023 07:02 CDT)

Richard G. Baraniuk
C. Sidney Burrus Professor of
Electrical and Computer Engineering

[illegible]

Andrew Lan
Assistant Professor, College of Information and Computer Sciences, University of Massachusetts Amherst

Santiago Segarra
Santiago Segarra (Apr 12, 2023 10:55 CDT)

Santiago Segarra
W. M. Rice Trustee Assistant Professor of
Electrical and Computer Engineering

[illegible]
Michael C Mozer (Apr 13, 2023 08:51 EDT)

Mike Mozer
Senior Staff Research Scientist, Google Brain, and Professor, Department of Computer Science and Institute of Cognitive Science, University of Colorado, Boulder

[illegible]

Anshumali Shrivastava
Associate Professor of Computer Science

HOUSTON, TEXAS
April 2023

ABSTRACT

Towards Personalized Human Learning At Scale:
A Machine Learning Approach

by

Zichao Wang

This thesis focuses on personalized learning in education, a promising and effective means of learning where the instructions, educational materials, learning paths, analytics, and reports are tailored to each learner to best support their individual learning paths and improve learning outcomes. Current personalized learning relies heavily on expert instructors and is costly, has limited availability, and is unable to scale to meet the massive demand of learning today. This thesis takes a machine-learning approach to address the aforementioned issues by developing computational models that learn from educational big data to perform the activities central to personalized learning in education. First, I will present a series of works for learning content customization, including methods and systems to generate, evaluate, represent, and analyze different types of learning content such as math word problems, factual quizzes, and scientific formulae. Second, I will present two frameworks for learning analytics, which enable the understanding and tracking of the progress of large numbers of learners effectively and efficiently. Finally, I will present methodologies for trustworthy machine learning, a necessity for deploying machine learning systems in real-world educational scenarios. These methodologies include theoretical tools for understanding recurrent neural networks, the powerhouse underlying modern knowledge tracing models, and controllable data generation, enabling machines to behave more precisely according to human instructions.

Acknowledgments

This journey is not possible without the support and trust of many individuals.

I owe deep gratitude to my thesis advisor, Rich Baraniuk, for cultivating me into an independent researcher, for making research a fun experience, and for instilling in me the confidence and passion to pursue a career in research. His vision, positivity, openness, and commitment to excellence have profoundly shaped the way I work and live.

I thank my long-time collaborator and mentor, Andrew Lan, for being my "go-to" person whenever I want to discuss research. Andrew taught me, patiently and hands-on, how to do research at the very beginning and is still doing so today. Most of my projects are the result of discussing and grinding with him. I hope our collaboration continues.

I thank the rest of my committee members, Santiago Segarra, Anshumali Shrivastava, and Mike Mozer, for their critical feedback on this thesis. I am also grateful to Mike for his insightful suggestions for my research during and beyond my time at Google Research.

I am fortunate to have spent productive and memorable times at three industry labs as a research intern under incredible mentorships: with Cheng Zhang at Microsoft Research Cambridge, with Anima Anandkumar and Weili Nie at NVIDIA Research, and with Caile Collins and Nathan Dass at Google Research. Their rigor, work ethic, patient guidance, and deep domain knowledge helped me grow as a researcher tremendously and expanded my research agenda significantly. Thanks also to my wonderful collaborators whom I had the fortune to work with and learn from: Simon Woodhead, Sebastian Tschiatschek, Simon Peyton Jones, José Miguel Hernández-Lobato, Chao Ma, Wenbo Gong, Zhuoran Qiao, Chaowei Xiao, and the entire Tivoli Team.

I thank the entire Richb Group for a stimulating research environment. Special thanks to those with whom I closely collaborated and discussed ideas, had coffee, and shared office:

Lucy Liu, Shashank Sonkar, Randall Balestriero, Sina Alemohammad, Indu Manickam, Ryan Burmeister, Blake Mason, Jasper Tan, Daniel LeJeune, CJ Barberan, Hossein Babaei, Imtiaz Humayun, Lorenzo Luzi, Paul Mayer, Pavan Kota, Chris Metzler, Ali Mousavi, AmirAli Aghazadeh, Hamid Javadi, and Gautam Dasarathy. Thanks also to Debshila Basu Mallick and Drew Waters for the many collaborations they enabled at OpenStax.

I thank my amazing teachers who cultivated my intellectual curiosity and passion for learning. In particular, I want to thank J. Dennis Huston and Mike Orchard for teaching me invaluable lessons from which I will benefit for the rest of my life: Dennis on effective writing and Mike on learning to learn. The experience of learning from them serves as a constant reminder that true learning only happens with deep thinking and consistent effort.

I cherish the lasting friendships that I have made over the years and that continue to nourish and inspire me, including Lantao, Zhenwei, Xiao, Stephen, Weili, Ruqi, Wanrong, Helen, Yi, Yuan, and many others who have provided me immense support and made this journey joyful.

Thanks to Xincheng, for always being there for me, for the precious memories we shared, and for helping me become a better version of myself.

Finally, I thank my parents for their unwavering love, encouragement, and belief in me. They are my pillars of strength and the reason for who I am today. This thesis is dedicated to them.

To my parents, Xiangwen Xiong and Weijun Wang.

Contents

Illustrations

Tables

Chapter 1

Introduction

Recent technology developments are rapidly bringing new innovations to many fields but education. Indeed, since the Prussia educational revolution [233] that largely shaped the education system we have today worldwide, the ways humans learn and teach have yet to witness fundamental and significant changes. For example, the concept of "one-size-fits-all", classroom-style learning still looms large practically, where an instructor is usually responsible for many learners simultaneously. Personalized learning [99], e.g., in the form of one-to-one tutoring, which is often considered a more effective means of learning [245], is still financially costly, manually effortful, and thus not yet universally accessible.

The need for reforms in education is urgent because our current education system struggles to deliver high-quality, effective learning experiences at scale. For example, in many scenarios such as online learning and remote learning, which are increasingly becoming the norm, the reduced instructors' presence and planning make learning much less engaging and personalized, resulting in significant negative impacts on learning. In science, technology, engineering, and mathematics (STEM) education, many graduates report that they are ill-prepared for the workforce and lack clear pathways to stay current with constantly evolving technologies. We are in pressing need of innovative ways to significantly transform the current practices to best support every human's success in their unique learning journeys.

Fortunately, the soil for technological innovations for education is ripe. First, we now have *massive volumes of data* on what and how we learn and teach, thanks to the increasing digitization in education. This rich data on learning and teaching behaviors, interactions among educational content, learners, and instructors, learner and instructor profiles, preferences, records, and beyond, provides an opportunity for developing *data-driven* technologies to automate certain educational practices, lowering cost, and reducing repetitive manual efforts. Second, we now have *reliable algorithms and robust models* at our disposal to develop technologies to accurately and effectively model educational big data. Third, we now have *highly powerful computing resources* to efficiently process data, train and validate models, and perform computational experiments and simulations. Taking all these factors into consideration, the emerging and maturing discipline of *machine learning* provides a promising direction for disruptive educational technology developments.

This thesis proposes several ideas that together have the potential to form the foundation for the next generation education system that is personalized, scalable, and accessible to everyone. I take a *machine learning* approach, i.e., developing models that *learns* to perform pedagogical tasks often encountered in personalized learning from educational big data, with the eventual goal to improve how humans learn. I complement this approach with ideas from recent advances in multiple other disciplines including natural language processing, generative modeling, psychometrics, and learning sciences.

The new personalized learning system will encompass at least the following three dimensions. First, it should be able to *understand and curate learning content* customized for different learners and instructors. Second, it should be able to *analyze and track the learning progress* of each individual learner throughout their education journey. Third, it should conduct the above processing and analyses in a *safe, robust, and unbiased manner*, making it trustworthy for learners and instructors to use in practical, high-stake education scenarios. This thesis makes contributions to all the aforementioned dimensions, summarized below. Each of the chapters mentioned below belongs to one of the three research dimensions and describes a complete, standalone research project.

1.1 Thesis Overview

Understand and curate learning content. High-quality learning content such as digital textbooks, video lectures, and assignments is key to effective learning. However, such content has largely remained static and cannot adapt and personalize for different learners to account for their individual differences and maximize their learning outcomes. Moreover, producing such content and customizing it for each instructor and learner typically requires intensive human labor and incurs high costs.

In the first part of the thesis, I focus on the generation, evaluation, representation, and analysis of various types of learning content. **Chapter 2** describes a methodology to automatically generate math word problems given an equation and a few context keywords. **Chapter 3** examines to what extent modern large language models generate factual quiz questions from textbooks and how human experts evaluate the generations. **Chapter 4** describes a methodology to represent scientific content such as equations and formulas into numeric embeddings suitable for search and retrieval applications. **Chapter 5** and **Chapter 6** describe data- and compute-efficient frameworks for analyzing the difficulty and quality of the generation leveraging weakly supervised learning and variational inference methods, respectively. These methods are among the first designed specifically for educational content; some have already found real-world applications that now benefit learners worldwide.

Personalized learning analytics. The data on the interactions between learners and learning content reveal critical insights such as learners' learning progress and knowledge mastery, which will enable more informed data-driven educational decisions. However, such data is large-scale (millions of such interaction records) and multi-modal (contains text, math, figures, handwriting, and so on), rendering analyses challenging.

In the second part of the thesis, I focus on building flexible and scalable systems suitable for mining large-scale, complex educational data for learning analytics. **Chapter 7** describes a flexible first-of-its-kind framework for analyzing learners' open-ended responses, a rich and informative type of data that almost all existing analytical methods overlooked. **Chapter 8** describes an efficient algorithm for Bayesian Bayesian Multidimensional Item Response Theory (MIRT) models, achieving 100x runtime reduction while retaining inference accuracy similar to classic Bayesian inference methods in MIRT on large educational datasets.

Trustworthy Machine Learning for Education and Beyond. Many state-of-the-art AI systems for education solutions rely on modern deep learning (DL) methodologies that are often black-box in nature and uncontrollable, rendering them unsafe and unreliable in high-stake applications such as education.

In the third and final part of the thesis, I focus on the fundamental algorithmic issues underlying popular modern methodologies that power intelligent applications in education and aim to understand and improve these algorithms, often with implications reaching beyond education. **Chapter 9** studies the powerhouse of modern knowledge tracing – recurrent neural networks (RNNs) – via theoretical tools such as the max-affine spline operator and reinterprets RNNs as classic operators in signal processing (e.g., matched filters), whose behaviors are well understood. **Chapter 10** studies controllable data generation, a necessity in the educational content generation process, and describes a retrieval-based generation framework that enables a user to have explicit, fine-grained control over the generated outputs and that has already found applications beyond education, i.e., for scientists in the drug discovery process.

Chapter 2

Math Word Problem Generation with Mathematical Consistency and Problem Context Constraints

2.1 Introduction

Math word problems (MWPs) are an important type of educational resource that help assess and improve students' proficiency in various mathematical concepts and skills [344, 340]. An MWP usually has a corresponding underlying math equation that students will need to identify by parsing the problem and then solve the problem using this equation. An MWP is usually also associated with a "context", i.e., the (often real-world) scenario that the math equation is grounded in, expressed in the question's text. The equation associated with an MWP is often exact and explicit, while the context of the MWP is more subtle and implicit. It is not immediately clear how the context information can be extracted or represented. Table 2.1 shows an example of an MWP and its associated equation.

In this chapter, we study the problem of *automatically generating MWPs* from equations and context, which is important for three reasons. First, an automatic MWP generation method can aid instructors and content designers in authoring MWP questions, accelerating the (often costly and labor-intensive) MWP production process. Second, an automated MWP generation method can generate MWPs tailored to each student's background and interests, providing students with a personalized learning experience [344] that often leads to better engagement and improved learning outcomes [58, 149, 150, 159, 163, 280]. Third, an automated MWP generation method can potentially help instructors promote academic

Table 2.1 : An example of MWP and its underlying equation. See Table 2.2 for more information on the datasets.

MWP: Joan found 70 seashells on the beach . She gave Sam some of her seashells . She has 27 seashells . How many seashells did she give to Sam ?
Equation: `x = (70 - 27)`

honesty among students. While new technologies create new learning opportunities, instructors have growing concerns of technologies that enable students to easily search for answers online without actually solving problems on their own [221, 176]. Automatically generated MWPs that are unique and previously unseen yet preserve the underlying math components can potentially reduce plagiarism.

In addition to its educational utility, MWP generation is also technically challenging and interesting. An important consideration for MWP generation is *controllability*: in practice, human instructors or content designers often have clear preferences in the type of MWPs they want to use. Therefore, an MWP generation method should be able to generate MWPs that are of high language quality and are textually and mathematically consistent with the given equations and contexts. To date, there exist limited literature on MWP generation. Most prior works focus on automatically answering MWPs, e.g., [184, 186, 266, 297, 348, 282, 375] instead of generating them [235, 367, 262, 68]. Existing MWP generation methods also often generate MWPs that either are of unsatisfactory language quality or fail to preserve information on math equations and contexts that need to be embedded in them. See Section 2.4 for a detailed discussion.

2.1.1 Contributions

In this work, we take a step towards controllable generation of mathematically consistent MWPs with high language quality. Our approach leverages a pre-trained language model

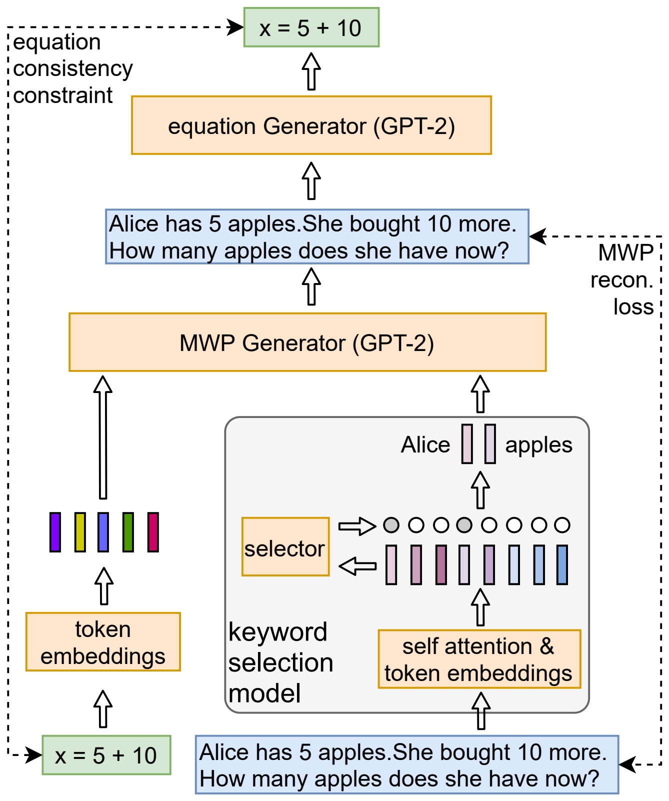

x = 5 + 10
equation consistency constraint
equation Generator (GPT-2)
Alice has 5 apples.She bought 10 more. How many apples does she have now?
MWP recon. loss
MWP Generator (GPT-2)
Alice
apples
selector
keyword selection model
self attention & token embeddings
token embeddings
x = 5 + 10
Alice has 5 apples.She bought 10 more. How many apples does she have now?

Figure 2.1 : An illustration of our MWP generation approach and its key components.

(LM) as the base model for improved language quality. The input to the LM is an equation and a context, from which the LM generates an MWP. On top of that, we introduce 2 components that impose constraints on the mathematical and contextual content of the generated MWP. First, to improve mathematical consistency and control over equations, we introduce an equation consistency constraint, which encourages the generated MWP to contain the exact same equation as the one used to generate it. Second, to improve control over contexts, we introduce a context selection model that automatically extracts context from an MWP. Quantitative and qualitative experiments on real-world MWP datasets show that our approach (often significantly) outperforms various baselines on various language quality and math equation accuracy metrics.

2.2 Methodology

We formulate the task of controllable MWP generation as a conditional generation problem. In this work, we consider datasets $\mathcal{D} = \{(M_i, E_i)\}_{i=1}^{N}$ in the form of N (MWP, equation) pairs where M_i and E_i represent MWP and its associated equation, respectively. In the remainder of the work, we will remove the data point index to simplify notation. This setup assumes each MWP in our dataset is labeled with an underlying equation but its context is unknown. Then, the MWP generation process can be described as

$$\begin{aligned} M &\sim \mathbb{E}_{E\sim\mathcal{D}}[p_\Theta(M|E)] \\ &= \mathbb{E}_{E\sim\mathcal{D},\,\mathbf{c}\sim p(\mathbf{c}|E,M)}[p_\Theta(M|E,\mathbf{c})]\,, \end{aligned} \tag{2.1}$$

where $M = \{m_1, \ldots, m_T\}$ represents the MWP as a sequence of T tokens (e.g., words or wordpieces [338, 268]). E and $\mathbf{c}$ are the controllable elements, where $\mathbf{c}$ represents a problem context. p_Θ is the MWP generative model parametrized by a set of parameters Θ.

In this work, we use a pre-trained language model (LM) as the generative model p_Θ, similar to the setup in [152]. We choose LMs over other approaches such as sequence-to-sequence (seq2seq) models because they can be pre-trained on web-scale text corpora. Pre-trained LMs thus often generate high-quality text and generalizes well to out-of-domain words not present in the training data. Under an LM, we can further decompose Eq. 2.1 into

$$p_\Theta(M|E,\mathbf{c}) = \prod_{t=1}^{T} p_\Theta(m|E,\mathbf{c},\{m_s\}_{s=1}^{t-1})\,. \tag{2.2}$$

To train p_Θ via fine-tuning the LM, we use the usual negative log-likelihood objective:

$$\mathcal{L}_{LM} = \sum_{t=1}^{T} -\log p_\Theta(m_t|E,\mathbf{c},\{m_s\}_{s=1}^{t-1})\,. \tag{2.3}$$

The above training objective serves as a proxy that optimizes for language quality. However, it alone is unsatisfactory in 2 ways. First, there is no guarantee that the generated MWP is mathematically valid; even if it is, its solution may correspond to an equation that is different from the input equation [409]. Second, while the context $\mathbf{c}$ can be manually specified, i.e., as a set of keywords, it is unobserved during training and needs to be inferred from data through the costly-to-compute posterior distribution. In the remainder of this section, we introduce our novel approach to tackle these challenges. We first describe our equation consistency constraint that improves the generated MWP's mathematical consistency and then detail our context selection method that learns to extract the context in the form of a set of keywords from an MWP. Figure 2.1 provides a high-level overview of our overall approach.

2.2.1 Equation Consistency

We propose an equation consistency constraint to promote the generated MWP to correspond to an equation that is the same as the input equation used to generate the MWP.

To formulate this constraint, we need a model to parse an equation given an MWP, i.e., a mwp2eq model, and a loss function, which we call $\mathcal{L}_{eq}$. The mwp2eq generative process can be written as

$$E' \sim \mathbb{E}_{M' \sim p_\Phi(M|E)}[p_\Phi(E|M')] \,,$$

where p_Φ is the mwp2eq model, E represents an equation, and M' represents the generated MWP. Here, we treat the equation as a sequence of math symbols e_t, making it appropriate for sequential processing. Specifically, we treat each variable (e.g., x, y), math operator (e.g., =, $\times$, +), and numeric value (e.g., integers, fractions, and decimal numbers) as a single math symbol. Therefore, we can decompose $p_\Phi(E, M')$ similar to Eq. 2.2. There are ways to represent math equations other than a sequence of symbols, such as symbolic trees [394, 66, 211]; finding ways to make them compatible to LMs is left for future work. Similar to $\mathcal{L}_{\text{LM}}$, we minimize a negative log-likelihood loss that uses the input equation E as the ground truth for the equation E' parsed from M':

$$\mathcal{L}_{\text{eq}} = \sum_{t=1}^{T} -\log p_\Phi(e_t | M', e_1, \ldots, e_{t-1}) \,. \tag{2.4}$$

This constraint is reminiscent of the idea of "cycle consistency" that has found success in image and text style transfer [411, 294], question answering [386, 353], and disentangled representation learning [134].

Gumbel-Softmax Relaxation. To back-propagate loss to p_Θ and compute gradient for Θ, we need the loss $\mathcal{L}_{\text{eq}}$ be differentiable with respect to Θ. The challenge here is that M' is sampled from p_Θ and that this discrete sampling process is non-differentiable, preventing gradient propagation [238]. To tackle this challenge, we resort to the Gumbel-softmax relaxation [130, 209] of the discrete sampling process $m_t \sim p_\Theta$. Details are deferred to the Supplementary Materials.

We remark that the gradient derived under the Gumbel-softmax relaxation is a biased but low-variance estimate of the true gradient [130, 209]. The low-variance property makes it more attractive for real applications than other unbiased but high-variance estimators such as REINFORCE [366]. We refer to [130, 209] for more details on the Gumbel-softmax method. In addition, while one can also use deterministic relaxation such as softmax, Gumbel-softmax injects stochastic noise during the training process, which regularizes the model and potentially improves performance; See an empirical comparison in Table 2.6.

2.2.2 Context Selection

In practice, we do not have access to the contexts $\mathbf{c}$ during training since they are not specified for real-world MWPs. Therefore, we need ways to specify the context for the MWP generative process. Existing methods characterize context as a "bag-of-keywords", using heuristic methods such as TF-IDF weights to select a subset of tokens as "keywords" from an MWP as its context. These methods are simple but lack flexibility: they either require one to specify the number of tokens to use for each MWP or heuristically select only certain types of tokens (e.g., nouns and pronouns) [409, 200].

In this work, we adopt this "bag-of-tokens" characterization of context, which fits well into LMs, but instead *learn* a context (token) selection method from data. To do so, we interpret $\mathbf{c}$ as a "context keyword selection" variable, i.e., a binary random vector whose dimension is the number of tokens in the vocabulary. Each entry $c^{(i)}$ in $\mathbf{c}$ is an i.i.d. Bernoulli random variable with prior probability ρ, i.e., $p_c(c^{(i)} = 1) = \rho$. Thus, $\mathbf{c}$ acts as a selector that chooses appropriate context tokens from the entire vocabulary. To circumvent the intractable posterior $p(\mathbf{c}|E, M)$, we resort to the auto-encoding variational Bayes (VAE) paradigm [157], similar to [295]. Under the VAE setup, we select a set of tokens conditioned on the MWP as $\mathbf{c} \sim q_\Psi(M)$ where $q_\Psi(M)$ is a proposal distribution, i.e., the keyword selection model.

Context Keyword Selection Model. Given an MWP, we first compute the contextualized embeddings of each token using a simple linear self-attention method as

$$\widetilde{\boldsymbol{m}}_t = \boldsymbol{M}\boldsymbol{a}_t\,, \quad \boldsymbol{a}_t = \mathrm{softmax}\left(\frac{\boldsymbol{M}^\top \boldsymbol{m}_t}{\sqrt{D}}\right),$$

where $\boldsymbol{M} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T] \in \mathbb{R}^{D \times T}$ is the matrix with all token embeddings and D is the embedding dimension. The $\sqrt{D}$ term is added for numerical stability [338]. Then, we compute $q_\Psi^{(i)}(M)$, the probability that each word in the vocabulary is selected as a context keyword, with a single projection layer with Sigmoid activation

$$q_\Psi^{(i)}(M) = \sigma(\mathbf{w}^\top \widetilde{\boldsymbol{m}}_t + b)\, \mathbf{1}_{\{V^{(i)} \in M\}}\,, \tag{2.5}$$

where $\mathbf{w}$ and b are part of the model parameters Ψ. The indicator function at the end ensures that only tokens that appear in M can be selected as context keywords. In practice, we also mask out stopwords and punctuation; these steps ensure that the context selector selects keywords that are relevant to MWPs and are not too generic.

Optimization Objective. Under the VAE paradigm, we optimize the keyword selection model using the so-called evidence lower bound (ELBO):

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{LM}} + \beta \mathcal{L}_{\text{c}}\,, \tag{2.6}$$

where $\mathcal{L}_{\text{c}} = \text{KL}(q_\Psi \| p_c)\, \mathbf{1}_{\{V^{(i)} \in M\}}$ and the Kullback-Leibler divergence term can be computed analytically thanks to our Bernoulli parametrization. $\mathcal{L}_{\text{c}}$ can be interpreted as a context constraint that prevents the keyword selection model from choosing too many keywords. The hyperparameter β and prior ρ controls the strength of this constraint. Because $\mathbf{c}$ is discrete and its sampling process is also non-differentiable, we use the straight-through estimator of the gradient [19] for Θ involved in $\mathcal{L}_{\text{LM}}$ in Eq. 2.6.

Table 2.2 : Summary statistics of datasets.

Dataset	#MWPs	avg #words per MWP	avg #symbols per eq
arithmetic	1,492	29.89	8.05
MAWPS	2,373	31.25	8.16
Math23K	23,162	35.23	8.78

2.2.3 Training

We train (fine-tune) the LM, the mwp2eq model, and the keyword selection model jointly. The mwp2eq model and keyword selection model are optimized using their respective objectives defined in Eqs. 2.4 and 2.6. The overall objective for the MWP generative model p_Θ is

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \alpha\mathcal{L}_{\text{eq}} + \beta\mathcal{L}_{\text{c}}\,,$$

where $\alpha > 0$ and $\beta > 0$ are hyperparameters that balance these constraint terms.

2.3 Experiments

We now perform a series of experiments to validate the effectiveness of our proposed MWP generation approach. Quantitatively, we compare our approach to several baselines on various automated language quality and mathematical consistency metrics. Qualitatively, we showcase the capability of our approach in generating controllable, high-quality MWPs.

Datasets. We focus on MWP datasets in which each MWP is associated with a single equation and each equation contains a single unknown variable. Therefore, we consider three such MWP datasets including **Arithmetic** [119], **MAWPS** [161], and **Math23K** [355]. Table 2.2 shows summary statistics for each dataset. We follow the preprocessing steps

Table 2.3 : A comparison of language quality and mathematical validity for MWPs generated by our method to various baselines. Numbers in brackets indicate the accuracy of the mwp2eq model trained on each dataset, which is an upper bound on the performance under the ACC-eq metric.

	Arithmetic				**MAWPS**				**Math23K**			
	BLEU-4	METEOR	ROUGE-L	ACC-eq (0.769)	BLEU-4	METEOR	ROUGE-L	ACC-eq (0.755)	BLEU-4	METEOR	ROUGE-L	ACC-eq (0.672)
seq2seq-rnn	0.075	0.152	0.311	0.413	0.153	0.175	0.362	0.472	0.196	0.234	0.444	0.390
+ GloVe	**0.351**	0.310	0.555	0.399	0.592	0.412	0.705	0.585	0.275	0.277	0.507	0.438
seq2seq-tf	0.339	0.298	0.524	0.405	0.554	0.387	0.663	**0.588**	0.301	0.294	0.524	**0.509**
GPT	0.237	0.248	0.455	0.401	0.368	0.294	0.538	0.532	0.282	0.297	0.512	0.477
GPT-pre	0.316	**0.322**	0.554	0.403	0.504	0.391	0.664	0.512	0.325	**0.333**	**0.548**	0.498
ours	0.338	**0.322**	**0.567**	**0.453**	**0.596**	**0.427**	**0.715**	0.557	**0.329**	0.328	0.544	0.505

in [409] by first replacing all numbers in both MWPs and equations to special tokens `num1`, `num2` etc. and then tokenizing both MWPs and equations into tokens and math symbols, respectively. In addition, we translate Math23K to English because this dataset is originally in Mandarin Chinese. Extension to languages other than English is left for future work.

Other popular MWP datasets such as Algebra [169, 328], Dolphin18K [122] and MathQA[1] [6] contain MWPs with multiple equations and many variables, which are challenging to generate even for humans. We leave the more challenging case of generating multi-variable, multi-equation MWPs to future work.

Setup and Baselines. We implement the LM and the mwp2eq models in our approach using pre-trained GPT-2 [268]; one can also use other models since our approach is agnostic to the specific model architecture. We consider three baselines: **seq2seq-rnn**, a sequence-to-sequence (seq2seq) model using LSTMs with attention that serves as the base architecture in [409, 200]; **seq2seq-rnn-glove**, a modification to the previous baseline with GloVe [255] instead of random embeddings at initialization; and **seq2seq-tf**, a seq2seq model with transformers [338]. We also compare our approach to vanilla GPT-2, either randomly

[1]MathQA is the most difficult MWP dataset we have encountered, which containing GRE and GMAT level questions.

Table 2.4 : % of generated MWPs that are *not* present in the training data. Our approach generates novel MWPs not seen in the training data most of the time while seq2seq-tf may simply memorize the training data.

	Arithmetic	**MAWPS**	**Math23K**
seq2seq-tf	6.24%	2.49%	38.88%
ours	**94.90%**	**63.77%**	**95.72%**

initialized or pre-trained; we denote these baselines as **GPT** and **GPT-pre**, respectively. For fair comparison, each baseline takes both equation and a set of keywords chosen by heuristics (see Section A.3.2) as input to be consistent with the setup in our approach. For each dataset, we perform five-fold cross-validation and report the averaged evaluation results. See the Supplementary Material for more details on the experimental setup and baselines.

Metrics. For language quality, we use the following three evaluation metrics: **BLEU-4** [246], **METEOR** [178], and **ROUGE-L** [194], following recent literature on question generation [361]. We implement these metrics using the package provided by [46]. For mathematical consistency, We use the equation accuracy (**ACC-eq**) metric that measures whether the generated MWP is mathematically consistent with the controlled input equation. The idea of this metric originates from other applications such as program translation and synthesis [48, 47]. In our case, because the equation associated with a generated MWP is not readily available, we resort to a mwp2eq model fine-tuned on each MWP dataset to predict the equation from an MWP. During the evaluation, we feed the generated MWP to the mwp2eq model as input and check whether the output of the mwp2eq model exactly matches the equation used as input to the MWP generator.

Table 2.5 : Generated MWP examples with fixed context and varying equations.

Context: `candies`	
Equation #1: `x = num1 + num2`	**Equation #2: `x = num1 - num2`**
seq2seq-tf: ethan has num1 presents . alissa has num2 more than ethan . how many presents does alissa have ? (in training data)	**seq2seq-tf**: mildred weighs num1 pounds . carol weighs num2 pounds . how much heavier is mildred than carol ? (in training data)
GPT-pre: There are num1 scissors in the drawer. Keith placed num2 scissors in the drawer. How many scissors are now there in total? (irrelevant to context)	**GPT-pre**: Joan has num1 blue balloons but lost num2 of them. How many blue balloons does Joan have now? (irrelevant to context)
ours: Mildred collects num1 candies. Mildred's father gives Mildred num2 more. How many candies does Mildred have? (✓)	**ours**: There are num1 candies in the jar. num2 are eaten by a hippopotamus. How many candies are in the jar? (✓)
Equation #3: `x = num1 * num2`	**Equation #4: `x = num1 / num2`**
seq2seq-tf: each banana costs $ num1 . how much do num2 bananas cost ? (in training data)	**seq2seq-tf**: there are num1 bananas in diane ' s banana collection . if the bananas are organized into num2 groups , how big is each group ? (in training data)
GPT-pre: Joan has saved num1 quarters from washing cars. How many cents does Joan have? (inconsistent with equation)	**GPT-pre**: Joan has num1 blue marbles. Sandy has num2 times more blue marbles than Melanie. How many blue marbles does Joan have? (inconsistent with equation)
ours: Each child has num1 candies. If there are num2 children, how many candies are there in all? (✓)	**ours**: There are num1 candies in the candy collection. If the candies are organized into num2 groups, how big is each group? (✓)

2.3.1 Quantitative Results

Table 2.3 shows the quantitative results of our experiments. The number in parenthesis below ACC-eq is the equation accuracy when we feed the mwp2eq model the ground-truth MWPs in the respective datasets. We see that our approach outperforms the best baseline on most occasions, especially on language quality metrics. However, there are a few exceptions, especially for the ACC-eq metric on the Math23K dataset. Specifically, we note that the seq2seq-tf baseline seems to yield an ACC-eq value even higher than the oracle accuracy at the first attempt. Upon closer investigation, we find that the baseline seq2seq models, especially the seq2seq-tf baseline, simply memorize the training data. Table 2.4 illustrates this finding and shows the percentage of generated MWPs that are *not* present in the training data. We see that the seq2seq-tf baseline tends to directly copy MWPs from the training data

Table 2.6 : Results of the ablation study, which validate the effectiveness of each component in our approach.

	Arithmetic		**MAWPS**		**Math23K**	
	BLEU-4	ACC-eq	BLEU-4	ACC-eq	BLEU-4	ACC-eq
$\mathcal{L}_{\text{eq}}$ (softmax)	0.110	0.417	0.308	**0.555**	0.284	0.466
$\mathcal{L}_{\text{eq}}$ **(Gumbel-softmax)**	**0.303**	**0.455**	**0.522**	0.527	**0.306**	**0.495**
keyword, TF-IDF	0.313	**0.424**	0.518	0.536	0.310	0.498
keyword, noun+pronoun	0.316	0.413	0.504	0.512	**0.325**	0.498
context selection	**0.320**	0.412	**0.533**	**0.542**	0.324	**0.501**
full model w/o $\mathcal{L}_{\text{c}}$	0.303	**0.455**	0.522	0.527	0.306	0.495
full model w/o $\mathcal{L}_{\text{eq}}$	0.320	0.412	0.491	0.500	0.324	0.501
full model w/o both	0.316	0.403	0.504	0.512	0.325	0.498
full model	**0.338**	0.453	**0.596**	**0.557**	**0.332**	**0.513**

as its “generated” MWPs, especially on the 2 smaller datasets. In contrast, our approach generates novel MWPs most of the time. We thus report ACC-eq only on the novel MWPs generated by the seq2seq-tf baseline on the Math23K dataset. Our approach outperforms seq2seq-tf on this modified ACC-eq metric.

Ablation Study. To validate that each component in our approach contributes to its success, we conduct an ablation study and compare our approach with several variants and several baselines after removing some of these components. For the use of Gumbel-softmax in the equation consistency constraint computation, we compare to softmax [98], which removes sampling from the Gumbel variable. For the context keyword selection model, we compare to several context keyword selection heuristics including TF-IDF [144, 409] and nouns+pronouns; see the Supplementary Material for more details on these baselines. Table 2.6 shows the ablation study results, reporting on BLEU-4 and ACC-eq as the representative metric for language quality and mathematical consistency, respectively. These comparisons validate the necessity of each component in our approach: Gumbel-

Table 2.7 : Generated MWP examples with novel context not present in the training data.

Equation: `x = num1 + num2 + num3`	
Context #1: `violin piano acoustic guitar`	**Context #2:** `beets eggplant`
seq2seq-tf: sara grew num1 onions , sally grew num2 onions , and fred grew num3 onions . how many onions did they grow in all ? (in training data)	**seq2seq-tf**: sara grew num1 onions , sally grew num2 onions , and fred grew num3 onions . how many onions did they grow in all ? (in training data)
GPT-pre: There are num1 dogwood trees currently in the park. Park workers will plant num2 dogwood trees today and num3 dogwood trees tomorrow. How many dogwood trees will the park have when the workers are finished? (irrelevant to context)	**GPT-pre**: There are num1 orchid bushes currently in the park. Park workers will plant num2 orchid bushes today and num3 orchid bushes tomorrow. How many orchid bushes will the park have when the workers are finished? (irrelevant to context)
ours: Mike joined his school's band. He bought a clarinet for \$ num1, a music stand for \$ num2, and a song book for \$ num3. How much did Mike spend at the music store? (✓)	**ours**: Sara grew num1 beets, Sally grew num2 beets, and Fred grew num3 beets. How many beets did they grow in total? (✓)

softmax outperforms softmax and our context keyword selection method outperforms other heuristic methods. We also see that our approach outperforms variants with either component removed and that the equation consistency constraint and the context keyword selection method tend to improve the mathematical consistency and language quality of the generated MWPs, respectively.

2.3.2 Qualitative Results

Since seq2seq baselines outperform our approach on a few occasions under the automated metrics, we now conduct a few case studies to investigate each approach. We investigate i) how controllable is each approach by giving it different input equations and contexts and ii) how generalizable each approach is by giving it unseen contexts in the dataset. Specifically, we conduct two qualitative experiments: First, we hold an input context fixed and change the input equation; Second, we hold the input equation fixed and change the context. We compare the MWPs generated by our approach to those generated by the seq2seq-tf and GPT-pre baselines trained on the MAWPS dataset, where these baselines perform well under automated metrics (see Table 2.3). The Supplementary Material contains additional qualitative examples.

Fixed Context, Changing Equation. Table 2.5 shows the MWPs generated by each approach using the same input context and different input equations. We see that every approach can generate MWPs with high language quality and are mathematically valid most of the time. However, upon closer inspection, we find that MWPs generated by the seq2seq-tf baseline are often exact copies of those it has seen in the training data. In other words, the model does nothing more than memorizing the training data and retrieving the most relevant one given the input equation and context; see Table 2.4 for a numeric comparison and the discussion in Section 2.3.1. This observation is not surprising because training only on small MWP datasets leads to overfitting. It also explains why the seq2seq baselines perform well on the automated metrics since these MWP datasets contain problems that lack language diversity, which results in many overlapping words and phrases that often appear in both the training and validation sets. The GPT-pre baseline, on the other hand, is sometimes capable of generating novel MWPs, but they are either irrelevant to the input context or are inconsistent with the input equation. Only our approach consistently generates MWPs that are both novel and mathematically consistent with the input equation.

Fixed Equation, Changing Context. Table 2.7 shows the MWPs generated by each approach using the same input equation and different input contexts. The keywords in these contexts are not part of the vocabulary of the training set and are thus unseen by the model during training/fine-tuning. Similar to the results of the previous experiment, here we also see that the seq2seq baseline simply retrieves an MWP from the training dataset as its "generated" MWP. This observation is unsurprising for the seq2seq baseline because it simply converts an out-of-vocabulary word in the input context into a special `unknown`

Table 2.8 : Examples of the keywords that are selected from a (possibly long) input context.

MWP: Emily collects num1 cards . Emily ’ s father gives Emily num2 more . Bruce has apples . How many cards does Emily have ?   **Context keywords:** Emily cards collects father
MWP: The school cafeteria had num1 apples . If they used num3 to make lunch for the students and then bought num2 more , how many apples would they have ?   **Context keywords:** apples cafeteria

token, which is uninformative. Interestingly, the GPT-pre baseline also generates MWPs that have minimal difference from MWPs in the training set or seems to ignore the input context. We again attribute this phenomenon to the small dataset size, on which the model also overfits if no additional constraints are introduced. Once again, in this setting, only our approach consistently generates novel and high-quality MWPs that are relevant to the input context.

Selected Context Keywords. To investigate our context keyword selection model, we show in Table 2.8 a few examples of the input context (which is the original MWP in our training setting) and the selected context keywords, i.e., those with $c^{(i)} > 0.5$ (recall Eq. 2.5). We see that our context keyword selection model can identify components that are key to the relevant underlying mathematical components in the MWP; for example, it identifies only “Emily collects cards father” as the key to this MWP and ignores the part with “Bruce apples”, which is unrelated to the math equation. Such a context keyword selection method is useful in practice to summarize (possibly long) input contexts provided by human instructors/content designers.

2.4 Related Work

MWP Generation and Answering. Earlier works on MWP generation do so in a highly structured way, explicitly relying on domain knowledge and even pre-defined equation and text templates [235, 367, 262, 68]. More recently, neural network-based approaches have shown significant advantages in generating high-quality questions compared to template-based approaches. Recent approaches on MWP generation also take this approach, usually using recurrent neural networks in a seq2seq pipeline [409, 200]. Instead of focusing on building new datasets or specific model architectures, we tackle the MWP generation problem from a controllable generation perspective, where we focus on the generated MWPs' language quality and mathematical consistency. This focus leads to our proposed approach that specifically aims at tackling these two challenges; our framework is model-agnostic and can be combined with almost any existing MWP generation approach.

Our approach also involves a model (mwp2eq) that parses an MWP into its underlying equation, which has been a very active research area with a plethora of related work, e.g., [121, 52, 379, 416, 184, 186, 266, 297, 349, 348, 282, 350, 6, 375]. In this work, we simply use a pre-trained LM as the mwp2eq model; investigation of leveraging the above recent advances to improve mathematical consistency of the generated MWPs is left for future work.

Controllable Text Generation. Our work is also related to a growing body of literature on controllable text generation [264, 354, 120, 152, 295]. In particular, our equation consistency constraint takes inspiration from the above works that impose similar constraints to improve control over the generation process. A major difference between our work and most of these prior works is that, in most of these approaches, the control elements, such as emotion, sentiment, and speaker identity, are usually represented as scalar numerical values. In contrast, our control elements (equation and context) consist of a sequence of math symbols or tokens rather than numeric values, which requires additional technical solutions to propagate gradient. The Gumbel-softmax trick [130, 209] that we employ has found success in text generation using generative adversarial networks (GANs) [170, 42, 238, 376, 135], a setting similar to ours where discrete sampling becomes an issue.

2.5 Conclusions and Future Work

In this chapter, we developed a controllable MWP generation approach that (i) leverages pre-trained language models to improve the language quality of the generated MWP, (ii) imposes an equation consistency constraint to improve mathematical consistency of the generated MWP, and (iii) includes a context selector that sets the context (in the form of a set of keywords) to use in the generation process. Experimental results on several real-world MWP datasets show that, while there is plenty of room for improvement, our approach outperforms existing approaches at generating mathematically consistent MWPs with high language quality.

Automatically generating MWPs remains a challenging problem and our work opens up many avenues for future work. First, our study is limited to the case of simple MWPs, each with a single equation and variable. While results are encouraging, our approach does not generalize well to the more challenging case when the input consists of multiple, complex equations. In these cases, we need more informative representations of the input equations [360]. Second, there is no clear metric that can be used to evaluate the generated MWPs, especially their mathematical validity. It is not uncommon when a generated MWP with high scores under our metrics is either unanswerable or inconsistent with the input equation. Therefore, future work should also focus on developing metrics for better evaluation of generated MWPs' mathematical validity. Last but not least, while we focus on

2 control elements (equation, context), an interesting future direction is to add more control elements to the generation process such as question difficulty and linguistic complexity.

Chapter 3

Towards Human-like Educational Question Generation with Large Language Models

3.1 Introduction

Practice questions and quizzes have been vital instruments for the assessment of learning [4, 202, 365]. Engaging in retrieval practice by answering expert-designed questions has shown to be more effective at improving learning outcomes [150, 151], by providing opportunities for recall of knowledge, applying knowledge to novel scenarios, and critical thinking and writing skills. The learning benefits are greater than other means of pedagogy such as passively re-reading course materials or studying notes [150, 151, 149, 58, 163, 159] or watching instructional videos [216]. However, these questions are also known to be challenging to create: they usually take subject matter experts (SMEs) a significant amount of time, which is both costly and labor-intensive [202]. Therefore, this question generation process does not easily generalize and scale to the continually expanding repositories of educational content that need large banks of assessments to be effective sources of instruction.

To create a scalable question generation process, several recent works leveraged artificial intelligence (AI) methods for *automatically* generating questions. For example, some prior works [293, 75, 361] focused on generating factual questions using recurrent neural network (RNN) architectures. [368] designed a method to select highly interesting phrases which a generated question is supposed to ask about. The implications of these works are far-reaching. In addition to reducing the labor and cost for producing assessment questions, automatic question generation methods have the potential to create a more engaging learning experience by generating (i) personalized questions that adapt to each student's learning trajectory [123] and (ii) real-time pop-up quizzes while the student is reading a textbook or watching instructional videos. Once trained, these methods have been shown to perform well on question generation tasks. However, they require custom model design and (sometimes significant) computational resources for training, making them a less appealing option for practitioners who desire a "plug-and-play" AI-assisted question generation process that allows them to easily interact with an AI system without the need for model training.

Recently, a new paradigm in text generation using large pretrained language models (PLMs), such as GPT-3 [30], is now making such "plug-and-play" question generation a possibility. These PLMs have been pretrained on web-scale data which equip the model with abundant knowledge of the language compared to their earlier counterparts. Furthermore, they can be easily and effectively adapted to various generation tasks via the "prompting" technique, where the user simply specifies the generation task that they would like to perform as a prompt. A *prompt* usually contains, in addition to a "query" from which the PLM will generate the outcome, a series of examples in an input-output structure that "teach" the model how to generate the output given the input specific to a particular task. Figure 3.1 gives an example of using prompting to adapt a PLM for machine translation and arithmetic question answering. Prompting provides an easy interface and high controllability for users to interact with PLMs and customize it for different generation tasks. Because of its simplicity and practicality, prompting techniques to adapt PLMs for downstream generation tasks have attracted increasing attention in the past few years [187, 201, 152, 196]. Figure 3.1 shows an example of prompting for machine translation, question answering.

Unfortunately, using prompts to adapt PLMs for question generation is challenging due to the open-ended nature of the process, i.e., it does not have a clearly defined input-output structure. This poses certain challenges such as, what content should the questions be generated from, how should we deal with the fact that multiple different questions can be asked about the same concept, etc. This open-ended nature makes question generation unique in contrast with other generation tasks commonly studied in existing literature (e.g., in machine translation, input and output are simply texts in the source and target languages, respectively). As a result, unlike other generation tasks where adapting PLMs via prompting is straightforward (e.g., see Figure 3.1 for an illustration), it is unclear how to design effective prompts for PLMs in order for question generation. To the best of our knowledge, to date no existing literature has investigated the modification of prompting strategy for question generation. To harness the power of AI for educational question generation, prompt design for question generation by PLMs is an exciting open problem.

3.1.1 Contributions

In this work, we investigate the problem of effectively prompting a PLM to generate desirable, high-quality, educational practice questions. An effective prompt strategy will enable us to leverage the power of PLMs with minimal effort and without having to conduct model training with large volumes of domain-focused content. We start with the core question: how do we design prompts such that a PLM can generate the most desirable and effective practice questions? We answer this question by proposing 5 different generation settings with a specific prompting strategy for each. We conduct a series of manual examinations of the generated questions as well as automatic evaluations, which lead to the empirical conclusion of the best combinations of our prompting strategy. This strategy serves as an empirical guideline for practitioners to set up PLMs to generate the best practice questions for educational purposes. Furthermore, we evaluated the educational value of PLM-generated

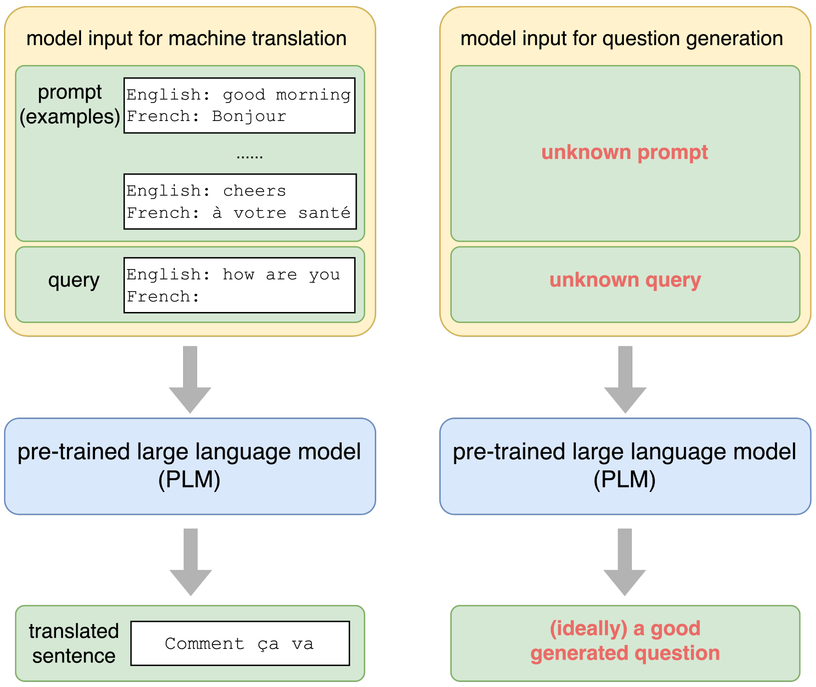


Figure 3.1 : Illustrations of adapting PLMs for machine translation and the challenges in designing prompts to adapt PLMs for educational question generation.

questions by presenting them alongside human-authored questions for SMEs to discern the human-authored from machine-authored questions. Evaluation by the respective SMEs (biology, psychology, and history) demonstrated that the generated questions achieved similar educational value relative to the human-authored ones, setting a strong case for their practical utility. In essence, we emulate how real practitioners and educators might be able to use these models to generate questions that meet their need in a practical setting.

3.1.2 Background: Large Pretrained Language Models and Prompting

We focus on large pretrained language models (PLMs) in this work, specifically, auto-regressive PLMs, such as GPT that have become the dominant tools for text generation. These models learn a distribution over text, which can be decomposed auto-regressively as follows:

$$\boldsymbol{x} \sim p_\theta(\boldsymbol{x}) = p_\theta(x_1) \prod_{t=2}^{T} p_\theta(x_t | x_1, \ldots, x_{t-1}) \,. \tag{3.1}$$

where p_θ is the LM where θ represents all model parameters. In this work, we focus on an LM that is already trained on massive data and thus assume p_θ is fixed throughout this chapter.

In practice, we will give the model some initial texts called a "prompt" as input which instructs the model to generate specific texts. This is possible because of the decomposition in Eq. 3.1. To see this, let $\boldsymbol{c} := [c_1, \ldots, c_L]$ denotes the prompt which consists of L ordered tokens c_l. Then the LM models a conditional distribution as follows:

$$p_\theta(\boldsymbol{x}|\boldsymbol{c}) = p_\theta(x_1|\boldsymbol{c}) \prod_{t=2}^{T} p_\theta(x_t | x_1, \ldots, x_{t-1}, \boldsymbol{c}) \,. \tag{3.2}$$

Eq. 3.2 makes it possible to adapt an LM for a wide range of generation tasks: depending on the interpretation of $\boldsymbol{c}$, we can adapt a pretrained LM for a wide range of tasks. [30] shows that, without further fine-tuning p_θ, simply changing $\boldsymbol{c}$ for different tasks perform on par with fine-tuning p_θ. This makes it very easy to use the LM because we only need to change the input to the model to adapt it for a variety of tasks. See Figure 3.1 for an illustration. The question now is how to design such a prompt for question generation.

3.2 Exploring Prompting Strategies in Question Generation

In the remainder of the chapter, we set out to answer the question: how do we design effective prompts for educational question generation? Answers to this question will provide practitioners with clear guidance on how to better control off-the-shelf PLMs for high-quality

Table 3.1 : Summary of the four factors in our prompting strategy and the choices under consideration for each factor.

Example structure for question generation	**Data source in the examples**	**Number of examples**	**Lengths of context and question in each example**
CAQ: context (C) and an answer (A) and the output contains a question (Q)	Content agnostic (SQuAD)	One-Shot	Small (avg. 15 words)
CTQA: (C) and a target (T) and the output contains a question (Q) and an answer (A)	Content specific	Few-Shot	Medium (avg. 25 words)
		Five-Shot	Large (40 and above)
		Seven-Shot	

question generation. We take an empirical approach and design a series of experiments to systematically investigate various factors that impact the effectiveness of prompting strategies for question generation with PLMs. We propose four factors that are crucial considerations to prompt design for question generation. Below, we detail these factors and the possible choices that we study for each factor (see Table 3.1 for a high-level summary). In contrast to automated prompting methods as in existing literature, our prompting design is interpretable and flexible, enabling practitioners to explicitly control and iteratively refine the generation process as needed.

3.2.1 Example structure for question generation

The first factor we investigate is the question generation formulation, i.e., the input-output structure in each example that we will use to instruct and adapt the PLM for question generation. Different formulations will likely impact the generated questions' quality. In this work, we focus on contextualized question generation, in which a question is asked and the answer to it can be found within a given paragraph. We compare two different generation setups. In the first setup, labeled as CAQ, the input contains a context (C) and an answer (A) and the output contains a question (Q). The context can be a short excerpt from a textbook and the answer should correctly answer the generated question. This setup has been considered in a wide range of question generation tasks [361, 75, 368]. In the second setup, referred to as CTQA, the input contains a context (C) and a target (T) and the output contains a question (Q) and an answer (A). The target does not need to be the answer to the generated question but guides the model to generate a question to ask *about* the particular part in the context specified by the target. The model also generated an answer in addition to the question. The intuition behind this setup is that the model may generate more on-topic and relevant questions because it is forced to also generate the answer. This setup is reminiscent of prior work that leverages question answering modeling for question generation [76, 189].

3.2.2 Data source in the examples

The second factor we investigate is the data source in each example, i.e., where do the context, question, answer (target) come from? This question arises when a user wants to generate questions for different subjects; depending on the subject, the examples in the prompt may need to change so that PLM is given the appropriate domain knowledge. We are most interested in whether we can use the same set of examples that come from a generic source for question generation across different subjects/content. We thus compare a *content-agnostic and a* content-specific selection of examples. In the content-agnostic setup, we choose examples from SQuAD [270], a generic, widely used question answering dataset that can also be used for question generation. In the content-specific setup, we choose examples in the same subject as the one in which the PLM will generate questions.

3.2.3 Number of examples

The third factor we investigate is the number of examples to include in the prompt. Usually, PLMs' performance improves with more examples. Nevertheless, because of the open-ended nature of question generation, it is unclear to what point increasing the number of examples will help. We thus consider four setups including One-shot, Few-shot, Five-shot, Seven-shot where "shot" refers to the number of examples.

3.2.4 Lengths of context and question in each example

The last factor we investigate is the length of context and question in each example. A context or question that is too short may limit the diversity and complexity of the generated questions. A context or question that is too long may contain irrelevant information which may confuse the PLMs, potentially leading to generated questions that are irrelevant or off-topic. We thus compare three different setups including small, medium, large contexts and questions depending on the length of texts they contain. Small corresponded to questions about 15 words in length, medium questions were around 25 words long, and large questions were about 40 words long on average. Small contexts consist of around 2 sentences, medium contexts around 4-5 sentences of information, and large contexts usually a full paragraph or multiple paragraphs.

3.3 Experiments

We recommend the best prompt setting for each generation strategy that yielded the best-generated questions. Code scripts, additional clarifications, and additional results such as examples of generated questions are publicly available. [1]

[1] `https://github.com/openstax/research-question-generation-gpt3`

Experiment Setup

We choose biology as the subject to generate questions and use the Openstax Biology 2e (Bio 2e) Textbook as the source for most of our example content. In this work, we focus on generating open-ended questions of Bloom's level below three because higher-order Bloom's questions typically involve making connections across larger content [24, 164]. Generating diverse types of potentially more challenging questions is left for future work. We also limit our investigation to textual content and remove images, tables, links, and references from the textbook. During generation, we first pre-select a fixed number of examples from the textbook (and SQuAD, for the data source experiment; see Section 3.2.2). During generation for all setups under each factor, we randomly pick a fixed number of examples to serve as the prompt and another two queries, i.e., with only the context (possibly also the target; see Section 3.2.1) from which the PLM is asked to generate questions. Unless otherwise noted, for each query in each setup under each factor, the PLM generates 75 questions for evaluation. When generating questions for a factor, all the other factors are set to the same value to ensure fairness in comparison. Throughout our experiments, we use the GPT-3 Davinci API from OpenAI with temperature = 0.9 and top_p = 1.

Evaluation Protocol

We primarily evaluate the quality and diversity of the generated questions. For quality, we report **perplexity** and **grammatical error**. Perplexity is inversely related to the coherence of the generated text; the lower the perplexity score, the higher the coherence. To make the process computationally efficient, we computed perplexity using a GPT-2 language model for all generations. We computed grammatical error using the Python Language Tool [231] which counts the number of grammatical errors averaged over all generated questions in each setup under each factor. For diversity, we report the **Distinct-3** score [185], which counts the

Table 3.2 : Results for the example structure comparisons, which show that the CTQA structure is distinctly better than the CAQ structure.

Gen. Format	diversity ↑	perplexity ↓	toxicity ↓	gramm. error ↓	% acceptable ↑
CAQ	0.895	64.683	0.153	**0.053**	26.7%
CTQA	**0.898**	**29.900**	0.153	0.080	**54.7%**

average number of distinct 3-grams in the generated questions. Furthermore, we believe that ensuring the generated questions are safe, i.e., without profanity or inappropriate language is critical for high-stakes educational applications. Therefore, we report the **toxicity** of the generated questions, using the Perspective API [256], which is often missing from the evaluation in existing question generation literature. Last but not least, we perform a preliminary human evaluation to mark **percentage of acceptable questions** for each setup under each factor. A question is considered acceptable if it is coherent, on-topic, answerable, grammatically correct, and appropriate. We conduct a more comprehensive human evaluation in Section 3.3.3.

3.3.1 Empirical Observations

Structure of Examples in the Prompt

Recall that this experiment compared CAQ and CTQA structures of the examples in the prompt (Section 3.2.1). The results, presented in Table 3.2, show that, although the CTQA structure produces questions of comparable diversity, quality, and toxicity, it generates about twice as many acceptable questions as the CAQ structure. This comparison suggests that CTQA is a superior example structure and confirms our earlier hypothesis that asking PLMs to generate the answer in addition to only the question is beneficial for improving the quality of generated questions. Additionally, the generated answers can be potentially useful for evaluating a student's performance on the generated question. Ensuring that the generated

Table 3.3 : Results for the example data source comparisons. Using content specific examples gives superior generation performance compared to content agnostic example.

Gen. Format	diversity ↑	perplexity ↓	toxicity ↓	gramm. error ↓	% acceptable ↑
SQuAD	0.884	102.840	0.201	0.093	18.0%
OpenStax	**0.895**	**64.683**	**0.153**	**0.053**	**26.7%**

Table 3.4 : Results for the number of examples comparisons. five- and seven-example settings yield better questions compared to one- and three-example settings.

# Examples	diversity ↑	perplexity ↓	toxicity ↓	gramm. error ↓	% acceptable ↑
1 example	0.897	37.954	0.384	0.182	24.9%
3 examples	0.924	36.586	0.232	0.151	37.8%
5 examples	**0.938**	35.990	0.208	0.119	**51.6%**
7 examples	0.918	**30.731**	**0.176**	**0.076**	44.9%

answer correctly answers the generated question is important ongoing work.

Data Source in Examples

Recall that this experiment compared whether the examples come from the same subject (Bio 2e) as the query or a generic dataset (SQuAD) (Section 3.2.2). The results in Table 3.3 showed that when a prompt consists of examples from the same subject, the PLM can generate questions about twice as effective as when using SQuAD examples across all metrics. These results suggest that a generic set of examples may not adapt to question generation for various domains and that appropriately choosing examples from desired subjects is a better setup for question generation.

Number of Examples

Table 3.4 shows the results comparing one-, three-, five-, and seven-shots, i.e., the number of examples in the prompt. The results show that one- and three-shots are ineffective; we

Table 3.5 : Results for the context and question length comparisons. We see that, in general, short context and question lengths in the examples improve generation quality.

Context Length	diversity ↑	perplexity ↓	toxicity ↓	gramm. error ↓	% acceptable ↑
Short	0.861	33.452	0.329	**0.380**	22.0%
Medium	**0.878**	30.692	**0.214**	0.410	**24.0%**
Long	0.877	**30.385**	0.331	0.420	**24.0%**
Question Length	**diversity ↑**	**perplexity ↓**	**toxicity ↓**	**gramm. error ↓**	**% acceptable ↑**
Short	**0.906**	34.275	**0.246**	**0.377**	**30.0%**
Medium	0.893	33.704	0.318	0.487	23.7%
Long	0.885	**30.38**	0.295	0.610	14.7%

observe that they produce a majority of unacceptable questions. The five-shot condition results were optimal followed closely by the seven-shot, with the one-shot being most inefficient. We prefer using the five-shot condition because here, the PLM generated more varied questions that are also of high quality. For example, although the model was only given free-response questions, it could produce a small number of multiple-choice or true-or-false questions.

Lengths of Context and Question in each Example

Table 3.5 shows the results comparing different lengths of the question and context in each example, respectively. In terms of question lengths, results suggested that a smaller question length generally yields the best performance. In terms of context lengths, results are mixed. This is likely because longer contexts contain information that is not directly useful for generating questions and because longer texts lead to longer prompts, which makes it more difficult to instruct the model to adapt to the question generation task.

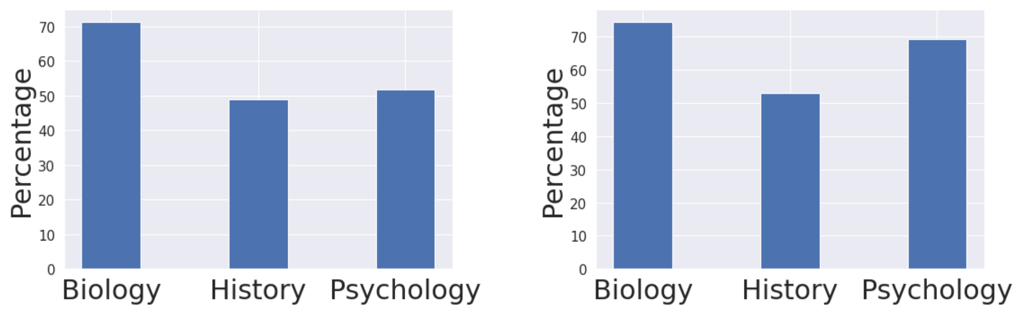


Figure 3.2 : Human evaluation results. **Left**: the percentage of PLM-generated questions that are recognized as human-authored by SMEs. **Right**: the percentage of PLM-generated questions that SMEs considered as ready-to-use in their classes.

3.3.2 Discussions

From the above quantitative results, we obtain a good understanding of how the different choices, while constructing the prompt for each generation strategy, will impact the quality of the generated questions. It is clear that when preparing examples to instruct and adapt PLMs for question generation, the PLM is likely to generate higher quality questions given the prompt design: if prompt contains five to seven examples that are in CTQA format, are chosen from the desired subject, rather than generic content, and contain relatively short contexts and questions. This recommendation has the potential to serve as a guideline for practitioners when adapting off-the-shelf PLMs for their unique question generation needs.

3.3.3 Human Expert Evaluation for Multiple Subjects

To validate the utility of the generated questions as well as to investigate whether our best prompt strategy would result in good question generations across domains (e.g., natural sciences, social sciences, and humanities), we engaged biology, psychology, and history subject matter experts (SMEs) respectively to evaluate the quality of questions from these domains generated using the best prompting strategy.

For each domain, we chose 5 examples as the prompt and another 5 examples with only the question and target as the query. For each query, we generated and selected the 10 best questions. We then mixed these 50 questions together with 5 reference human-authored questions (one for each query) and presented them to SMEs. SMEs were asked to (1) distinguish whether each question is machine or human authored and (2) evaluate whether they would use the question in the assessment they create for their class.

The results, shown in Figure 3.2, suggest that the majority of PLM-generated questions look "human-like" for SMEs and can be readily used in their respective classes. This result is encouraging because it suggests that our prompting strategy can effectively adapt a publicly available, general-purpose text generation model for high-quality educational question generation. Table 3.6 demonstrates a few successful generated questions.

3.3.4 Limitations and Future Work

Question generation is complex and is an active ongoing area of research. In this work, we proposed four factors as the prompt design framework and conducted a controllable scientific study on those factors. However, our investigation was by no means exhaustive and several other factors have the potential to impact question generation such as question difficulty, type (i.e., multiple-choice, fill in the blank, etc.), and modality (i.e., potentially including tables and equations). Investigations of more factors and complex interactions among them are left for future work. In addition, our human evaluation was a small-scale experiment because we were only able to engage the SMEs for a short time. The next step is to conduct a large-scale evaluation that involves both instructors and students in a safe environment to obtain a better understanding of the educational utility of machine-generated questions. Lastly, our prompting strategy generated questions with grammatical errors and other problems at times; we show some failed examples in Table 3.7. A promising

Table 3.6 : Examples of three generated questions using our recommended prompting strategy along with the context, target, and reference human-authored questions.

Context (Biology): At 0.1 to 5.0 μm in diameter, prokaryotic cells are significantly smaller than eukaryotic cells, which have diameters ranging from 10 to 100 μm (Figure 4.6). The prokaryotes' small size allows ions and organic molecules that enter them to quickly diffuse to other parts of the cell. Similarly, any wastes produced within a prokaryotic cell can quickly diffuse. This is not the case in eukaryotic cells, which have developed different structural adaptations to enhance intracellular transport. **Target**: size **Reference human-authored question**: Some unusually large cells do exist. How might such cells solve the problem of diffusion? **Machine-generated question**: Compare the size of prokaryotic and eukaryotic cells to describe why diffusion is more rapid through prokaryotic cells.
Context (Psychology): Sleep debt and sleep deprivation have significant negative psychological and physiological consequences. As mentioned earlier, lack of sleep can result in decreased mental alertness and cognitive function. In addition, sleep deprivation often results in depression-like symptoms. These effects can occur as a function of accumulated sleep debt or in response to more acute periods of sleep deprivation. It may surprise you to know that sleep deprivation is associated with obesity, increased blood pressure, increased levels of stress hormones, and reduced immune functioning. **Target**: sleep deprivation **Reference human-authored question**: Which animal would be most likely to die from a lack of sleep? **Machine-generated question**: Sleep debt and sleep deprivation are associated with significant negative psychological and physiological consequences. What are some of the potential consequences of a person having less than optimal sleep?
Context (U.S. History): Opponents made clear their resistance to Garrison and others of his ilk; Garrison nearly lost his life in 1835, when a Boston anti-abolitionist mob dragged him through the city streets. Anti-abolitionists tried to pass federal laws that made the distribution of abolitionist literature a criminal offense, fearing that such literature, with its engravings and simple language, could spark rebellious Black people to action. Their sympathizers in Congress passed a "gag rule" that forbade the consideration of the many hundreds of petitions sent to Washington by abolitionists. A mob in Illinois killed an abolitionist named Elijah Lovejoy in 1837, and the following year, ten thousand protestors destroyed the abolitionists' newly built Pennsylvania Hall in Philadelphia, burning it to the ground. **Target**: anti-abolitionists **Reference human-authored question**: How did anti-abolitionists respond to the abolition movement? **Machine-generated question**: What were some of the responses to the abolitionist movement?

Table 3.7 : Examples of failed cases and the failing reasons. Our prompting strategy can still generate questions that contain grammatical errors and other types of errors.

(Biology): What is the correct statement is about centrosomes? (Multiple-choice question with no options and bad grammar)
(Psychology): Sleep deprevation can lead to serious changes in the body. Which one of these changes characterized by sleep deprivation? (grammatical and spelling errors)
(History): During the Gold Rush, the Forty-Niners did not find wealth so easy to come by, most did not. (not a question)

future direction is to develop automated filters capable of removing undesirable generated questions and only select the highest quality ones, preferably also personalized to each student and instructor.

3.4 Conclusion

In this work, we investigate the best practices to prompt a PLM for educational question generation. We develop and empirically study a prompting strategy consisting of four different factors. Based on a series of quantitative experiments, we recommended the choices for each factor under our prompting strategy that led to high-quality generated questions. Human evaluations by subject experts in three different educational domains suggest that most of the questions generated by a PLM with our recommended prompting strategy are human-like and ready-to-use in real-world classroom settings. Our results indicate that properly prompting existing off-the-shelf PLMs is a promising direction for high-quality educational question generation with many exciting future research directions.

Chapter 4

Scientific Formula Retrieval via Tree Embeddings

4.1 Introduction

Recent years have seen increasing proliferation of *scientific formulae* as a data format, c.f. Table 4.1. With its unique set of symbols and language structure, scientific formulae complement natural language in concisely and precisely communicating essential scientific knowledge. These formulae are also an indispensable part of an ever-growing scientific corpus. However, the large quantity of these formulae also poses challenges for effectively organizing and synthesizing scientific formulae in order to derive new knowledge and insights. An important and common real-world use case is *formula retrieval*, i.e., finding relevant formulae similar to a query formula (e.g., [66]). This scenario arises when, for example, researchers search for related work in a large collection of scientific papers or when students look for relevant practice problems in a textbook when doing algebra homework. Scientific formula retrieval is labor-intensive and time-consuming for humans, making an automatic method highly beneficial.

An emerging line of research for automatic scientific formula retrieval leverages the *symbolic tree* representation of scientific formula since they often have an inherent hierarchical structure that can be well-captured by trees [393]. Compared to representing a formula simply as a *sequence* of symbols, the symbolic tree representation has the advantage to encode both the semantic and structural properties of a formula. several recent works incorporate symbolic tree representations, leading to improved performance on the formula

Table 4.1 : Examples of scientific formulae in various domains.

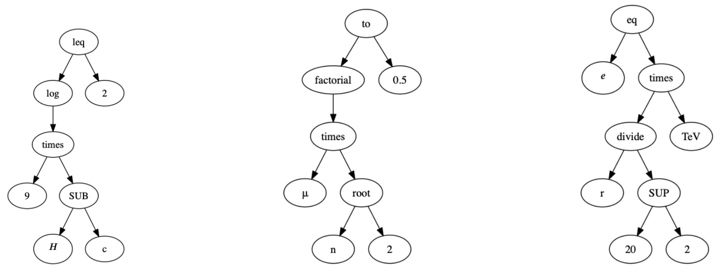
$N = \left\lfloor 0.5 - \log_2 \left(\frac{\text{Frequency of this item}}{\text{Frequency of most common item}} \right) \right\rfloor$	(physics)
${}^{238}_{92}\mathrm{U} + {}^{64}_{28}\mathrm{Ni} \to {}^{302}_{120}\mathrm{Ubn}^{*} \to \textit{fission only}$	(chemistry)
$ax^2 + bx + c = 0$	(algebra)
$O(mn \log m)$	(comp. sci.)
$A \oplus B = (A^c \ominus B^s)^c$	
$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cosh\frac{a}{k}$	
$\tau_{\text{rms}} = \sqrt{\frac{\int_0^\infty (\tau-\overline{\tau})^2 A_c(\tau)d\tau}{\int_0^\infty A_c(\tau)d\tau}}$	

retrieval task [66, 407, 406] over other formula representations, e.g., [86]. However, most of the above approaches are data-independent, i.e., they perform retrieval based on a set of user-defined rules and are thus not capable of *learning* representations from the large collection of scientific formulae to further improve retrieval performance. [211] is one of the few data-driven formula retrieval methods to date, which demonstrate the benefit of *learned* formula representations for formula retrieval compared to data-independent methods. However, [211] does not fully leverage the tree structure since it linearizes formula trees into sequences and then trains a sequential language model on them.

An additional desirable property of a learned formula representation is that it enables interesting applications beyond formula retrieval such as automatically generating content that involves formulae. For example, [387] uses a learned formula representation to generate a textual description for this formula. [390] automatically generates a headline that summarizes the substance of a mathematical question. However, these approaches treat scientific formulae simply as sequences of symbols and thus ignore their inherent hierarchical structure. [171] validates their rule-based representations for discrete data using a math formula generation task. However, their experiment involves only simple, synthetically-generated math equations with up to one variable, two math functions (sinusoid and exponential), and three numbers. This level of complexity is dwarfed by that of real-world scientific formulae [171]; therefore, it is unclear whether their approach is applicable in this setting. Furthermore, the above methods are *supervised* and therefore must depend on large, *labeled* datasets which are challenging to construct. Therefore, a desirable scientific formula representation learning method should be unsupervised, take advantage of the inherent formula structure, and enable both retrieval and generation.

4.1.1 Contributions

We propose **FORTE**, a novel *unsupervised* framework for scientific **FO**rmula **R**epresentation learning via **T**ree **E**mbeddings, capable of both similar formula retrieval and formula generation. Our framework fully exploits the tree structure of scientific formulae in an autoencoding model design to learn an effective representation. FORTE consists of two key components. First, a *tree encoder* encodes a formula in operator tree format into an embedding vector that can be used for various downstream tasks, including similar formula retrieval. Second, a *tree decoder* reconstructs a formula tree from its encoded embedding vector. The decoder can also generate novel, unseen formula trees from any input vector. To improve generation quality, we also propose a novel *tree beam search* method that extends the classic beam search method for sequential data to improve formula generation by keeping multiple trees (possibly with different structures) at every search step. To evaluate our framework, we combine and parse several existing datasets collected from real-world sources (such as Wikipedia and arXiv articles) into a dataset with over 770k scientific formulae. To the best of our knowledge, this dataset is the largest one to date. We conduct extensive quantitative and qualitative experiments on this dataset for both formula reconstruction and similar formula retrieval. Experimental results show that FORTE



(a) $\log(9 \times H_c) \leq 2$ (b) $(\mu \times \sqrt[2]{n}) \rightarrow 0.5$ (c) $e = \frac{r}{20^2} \times \mathrm{TeV}$

Figure 4.1 : Examples of simple formulae and their corresponding OTs. These examples are randomly generated by our proposed FORTE framework; see Section 4.3.2.

(sometimes significantly) outperforms various existing methods.

4.2 The FORTE Framework

We now detail the FORTE framework. We first introduce tree representations of formulae, especially operator trees, which is a crucial pre-processing step in our framework. We then set up the formula representation learning problem and introduce the various FORTE components, including the tree encoder, the tree decoder, and the tree beam search algorithm for formula generation. Figures 4.2–4.4 together provide a high-level overview of our framework.

4.2.1 Preliminary: Formulae As Operator Trees

A scientific formula is inherently tree structured [394, 66] and can be represented as a symbolic *operator tree* (OT) that we denote as X:

$$X = (U, <) \in S, \quad u \in U, \quad U \subseteq V \tag{4.1}$$

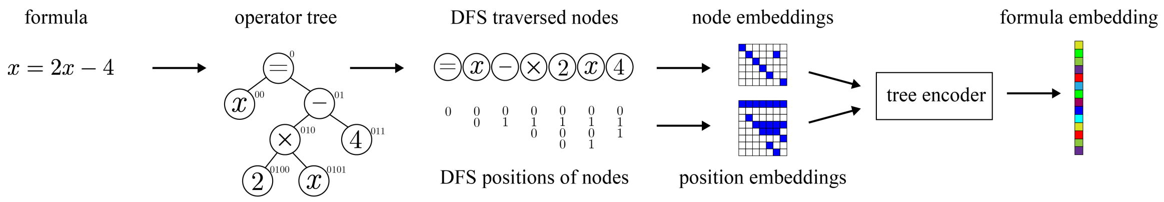


Figure 4.2 : Illustration of FORTE's encoding process of a formula.

where u is a scientific (most often math) symbol that corresponds to a node in the OT. Throughout the chapter, we will refer to "symbol", "scientific symbol", and "node" interchangeably depending on context. U is the set of symbols in X, and V is the "vocabulary", i.e., all unique scientific symbols in the data set. $<$ represents partial binary parent-child relation $\forall u \in U$ [168]. S is the set of all valid OTs. Intuitively, the OT organizes the scientific symbols in a formula, such as math operators, variables, and numerical values, as *nodes* in an explicit tree structure. We remark that OT is not the only tree representation of scientific formulae and there exist other formula tree representations such as the symbol layout tree [66]. In principle, our FORTE framework is agnostic to the underlying tree representation; we choose OT because of its intuitive interpretation and ability to preserve the semantic and structural information in scientific formulae. Figure 4.1 gives a few simple examples of formulae and their corresponding OTs.

4.2.2 Problem Formulation

We set up the formula representation learning problem as an unsupervised "autoencoding" task, motivated by the downstream tasks that we envision our framework will perform, including formula generation and retrieval. Specifically, our framework aims to reconstruct the input formula in its OT representation through an encoder-decoder design with a bottleneck embedding vector. This problem setup enables us to use the latent embedding obtained from the encoder for many downstream tasks, i.e., formula retrieval, and the generated formula from the decoder for generation-related tasks.

Given our autoencoding task formulation, we use a model structure consisting of an encoder and a decoder, where the encoder encodes a scientific formula into a vector representation and the decoder reconstructs a scientific formula from its vector representation. Concretely, we have

$$\begin{aligned} \text{encode}: &\quad \boldsymbol{e} = f_{\text{enc}}(X; \Theta)\,, \\ \text{decode}: &\quad X_{\text{out}} \sim p_{\text{dec}}(\boldsymbol{e}; \Phi)\,, \end{aligned}$$

where $\boldsymbol{e} \in \mathbb{R}^M$ is the M-dimensional vector representation of a formula. f_{enc} and p_{dec} are the encoder and decoder models parametrized by a set of parameters Θ and Φ, respectively. We will detail their specific forms in Sections 4.2.3 and 4.2.4. X_{out} denotes the formula tree for the decoder, which is slightly modified from X. The reason that we need a different tree format for the decoder will become clear when we detail our decoder design in Section 4.2.4.

Optimization

Given our formulation and a dataset of N scientific formulae, we train f_{enc} and p_{dec} jointly to maximize the scientific formula reconstruction accuracy by minimizing the cross entropy loss

$$\mathcal{L}(\Theta, \Phi) = \frac{1}{N} \sum_{i=1}^{N} -\log p_{\text{dec}}(f_{\text{enc}}(X^{(i)}; \Theta); \Phi)\,. \tag{4.2}$$

i is the index of formulae, which we will drop for notation simplicity whenever we discuss a single formula.

Formula Retrieval and Generation

In formula retrieval, we are given a query formula X_{q} and a collection $\mathcal{D}$ of candidate formulae $X_{\mathrm{r}} \in \mathcal{D}$ for retrieval. We will first compute the embeddings of the query and the candidate formulae and then select a subset $\mathcal{G} \subset \mathcal{D}$ consisting of those that are most similar (e.g., measured by cosine similarity scores). Concretely, we have:

$$\mathcal{G}{=}\Big\{X_r^{(i)}{\in}\mathcal{D} \;\Big|\; |\{X_r^{(i)\prime}{\in}\mathcal{D}{:}\,\mathrm{sim}(\boldsymbol{e}_r,\boldsymbol{e}_q){<}\mathrm{sim}(\boldsymbol{e}_r',\boldsymbol{e}_q)\}| < R\Big\}$$

where $\boldsymbol{e}_r$ and $\boldsymbol{e}_q$ are the embeddings for candidate retrieval X_r and query X_q, respectively. $\mathrm{sim}(\boldsymbol{e}_r, \boldsymbol{e}_q)$ is the cosine similarity function

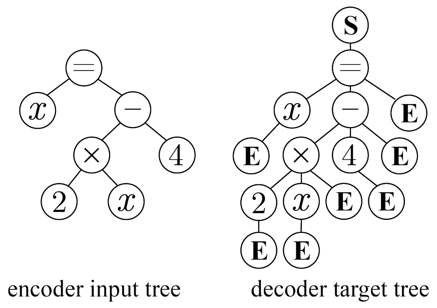
$$\mathrm{sim}(\boldsymbol{e}_r, \boldsymbol{e}_q) = \frac{\boldsymbol{e}_r^\top \boldsymbol{e}_q}{\|\boldsymbol{e}_r\|\|\boldsymbol{e}_q\|} \,.$$

In formula generation, given an input vector $\boldsymbol{e}$ (not necessarily a formula embedding), the decoder generates a formula X' with the highest log-likelihood:

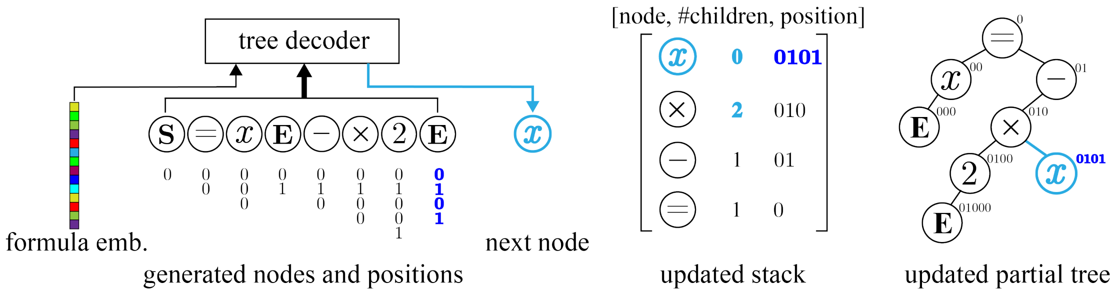
$$X' = \underset{X_{\mathrm{out}}}{\mathrm{argmax}} \; \log p_{\mathrm{dec}}(X_{\mathrm{out}} \,|\, \boldsymbol{e}; \Phi) \,.$$

4.2.3 Formula Tree Encoder

Recall that our *tree encoder* takes a formula tree X as input and outputs an embedding of this formula $\boldsymbol{e}$. The key idea is to properly encode all information underlying the formula tree. To this end, we use two methods including *tree traversal*, which extracts content information (the symbol each node corresponds to), and *tree positional encoding*, which extracts structural information (the relative positions of nodes). These methods are inspired by prior works in program translation [48, 301] and natural language processing [356, 237]



(a) Input and output formula trees.



(b) FORTE's decoding process.

Figure 4.3 : (**4.3a**) Illustration of FORTE's input and output operator trees of the same formula. The "S" node represents the special "`<start>`" node at the root of the tree. The "E" nodes represent the special "`<end>`" node attached as the last child to every node. (**4.3b**) Illustration of FORTE's decoding process at a particular time step. First, the position of the next node to be generated is computed (dark blue). Next, the next node (light blue) is generated by the decoder using already generated nodes and positions and the newly computed position. Finally, the partial tree and the stack are updated.

that achieve state-of-the-art performance. These works involve traversing structured data (e.g., the compiler stack of a program, the parse tree of a sentence) and keeping the traversal order, which resembles what we employ in our work. Figure 4.2 provides an overview of our formula tree encoder.

Formula Tree Traversal and Node Embedding.

We traverse each formula tree in the depth first search (DFS) order to extract the node symbols. This step returns a DFS-ordered list of nodes $\{u_t\}_{t=1}^{T}$ where t indexes the nodes

u's in the DFS order and T is the total number of nodes in the formula tree. Each node u_t is then represented as a trainable embedding $\widetilde{\boldsymbol{x}}_t \in \mathbb{R}^M$.

Tree Positional Encoding.

To extract the structure of a formula tree, we propose a two-step method that first computes and then embeds the relative positions of nodes in the tree.

Let v be the parent of node u. Let q_v and q_u be the positions of v and u, respectively. Let n_u denote that u is the n-th child of v from left to right; n starts with 0. In the first step, we compute the position q_u of each node as follows:

$$q_u = 10q_v + n_u\,. \tag{4.3}$$

When u is the root node, we set q_u = 0. The above construction is intuitive and informative because i) the number of digits in the biggest q_u in a tree represents the depth of this tree and ii) the largest number of all q_u in a tree represents the maximum degree of this tree. The formula OT in Figure 4.2 illustrates the above computation. For example, the position 011 of the numeric node "4" is composed of 01 which is its parent's position and 1 because it is the second child of its parent.

The numeric values of different positions p_u may differ significantly, i.e., between 0 of the root node and tens of thousands of a leaf node in trees that are deep. Therefore, in the second step, we embed these positions p_u into fixed-dimensional *tree positional embeddings*. We propose a binary tree positional embedding method where each digit in p_u is converted to its corresponding binary number in the base-2 numeric system and then concatenated together. Concretely, let $p_u = [p_{u,1}, \ldots, p_{u,l}]_{10}$ be the base-10 representation of p_u where $l = \lfloor \log_{10}(p_u) \rfloor$ is the number of digits in p_u and $p_{u,j}$ is the j-th digit in p_u from left to right.

Let $\boldsymbol{q}_u$ be the embedding of q_u. Then:

$$\widetilde{\boldsymbol{q}}_u = \left[\operatorname{bin}(p_{u,1})^\top, \dots, \operatorname{bin}(p_{u,l})^\top\right]^\top \in \mathbb{R}^{D_u}\,, \tag{4.4}$$

$$\boldsymbol{q}_u = \left[\widetilde{\boldsymbol{q}}_u^\top, \boldsymbol{0}_{D-D_u}^\top\right]^\top \in \mathbb{R}^{D}\,, \tag{4.5}$$

where $\operatorname{bin}(\cdot)$ is the binarization operator (e.g., $\operatorname{bin}(5) = 101$). $D = L\log_2(C)$ where L and C are the maximum depth and maximum degree of all formula trees under consideration. $\boldsymbol{0}_{D-D_u} \in \mathbb{R}^{D-D_u}$ is an all-zero vector to make the dimension of every $\boldsymbol{q}_u$ the same.

Formula Tree Embedding.

To transform the formula tree into its embedding, we utilize an embedding function $f_{\text{enc}} : \mathbb{R}^{(M+D)\times T} \to \mathbb{R}^K$ where K is the dimension of the formula tree embedding and T is the total number of nodes in a formula tree. We concatenate the node and tree positional embeddings such that the encoder is aware of both the nodes and their positions. The concatenation setup enables more modeling flexibility because we do not need to enforce $M = D$. Concretely, the formula tree embedding is:

$$\boldsymbol{e} = f_{\text{enc}}(\{\boldsymbol{x}_t\}_{t=1}^T; \Theta), \qquad \text{where } \boldsymbol{x}_t = \left[\widetilde{\boldsymbol{x}}_t^\top, \boldsymbol{p}_t^\top\right]^\top. \tag{4.6}$$

In this work, we use a bidirectional gated recurrent unit network (bi-GRU) [53] that recurrently computes a hidden state $\boldsymbol{h}_t$ for each $\boldsymbol{x}_t$. The forward direction is computed as follows:

$$\overrightarrow{\boldsymbol{z}_t} = \sigma(\mathbf{W}_z \big[\widetilde{\boldsymbol{x}}_t^\top, \boldsymbol{p}_t^\top\big]^\top + \mathbf{U}_z \overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_z)\,, \tag{4.7}$$

$$\overrightarrow{\boldsymbol{r}_t} = \sigma(\mathbf{W}_r \big[\widetilde{\boldsymbol{x}}_t^\top, \boldsymbol{p}_t^\top\big]^\top + \mathbf{U}_r \overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_r)\,, \tag{4.8}$$

$$\overrightarrow{\boldsymbol{c}_t} = \sigma(\mathbf{W}_c \big[\widetilde{\boldsymbol{x}}_t^\top, \boldsymbol{p}_t^\top\big]^\top + \overrightarrow{\boldsymbol{r}_t} \cdot \mathbf{U}_c \overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_c)\,, \tag{4.9}$$

$$\overrightarrow{\boldsymbol{h}_t} = \overrightarrow{\boldsymbol{z}_t} \cdot \overrightarrow{\boldsymbol{h}}_{t-1} + (1 - \overrightarrow{\boldsymbol{z}_t}) \cdot \overrightarrow{\boldsymbol{c}_t}\,, \tag{4.10}$$

where $\mathbf{W}$, $\mathbf{U}$, and b are part of the set of parameters Θ. The backward direction $\overleftarrow{\boldsymbol{h}_t}$ is computed similarly, with the same Θ. The final formula embedding $\boldsymbol{e}$ is a simple weighted combination of the latent states $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$:

$$\boldsymbol{e} = \sum_{t=1}^{T} a_t \boldsymbol{h}_t\,, \qquad \boldsymbol{a} = \mathrm{softmax}(\mathbf{W}_a^\top [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T])\,,$$

where $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_t}^\top, \overleftarrow{\boldsymbol{h}_t}^\top]^\top$ and $\mathbf{W}_a$ is also part of Θ.

4.2.4 Formula Tree Decoder

The decoder takes a formula embedding vector, i.e., the output from our tree encoder, as input and generates a formula in OT format. In contrast to decoders often used in NLP that generate a sequence of symbols as output, we develop a decoder that generates symbols laid out in a tree. Our tree generation strategy leverages the fact that one only needs to know all symbols in the tree and all symbols' positions in the tree to perform reconstruction. Using this insight, our decoder first computes the next node's position and then generates the next node symbol at a given time step. We first describe node symbol generation and node position computation in the context of greedy tree generation. We then propose a tree beam search algorithm that extends and improves greedy tree generation.

Computing the Node Positions

The key difference between the decoder and the encoder regarding the tree positional embedding: in the decoder, the tree position embedding at time step t is *not* for the node at time step t (recall Eqs. 4.7–4.10) but rather for the node at time step $t+1$. The reason for this design is that, unlike the encoder that has access to all positions for all nodes in the input formula tree, the decoder has no positional information and needs to compute the positions for all nodes during the generation process in addition to generating the node symbols themselves. The $t+1$-th node's position is computed as Eq. 4.3, using its parent's position, and its parent's current number of children. Knowing a node's parent during generation requires maintaining the structure of the current, partially generated tree, which we detail in Sections 4.2.4 and 4.2.4.

Generating the Node Symbols

For node symbol generation, we use a causal, uni-directional GRU network, in which the hidden state at each recurrent step is computed as:

$$\begin{aligned}
\boldsymbol{z} &= \sigma(\mathbf{W}_z \big[\widetilde{\boldsymbol{x}}_t^\top, \boldsymbol{p}_{t+1}^\top, \boldsymbol{x}^\top\big]^\top + \mathbf{U}_z \boldsymbol{s}_t + b_z)\,, \\
\boldsymbol{r} &= \sigma(\mathbf{W}_r \big[\widetilde{\boldsymbol{x}}_t^\top, \boldsymbol{p}_{t+1}^\top, \boldsymbol{x}^\top\big]^\top + \mathbf{U}_r \boldsymbol{s}_t + b_r)\,, \\
\boldsymbol{c} &= \sigma(\mathbf{W}_c \big[\widetilde{\boldsymbol{x}}_t^\top, \boldsymbol{p}_{t+1}^\top, \boldsymbol{x}^\top\big]^\top + r \cdot \mathbf{U}_c \boldsymbol{s}_t + b_c)\,, \\
\boldsymbol{s}_{t+1} &= z \cdot \boldsymbol{s}_t + (1 - z) \cdot \boldsymbol{c}\,. \\
\boldsymbol{y}_{t+1} &= \mathrm{softmax}\left(\mathbf{W}_y \boldsymbol{s}_t + b_y\right)\,.
\end{aligned}$$

Here, t denotes the time step in DFS order, which is consistent with the encoder's node traversal order. $\widetilde{\boldsymbol{x}}_t$ and $\boldsymbol{s}_t$ are the embedding of the generated node and the decoder hidden state at the t-th time step, respectively. We concatenate the embedding of the input formula tree $\boldsymbol{e}$ at each step of the generation to inform the decoder and guide the generation towards the formula tree corresponding to $\boldsymbol{e}$, similar to [48]. $\boldsymbol{p}_{t+1}$ is the tree positional embedding of the *next* node, which is decided given the DFS order and the symbol generated for the last node (see Section 4.2.4 for details). $\boldsymbol{y}_{t+1}$ is the probability distribution over all symbols to be generated at time step $t+1$. To generated the next node symbol, a simple strategy is greedy search, i.e., the decoder selects the next symbol by choosing the one with the highest probability:

$$u_{t+1} = \operatorname*{argmax}_i \boldsymbol{y}_{t+1} \,.$$

Maintaining the Tree Structure Using a Stack

As mentioned in Section 4.2.4, computing the positions of node u_t requires knowing the node's parent v_t, which then requires the decoder to keep track of the structure of the partial tree generated so far. To do so, we employ a stack S to keep track of node positions in the DFS order. Each element (implemented as a `struct`) in the stack records three items: the node symbol u_t, its position q_t, and its current number of children a_t (recall Section 4.2.3 for definitions of these variables). This way, the decoder knows that the next node u_{t+1} will be attached to the node v_{t+1} at the top of the stack as its next child. The next node's position can then be computed from q_t and a_t (See Eq. 4.3).

Updating the Stack

Generally speaking, when a new node is generated, we i) increment the number of children of this node's parent and ii) push the element containing this node, its position, and its number of children (which is 0 when the node is just generated) to the stack. When the generation of all children of a node finishes, we pop the element containing this node from the stack. Because the decoder generates nodes on the fly and does not have access to the entire tree structure, we need to know when to finish generating children for a parent node. To do so, we introduce an additional special node "<end>", which is attached as the last child of every node. Therefore, whenever the <end> node is generated, we finish generating the children of a parent node v and pop the top element of the stack that contains v. The next generation step will use whichever element at the top of the stack now to determine the position of the next node. In addition, to initialize the generation process, we introduce another special node "<start>" as the parent of the root of every formula tree. <start> and <end> nodes modify the encoder input tree X, resulting in X_{out}, which is the target for the decoder. Figure 4.3a illustrates the modified decoder target formula tree with these additional special nodes and compares it to the encoder input formula tree. The termination condition of tree generation is when the stack is empty, i.e., when there are no more node symbols for which we need to generate children. The stack update process is illustrated in Figure 4.3b.

4.2.5 Tree Beam Search for Tree Generation

The generation process detailed above is a *greedy* process that generates each node optimally but not necessarily the optimal tree. To generate higher-quality formula trees, we develop a tree beam search (TBS) algorithm that extends the classic beam search algorithm commonly used for sequence data [316]. The intuition is to maintain a set of B candidate *trees* during the generation process where B is the beam size that controls the size of the search space. Because of the expanded search space, TBS can keep trees with potentially different structures at each time step in the beam. Therefore, TBS improves over greedy search which only keeps one tree that may become suboptimal after more nodes are generated. When

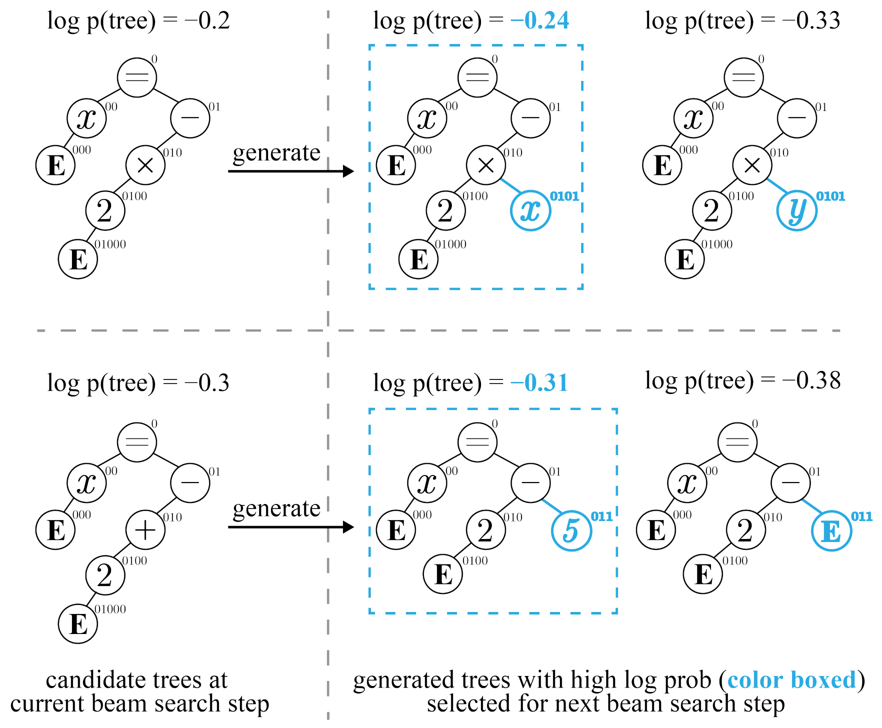


Figure 4.4 : Illustration of the tree beam search algorithm (TBS) at a particular time step with a beam size of 2. TBS enables search over different formula tree structures.

we implement TBS in practice, we need to keep B different stacks for each tree in the beam. During the generation process, for each beam, the decoder first decides the position of the next node and then generates the top B most probable symbols for the node, using the current generated tree (symbols, positions) as input. This process results in a total of B^2 candidate trees to select from (B beams and B possible next nodes for each beam), from which we select B most probable candidate trees to keep in the beam. This process continues until B trees finish their generation process (e.g., when their stacks are empty) or until a preset maximum number of steps is reached. The entire TBS algorithm is illustrated in Figure 4.4 and Algorithm 1.

4.2.6 Relation to Prior Work

We now conduct a review of prior works on learning representations for tree/graph-structured data and provide a detailed comparison of the technical components in FORTE with prior works.

Tree-Structured Data Analysis

Many data types, including computer programs [48, 301], scientific formulae [393], molecule structures [171], neural network architectures [401], and the syntax of natural language sentences [237], have inherent structure such as trees and directed acyclic graphs (DAGs). To better leverage these structures in downstream tasks, an important line of work focuses on learning representations for these structured data types.

Our work differs from prior works that deal with tree/graph-structured data in three ways. First, some prior works only consider tree encoding[319, 296, 356, 237, 314], whereas we consider both tree encoding and generation. Second, our encoder design enables more efficient input processing. Compared to [319, 48], which perform tree traversal *during* training and thus can only process a single data point per iteration, our encoder performs traversal *before* training, which enables mini-batch processing during training. As a result, our approach removes this computationally intensive traversal step from the training process and significantly speeds up training. Compared to [301], which uses a onehot-style tree positional embedding, our encoder employs a different binary tree positional embedding which reduces the space complexity from $\mathcal{O}(LC)$ to $\mathcal{O}(L\log_2(C))$, resulting in faster computation and less memory usage. This reduction is especially significant for trees with a large degree C. Third, our decoder design, i.e., the use of the special <end> node to signal generation termination, enables flexible formula tree generation, which is suitable for real-world scientific formulae. In contrast, many prior works resort to constrained generation processes

Algorithm 1: Tree Beam Search

```
Require : Decoder $p_{\text{dec}}$, maximum generation step $T$, beam size $B$
Input   : tree embedding $\boldsymbol{e}$
Output  : (node, position) tuples $F = \{(U_b, Q_b)\}_{b=1}^{B}$ where $U_b = \{u_{t,b}\}_{t=1}^{T'}$ and $Q_b = \{q_{t,b}\}_{t=1}^{T'}$
Initialize stacks $S_1, \dots, S_B$; Initialize $U_1, \dots, U_B$, $Q_1, \dots, Q_B$;
$u_0 =$ <start>, $q_1 = 0$;
Generate next $B$ nodes $u_{1,1}, \dots, u_{1,B}$ ;
Add $u_{1,b}$ to $U_b$ and $q_{1,b}$ to $Q_b$ for $b \in [1, B]$;
for $b = 1, \dots, B$ do
    if $u_{1,b}$ is not <end> then
        Initialize struct $A_{1,b}$;
        $A_{1,b}.u$ = $u_{1,b}$, $A_{1,b}.a$ = 0, $A_{1,b}.q$ = $q_{1,b}$;
        Push $A_{1b}$ onto $S_b$;
        Compute $q_{2,b}$ via Eq. 4.3;
    else
        Add $(U_b, Q_b)$ to $F$;
for $t = 1, \dots, T$ do
    Generate next nodes $u_{t+1,b}$ and update stacks $S_b$;
    for $b = 1, \dots, B$ do
        Add $u_{t+1,b}$ to $U_b$ and $q_{t+1,b}$ to $Q_b$;
        if $Q_b$ not empty then
            Update stack $S_b$;
            Compute $q_{t+2,b}$ via Eq. 4.3;
        else
            Add $(U_b, Q_b)$ to $F$;
    if $\text{card}(F) \geq B$ or $S_b = \emptyset \, \forall b$ then
        Break;
Return $F$;
```

by enforcing the generated data to have a rigid structure. For example, [171] and [401] propose methods that extensively leverage the limited data diversity in their applications by pre-specifying possible nodes and edges. Some works also restrict the number of children each node must have [48, 301]. The scientific formulae that we work with are much more complex: they contain numerous distinct nodes and edges and many nodes can also have varying numbers of children. It is thus unclear whether these methods generalize to our problem. Section 4.3.2 performs a quantitative comparison between FORTE and some of these applicable baselines and demonstrates FORTE's superior generation performance.

Beam Search for Tree Generation

Beam search has been extensively applied for natural language generation tasks [96, 160, 83] but it is only applicable for sequential rather than tree-structured data. Our TBS method thus builds on and significantly extends sequential beam search. [413] is the only prior work that involves beam search over trees to our knowledge, which appears to resemble our proposed TBS at first glance. However, our TBS is fundamentally different from the "tree beam search" in [413]. The tree beam search in [413] is used for *retrieval*, i.e., for searching nodes deemed as relevant to a query by some scoring function. In contrast, our TBS is used for *tree generation* and thus needs to address issues such as generation termination conditions and partial tree structure maintenance, none of which is an issue in [413]. Moreover, the tree construction in [413] differs from ours: they treat the *entire dataset* as a tree in which each node corresponds to a data point, whereas we treat *each data point* (formula) as a tree in which each node corresponds to a scientific symbol. Thus, even though [413] may resemble a single TBS step, our full TBS method (Algorithm 1) is still first-of-its-kind to the best of our knowledge.

Other applications Involving Formulae

There are other applications that involve scientific/mathematical content. For example, some works focus on automatically *solving* math problems, i.e., generating a solution to a given math problem that consists of numbers or math expressions [172, 289]. Other works focus on the interplay between formulae and natural language. Some examples include learning topic words for a formula [387] and automatic summary generation for a math question post [390]. Our work does not consider these applications but rather focuses on the fundamental research question of how to represent scientific formula. Nevertheless, our work can potentially be integrated into some of these applications; we leave these extensions for future work.

4.3 Experiments

We conduct two experiments to validate FORTE's effectiveness. In the first experiment, we demonstrate the advantage of FORTE over other tree and sequence generation methods in a formula reconstruction task. In the second experiment, we demonstrate the advantage of FORTE over existing methods in a similar formula retrieval task. In all experiments, we implement FORTE using a 2-layer bidirectional GRU as the encoder and a 2-layer unidirectional GRU for the decoder.

4.3.1 Dataset

We collected a large real-world dataset of more than 770k scientific formulae from existing sources including scientific articles on Wikipedia [1] and papers on arXiv.[2,3]

Dataset Preprocessing The formulae in the raw source are in either MathML or LaTeX format. We employ a parser [66] to convert them to the same OT format. We retain formulae that do not incur any errors during the parsing process and whose depth is below 20, degree is below 10, and the number of nodes is below 250. We set these thresholds to remove rare formulae with extremely complicated structures that significantly slow down the training and evaluation process. For some baselines that operate on LaTeX format of the formulae (see Section 4.3.2), we use the tokenizer in [69] to process each LaTeX formula into a sequence of symbols.

[1]https://www.cs.rit.edu/~rlaz/NTCIR-12_MathIR_Wikipedia_Corpus.zip

[2]https://sigmathling.kwarc.info/resources/arxmliv-dataset-082019/

[3]https://nlp.stanford.edu/projects/myasu/topiceq/context_eq_data_20190220.zip

Table 4.2 : Formula reconstruction results. FORTE outperforms all other methods.

Methods	**ACC-1** $\uparrow$	**ACC-5** $\uparrow$	**TED-struct** $\downarrow$	**TED-full** $\downarrow$
GVAE [171]	30.10%	-	-	-
DVAE [401]	50.29%	-	1.178	1.791
tree2treeRNN [48]	71.73%	-	0.507	0.709
tree2treeTF [301]	77.20%	-	0.476	0.507
seq2seqRNN	92.60%	95.56%	0.084	0.176
FORTE (binary, greedy)	94.51%	-	0.053	0.125
FORTE (binary, beam)	**94.67%**	**97.38%**	**0.048**	**0.116**
FORTE (onehot, greedy)	94.28%	-	0.058	0.130
FORTE (onehot, beam)	94.42%	97.22%	0.054	0.124

Scientific Symbol Vocabulary. Recall from Section 4.2.1 that the scientific symbol vocabulary V contains all unique scientific symbols in all the formula OTs in our dataset. The size of this vocabulary V may be unbounded (e.g., every element in the real number set $\mathbb{R}$, which is uncountably infinite, could appear in V as a separate symbol); however, most symbols rarely appear. We thus propose the following truncation method to work with a finite vocabulary in practice. First, we partition the vocabulary V into five disjoint sub-vocabularies according to five symbol *types*, including numeric V_{num} (numbers, decimals), functional V_{fun} (multiplication, subtraction etc.), variable V_{var}, textual V_{txt} and others V_{o}. We do so because different types of math symbols carry different semantic meanings. Then, we retain only the most frequent K symbols in each sub-vocabulary and convert others to an "unknown" symbol *specific to each type*. This setup guarantees that the semantics of symbols that do not occur frequently are preserved.

4.3.2 Formula Reconstruction

In this experiment, we test FORTE's ability to reconstruct a formula. Because some baselines only work on binary trees [48, 301], for this experiment we select a subset of 170k formulae from our dataset whose operator trees are binary.

Baselines.

We consider the following baselines: **seq2seqRNN** which implements the same encoder and decoder as our framework but processes formulae as sequences of math symbols; **tree2treeRNN** [48] which is an RNN-based method capable of encoding and decoding only binary trees; **treeTransformer** [301] which is a Transformer-based method that shows success only on binary trees; **GVAE** [171] and **DVAE** [401], both of which are variational auto-encoders (VAEs) suitable for tree-structured data. More discussion on these baselines are in Section 4.2.6. We also include four variants of our framework to evaluate the utility of i) binary against onehot tree positional encoding and ii) TBS against greedy search for tree generation.

Evaluation Metrics.

We use two groups of metrics. The first group of metrics measure formula reconstruction accuracy (ACC), i.e., the percentage of the decoder outputs that are exactly the same as the ground-truth decoder target formulae. We compute both **ACC-1**, using only the output formula with the highest likelihood, and **ACC-5**, using the five formulae with highest likelihood. Specifically, let $X_{\text{out}}^{(i)}$ be the i-th ground-truth output tree in the test set and $A = \{\widehat{X}_{\text{out}}^{(ij)}\}_{j=1}^{J}$ be the set of J generated trees for $X_{\text{out}}^{(i)}$. Then

$$\text{ACC} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbf{1}_A(X_{\text{out}})\,,$$

where

$$\mathbf{1}_A(X_{\text{out}}) = \begin{cases} 1 & \text{if } X_{\text{out}} \in A \\ 0 & \text{if } X_{\text{out}} \notin A \end{cases}$$

Table 4.3 : Formula reconstruction visualizations comparing FORTE with baselines using two input formulae (top row). Only FORTE succeeds in exactly reconstructing the input.

Methods	$\Theta^{A}_{\text{diff}} = \Theta_{\text{state}}$	$K \wedge \{\neg f \mid f \in F\}$
seq2seqRNN	$\Theta_{\text{diff}} = A^{\text{tharte}}_{V}$	$K \wedge \{\neg\ f\ f \in F\}$
tree2treeRNN	invalid formula	$\wedge \rightarrow \times\times \in$
tree2treeTF	$\Theta^{A}_{\text{diff}} = \Theta$	$(K \in \lambda) \wedge fF \cup (f \in R)$
FORTE	$\Theta^{A}_{\text{diff}} = \Theta_{\text{state}}$	$K \wedge \{\neg f \mid f \in F\}$

and N_{test} is the total number of formula trees in the test set. ACC-1 uses $J = 1$ and greedy search for generation whereas ACC-5 uses $J = 5$ and beam search for generation. For the seq2seqRNN baseline, X_{out} and $\widehat{X}_{\text{out}}$ are both in the format of a sequence of math symbols instead of a tree.

The second group of metrics measures how much the generated formula differs from the ground-truth decoder target formula under the tree format. We use *tree edit distance* (TED) which measures the distance between two trees by computing the minimum number of operations needed, including changing nodes and node connections, to convert one tree to the other. We implement the TED metric using the `apted` package [260] and refer to [254, 217] for an overview of TED. We compute both **TED-full** which considers both node and structural differences and **TED-Struct** which only considers structural differences. For the seq2seqRNN baseline that does not output formula in the OT format, we first use the formula tree parser [66] to convert generated formulae to the corresponding OT and then compute TED. Note that some generated formulae from the seq2seqRNN baseline may incur errors when we convert them to OTs. We do not count these cases in our TED computation, which gives an advantage to the seq2seqRNN baseline; nevertheless, seq2seqRNN still underperforms our method.

Experiment Setup.

We construct training, validation, and test sets by randomly splitting the 170k dataset (recall beginning of Section 4.3.2) 80%-10%-10% for five times. During training, we save the best performing model and parameters using the validation set and then perform formula reconstruction on the test. Whenever beam search is applicable, We use beam size $B = 10$. For the seq2seqRNN baseline and all FORTE variants, we use 500-dimensional hidden states and node embeddings, 2-layer GRUs, 96 batch size, and 50 training epochs. For the two tree2tree baselines [48, 301], GVAE [171], and DVAE [401], we follow the original configurations. All methods are trained on a single NVIDIA RTX 8000 GPU.

Quantitative Results.

Table 4.2 shows the formula reconstruction results, averaged over all five random data splits, comparing FORTE against various baselines. We observe that both GVAE and DVAE do not work well for this task likely because of a mismatch between their model designs and the data type for our task. For example, both GVAE and DVAE leverage rigid rules during generation, i.e., by specifying which node must connect to which node. Because the scientific formulae that we work with have very diverse structures, it is likely that these rules significantly constrain the generation, leading to unsatisfactory reconstruction. This mismatch between baseline designs and our data can also explain the unsatisfactory performance of the two tree2tree baselines. Specifically, these baselines are designed for computer program translation, where the tree structure is also much less varied than those for scientific formulae. The decoder design in these baselines also incorporates multiple constraints, e.g., by specifying the number of children each node must have. Such constraints likely limit the tree2tree baselines' ability to generate high-quality formula trees. In contrast, FORTE almost perfectly reconstructs complex formulae in our dataset, showing excellent

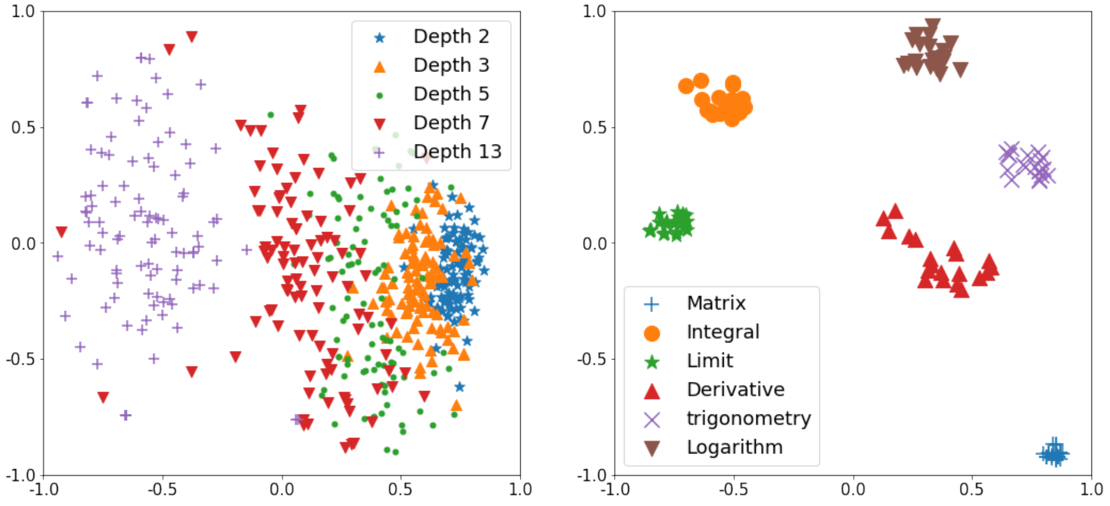


Figure 4.5 : T-SNE visualizations of FORTE formula embeddings for formulae of different tree structures (left) and different content (right). We see clear separation and clustering of different formulae.

robustness and flexibility.

We also see that FORTE outperforms the seq2seqRNN baseline. This observation is not surprising since FORTE exploits the inherent tree structure of scientific formulae whereas seq2seqRNN does not. Moreover, the results for the four FORTE variants clearly demonstrate the benefits of both binary tree positional embedding and TBS, leading to improvements in all four metrics compared to onehot tree positional embedding and greedy search, respectively. We repeat this experiment on the entire 770k formula dataset comparing only FORTE and seq2seqRNN since the tree-based baselines cannot process non-binary trees. FORTE achieves 85.87% compared to seq2seqRNN's 84.30% on TOP-1 ACC and 90.30% compared to seq2seqRNN's 88.52% on TOP-5 ACC, respectively. These results further validate the advantage of representing scientific formulae as OTs over as sequences.

Table 4.4 : Examples of top 5 retrieval results comparing FORTE to TangentCFT. Less ideal retrieved formulae are in red.

	$O(mn\,\log m)$		$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cosh\frac{a}{k}$	
Rank	**FORTE**	**TangentCFT**	**FORTE**	**TangentCFT**
1	$O(mn\,\log m)$	$O(mn\,\log m)$	$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cosh\frac{a}{k}$	$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cosh\frac{a}{k}$
2	$O(n\,\log m)$	$O(n\,\log m)$	$\cos A = -\cos B\cos C + \sin B\sin C\cosh a$	$\cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta$
3	$O(nk\,\log k)$	$O(nm)$	$\cos(A) = -\cos(B)\cos(C) + \sin(B)\sin(C)\cos(a)$	$a = \arccos\left(\frac{\cos\alpha+\cos\beta\cos\gamma}{\sin\beta\sin\gamma}\right)$
4	$O(nk\,\log((n))$	$O(n * m)$	$\cos A = -\cos B\,\cos C + \sin B\,\sin C\,\cos a$	$\cos A = -\cos B\cos C + \sin B\sin C\cosh a$
5	$O(n\,\log h)$	$O(mn)$	$\cos a = \cos b\cos c + \sin b\sin c\cos\alpha$	$\cos C = -\cos A\cos B + \sin A\sin B\cosh c$

Table 4.5 : Examples of top 5 retrieval results comparing FORTE to TangentCFT. Less ideal retrieved formulae are in red.

	$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cosh\frac{a}{k}$	
Rank	**FORTE**	**TangentCFT**
1	$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cosh\frac{a}{k}$	$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cosh\frac{a}{k}$
2	$\cos A = -\cos B\cos C + \sin B\sin C\cosh a$	$\cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta$
3	$\cos(A) = -\cos(B)\cos(C) + \sin(B)\sin(C)\cos(a)$	$a = \arccos\left(\frac{\cos\alpha+\cos\beta\cos\gamma}{\sin\beta\sin\gamma}\right)$
4	$\cos A = -\cos B\,\cos C + \sin B\,\sin C\,\cos a$	$\cos A = -\cos B\cos C + \sin B\sin C\cosh a$

Qualitative Results.

We first visualize in Figure 4.1 some simple formulae generated by passing randomly sampled embedding vectors, i.e., $\boldsymbol{e} \sim \mathcal{N}(0, \boldsymbol{I})$, through FORTE's decoder. Despite their simplicity, these examples show that our tree decoder can generate valid and diverse formulae.

We also visualize some reconstruction results in Table 4.3 on two input formulae, comparing FORTE against seq2seqRNN and the two tree2tree baselines. The ground-truth formulae are at the top of the table and each subsequent row contains the corresponding formulae reconstructed by each method. The two tree-based baselines can correctly generate part of or all symbols in the ground-truth formulae but sometimes in an incorrect order, resulting in formulae that are visually different and even invalid. This observation is likely caused by the absence of a clear termination signal for the generation of children of each parent node during training; since these baselines pre-specified the number of children for each node, they may not be able to properly learn the structural aspects of scientific formulae, which results in generating only partially valid or partially correct formula trees. The seq2seqRNN baseline generates most of the symbols correctly and in the right order but misses or misplaces certain symbols. This observation is likely caused by the loss of tree structural information when we treat a formula as a sequence of symbols, which may lead to incorrect reconstruction. In contrast, thanks to the clear termination signal at each level of the tree and the tree structures being preserved, FORTE perfectly reconstructs both input formulae.

Finally, we visualize FORTE's learned embedding space for formulae in Figure 4.5. We perform two sets of visualizations to examine whether FORTE has learned both the structure and content of formulae. For the first set, we sample formula trees of varying depth and for the second set, we sample formula trees that belong to 6 distinct subjects. We compute their embeddings and plot their 2-dimensional t-SNE embeddings [331]. From Figure 4.5, we can see clear separation and clustering both for formulae with varying tree depths and for formulae with distinct content. Formulae with deeper trees are less clustered (left plot in Figure 4.5) likely because they are also more diverse. Overall, these two sets of visualizations further demonstrate that FORTE learns a meaningful embedding space for scientific formulae.

4.3.3 Formula Retrieval

In this experiment, we evaluate FORTE's capability in a formula retrieval task. Given a query formula (the query), a retrieval method returns the most related formulae (the retrievals) from a collection of candidate formulae.

Query and Retrieval Formulae Processing

We use the first 20 concrete queries (e.g., formula without unknown parts) in the NTCIR-12 Wikipedia math formula retrieval task. We remove one that is too simple (it contains only a single variable and nothing else) and one that incurs an error when being converted to OT. The resulting query set contains 18 queries. Table 4.1 shows a few example queries that we consider. The collection of candidate formulae for retrieval is the NTCIR-12 Wikipedia math formula dataset [393] which is a subset of our 770k formula dataset. Because some formulae in this subset are identical, i.e., formulae that appear in multiple Wikipedia articles, we remove duplicates before performing the retrieval experiment. We also remove formulae that incur errors when being converted to OTs. The resulting candidate formulae collection contains roughly 300k formulae. All of the 18 queries are present in this collection. The processing steps for both queries and candidate retrievals are consistent with that in Section 4.3.1.

Baselines.

We consider three state-of-the-art baselines designed specifically for the formula retrieval task including **Tangent-CFT** [211], **Tangent-S** [66], and **Approach0** [407]. The first baseline is one of the few data-driven formula retrieval methods to date, while the latter two are based on symbolic sub-tree matching and are data-independent.

Evaluation Metrics. We perform a "pooled" human evaluation for the formula retrieval experiment consistent with prior work [393]. First, for each method and each query, we choose the top 25 retrievals and mix them into a single pool for evaluation. Then, for each query and each retrieval, we ask human evaluators how relevant is the retrieval to the query. Possible judgment ratings are relevant, partially relevant, or irrelevant. To encourage fair

Table 4.6 : Formula retrieval results.

	Metrics (partial)		**Metrics (full)**	
Methods	**map**	**bpref**	**map**	**bpref**
Approach0	0.404	**0.537**	0.486	0.507
Tangent-S	0.403	0.449	0.461	0.472
TangentCFT	0.418	0.471	0.462	0.464
FORTE	0.395	0.455	0.475	0.485
FORTE-App	**0.423**	0.484	**0.509**	**0.513**

and consistent evaluation, we first ask the evaluators to browse through all retrieval formulae for a given query before performing the evaluation. This step calibrates the evaluators' judgments. We also provide evaluators with the following evaluation guideline, quoted from [393]: "*A retrieval is considered* relevant *if both its appearance and the content of the formula match that of the query. If either the retrieval's appearance or content matches that of the query but not both, the retrieval is considered* partially relevant. *Otherwise, the retrieval is* irrelevant *to the query*". In total, three human evaluators are involved, each of whom provides us with his/her independent evaluations for each retrieval in the pool for each query.

We use mean average precision (**MAP**) [342] and **bpref** [33] as the evaluation metrics. They are computed by comparing the human evaluation of the pooled retrievals for each query with each method's top 1000 retrievals. Compared to other retrieval evaluation metrics, Both MAP and bpref are easy to interpret and appropriate for evaluating multiple queries and for comparing multiple retrieval methods. We implement these metrics using a common information retrieval evaluation package [32] for both partially relevant and fully relevant retrievals.

Experiment Setup

We use the entire 770k formula dataset to train our FORTE framework and then use the trained encoder to obtain an embedding vector for each formula. For each query, we compute the cosine similarity between its embedding vector and the embedding vector of each formula in the dataset. Finally, we return the formulae with the highest similarity scores as the retrieved ones; See Section 4.2.2 for detailed computation. Because the retrieval results for each baselines is publicly available,[4,5,6] we do not rerun each baseline and simply use the provided retrieval results for our evaluation.

Quantitative Results.

Table 4.6 shows the quantitative evaluation results, averaged over the three evaluators' scores. We observe that FORTE performs well for the fully relevant retrieval evaluation, outperforming both Tangent-S and the data-driven method, TangentCFT. On partially relevant retrieval evaluation, FORTE sometimes falls behind the other baselines. The reason is that, unlike TangentCFT that embeds linearized, sub-components of a formula tree, FORTE focuses on the full tree structure. Therefore, a tree with similar sub-tree components to another tree may differ significantly in their overall structures and get a retrieval score higher from TangentCFT than from FORTE. In addition, unlike Approach0 (and Tangent-S) that directly computes similarity using the symbolic sub-tree components, FORTE (and TangentCFT) computes cosine similarity on the much more abstract formula embeddings, which may cause loss of information compared to explicit symbolic computations used in Approach0. Therefore, following [211], we use a linear combination of the retrieval scores by FORTE

[4] `https://github.com/BehroozMansouri/TangentCFT`

[5] `https://github.com/approach0/search-engine/tree/ecir2020`

[6] `https://www.cs.rit.edu/~dprl/files/release_tangent_S.zip`

Table 4.7 : Zero-shot formula reconstruction results (ACC) on the ARQMath dataset for methods trained on our dataset. FORTE generalizes well to the new dataset.

Methods	**ACC-1** $\uparrow$	**ACC-5** $\uparrow$
tree2treeRNN [48]	46.31%	-
tree2treeTF [301]	72.37%	-
seq2seqRNN	43.46%	52.66%
FORTE (binary, greedy)	89.16%	-
FORTE (binary, beam)	**89.51%**	
FORTE (onehot, greedy)	89.00%	-
FORTE (onehot, beam)	89.43%	94.44%

and Approach0 as a new retrieval method, which we dub **FORTE-App**, that combine the advantage of both methods. This method achieves state-of-the-art retrieval performance on three of the four metrics.

Qualitative Results.

Table 4.4 shows a few qualitative examples comparing formulae retrieved by FORTE to those retrieved by the TangentCFT for the same query. For the first query, all formulae retrieved by FORTE either contain $\log$ or are in the form of $O(\text{variable} \times \text{variable } \log \text{ variable})$, which is the same as that for the query. Similarly, for the second query, all formulae retrieved by FORTE are mostly the same as the query except for a few variables, signs, and functions (e.g., the last $\cos$ function in the 3rd–5th ranked formulae) differences. These examples illustrate FORTE's advantage over baselines in preserving the structure of the query formula and the semantic meaning of symbols in the formula.

Table 4.8 : Zero-shot formula retrieval results (bpref) on the ARQMath dataset. When combining FORTE with Approach0, we achieve the state-of-the-art retrieval performance.

Methods	**Partial**	**Full**	**Harmonic mean**
Approach0	0.477	0.325	0.386
Tangent-S	0.441	0.251	0.320
TangentCFT	0.437	0.305	0.359
FORTE	0.409	0.308	0.351
FORTE-App	**0.502**	**0.328**	**0.398**

4.3.4 Zero-Shot Generalization

We also validate FORTE using a recently released formula dataset, ARQMath,[7] where formulae are collected from Math Stack Exchange,[8] a domain where most formulae are math equations that are very different from those in scientific documents. Specifically, we test the zero-shot generalizability of FORTE (after training on our dataset) to the ARQMath dataset without further fine-tuning.

Table 4.7 reports the formula reconstruction performance with respect to the ACC-1 and ACC-5 metrics and compares FORTE with the baselines. We see that that performances for all methods drop compared to Table 4.2. Nevertheless, FORTE still performs well and significantly better than the baselines. In addition, using binary positional encoding with tree beam search still achieves the best performance among different settings, which is consistent with previous results. Table 4.8 reports the formula retrieval performance with respect to the bpref metric comparing FORTE with the baselines. We see that the performance of data-driven methods, including TangentCFT and FORTE, slightly drops without training or fine-tuning on the new dataset. In contrast, the performance of the best data-agnostic method, Approach0, does not drop. These comparisons suggest that improving FORTE's

[7]`https://www.cs.rit.edu/~dprl/ARQMath/`

[8]`https://math.stackexchange.com/`

generalizability is an important future research direction. When combining FORTE and Approach0, we achieve the state-of-the-art retrieval performance on the ARQMath dataset, which is consistent with the observation in Table 4.6. This result suggests that combining both data-driven and non-data-driven methods is a promising approach for formula retrieval.

4.4 Conclusions and Future Work

In this work, we propose FORTE, a novel, unsupervised scientific formula processing framework by leveraging tree embeddings. By encoding formulae as operator trees, we can explicitly capture the inherent structure and semantics of a formula. We propose an encoder and a decoder capable of embedding and generating formula trees, respectively, and a novel tree beam search algorithm to improve generation quality at test time. We evaluate our framework on the formula reconstruction and the formula retrieval tasks and demonstrate our framework's superior performance in both experiments compared to baselines.

Our work opens doors to many future avenues of research. One direction is to combine our framework's dedicated capability to encode and generate formulae with state-of-the-art NLP methods to enable cross-modality applications that involve both mathematical and natural language. For example, our framework can serve as a drop-in replacement for the formulae processing part in several existing works to potentially improve performance, i.e., in [387] for joint text and math retrieval, in [390] for math headline generation, in [174, 400] for grading students' math homework solutions and providing feedback, and in [289, 172] for neural math reasoning.

Chapter 5

Towards Bloom's Taxonomy Classification Without Labels

5.1 Introduction

Educational assessments, e.g., homework and quiz questions, are important pedagogical instruments that help assess students' knowledge retention and foster higher-order cognitive processing such as thinking and reasoning [318]. To effectively use such questions to improve learning, it is important to know which ones are appropriate for which students and maximize the alignment between the course content and assessments [26]. To this end, teachers often utilize the *Bloom's Taxonomy* of educational objectives [25, 164] as a framework to categorize questions based on the specific cognitive functions that they exercise. This framework provides practical guidelines on how to characterize existing questions such that they facilitate specific cognitive processes and how to author *new* questions. However, teachers often do not assign Bloom's labels when authoring new questions, making it difficult for other teachers to reuse them. The reason is that manually assigning Bloom's level labels is incredibly time-consuming, expensive, and error prone [2, 242].

To reduce the cost of manually labeling questions with Bloom's levels, researchers have developed various automatic labeling methods that almost universally formulate the task as a supervised learning classification problem [242, 383, 131, 229, 230, 243, 287, 263, 315, 72]. The resulting models must learn using *supervised training data*, i.e., a large collection of questions already labeled with Bloom's levels, in order to accurately predict Bloom's levels [242]. Gathering such data involves the expensive and problematic process of manually collecting Bloom's level labels outlined above. Likely due to the high cost of collecting labeled data, previous works have used very small labeled datasets, which raises concerns about their robustness and generalizability [242]. In contrast, it is straightforward to collect questions that *do not* have Bloom's level labels. The abundance of unlabeled data and the high cost of collecting labeled data calls for methods *other than supervised ones* for Bloom's level classification.

5.1.1 Contributions

In this work, we develop a new framework for Bloom's level classification based on *weakly supervised learning* (WSL) that accurately classifies questions into Bloom's levels *without requiring labeled training data*. The fundamental idea behind BLACBOARD (for *B*loom's *L*evel cl*A*ssifi*C*ation *B*ased *O*n we*A*kly supe*R*vise*D* learning) is to codify experts' domain knowledge in Bloom's Taxonomy into a set of *labeling functions* (LFs) and then programmatically generate Bloom's level labels using these functions to form a *weakly* labeled dataset. In this way, we create a labeled dataset using entirely unlabeled data and human experts' domain knowledge.

Our framework consists of three main components. The first component is a novel set of *LFs* carefully crafted from domain experts' knowledge of Bloom's Taxonomy, which generates a set of (noisy) Bloom's level labels for each question. The second component is a *probabilistic graphical model* that infers the most appropriate (weak) Bloom's level label for each question from the set of (noisy) labels. The third and last component is a *supervised classifier* that we train on the inferred weakly labeled dataset and use for the final Bloom's level assignment. We experimentally evaluate our framework on a large, real-world question bank spanning a variety of subjects such as calculus, physics, sociology, and history. Preliminary results on a binary Bloom's level classification task demonstrate that

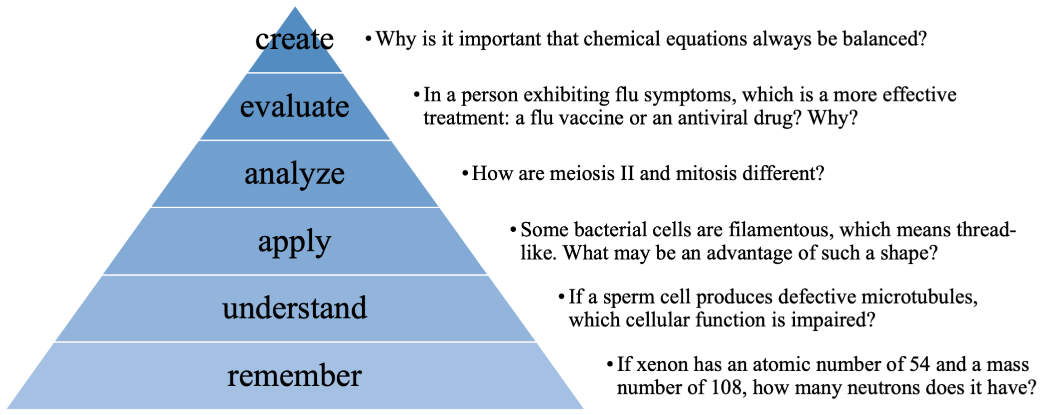

create
• Why is it important that chemical equations always be balanced?
evaluate
• In a person exhibiting flu symptoms, which is a more effective treatment: a flu vaccine or an antiviral drug? Why?
analyze
• How are meiosis II and mitosis different?
apply
• Some bacterial cells are filamentous, which means thread-like. What may be an advantage of such a shape?
understand
• If a sperm cell produces defective microtubules, which cellular function is impaired?
remember
• If xenon has an atomic number of 54 and a mass number of 108, how many neutrons does it have?

Figure 5.1 : Illustration of the Bloom's Taxonomy levels from level 1 (bottom) to level 6 (top), with an example question corresponding to each level from a biology textbook.

our proposed WSL framework achieves competitive classification performances compared to fully supervised learning methods. Notably, our framework obtains such results *without any a priori known labels*.

5.2 Preliminaries and Related Work

Bloom's levels and classification.

The Bloom's Taxonomy [164] that we use in this work consists of 6 levels, each aiming to evaluate cognitive processes that increase in difficulty. Figure 5.1 illustrates each Bloom's level. The 6 Bloom's levels have in several instances been re-categorized into *two levels* to reflect lower-order cognitive skills (LOCS) and higher-order cognitive skills (HOCS) [318, 25, 415, 148, 62], where higher-order cognitive skills refers to cognitive processes that require more than merely retrieving information [14, 114]. Aligned with this perspective, in this work we consider this LOCS and HOCS binary Bloom's level categorization, where LOCS contains Bloom's level 1 and HOCS contains Blooms level 2 – 6.

Existing research has developed methods for Bloom's level classification based on supervised learning. Most of the prior work is based on support vector machines (SVMs) [383, 131, 287, 263] with a few others using naïve Bayes [315] and ensemble methods [229, 242]. Other work explores text representation methods such as augmenting the TF-IDF representation [230] or integrating linguistic features [72, 243]. However, as mentioned earlier, all of the above rely on fully labeled datasets, which are difficult and expensive to obtain in practice. Indeed, most of the above works use very small datasets of only a few hundred or fewer questions, which severely limits their practical and research impacts.

Weakly supervised learning.

Weakly supervised learning (WSL) [274] is an emerging machine learning paradigm that enables one to solve classification problems *without using any labeled data.* We refer readers interested in WSL to [274] for a thorough introduction and focus on its notable features here. Compared to traditional supervised learning, WSL requires no a priori known labels; the labels are created during the modeling process. This overcomes supervised learning methods' reliance on labeled data, which is often limited in quantity and difficult to collect. Thus, WSL makes it possible to solve classification problems using only unlabeled data, which in some applications is massive and cheap to collect. WSL also has rigorous theoretical foundations [274, 337, 336, 11] and has had promising empirical success in a wide range of real-world, high-stakes applications. For example, in medical applications, WSL has contributed to medical entity recognition [84], MRI image classification [85], medical device surveillance [35], and genomic information compilation [167]. WSL has not yet been applied in education except for one work that proposes a weak supervision-based conversational agent for teacher education [65].

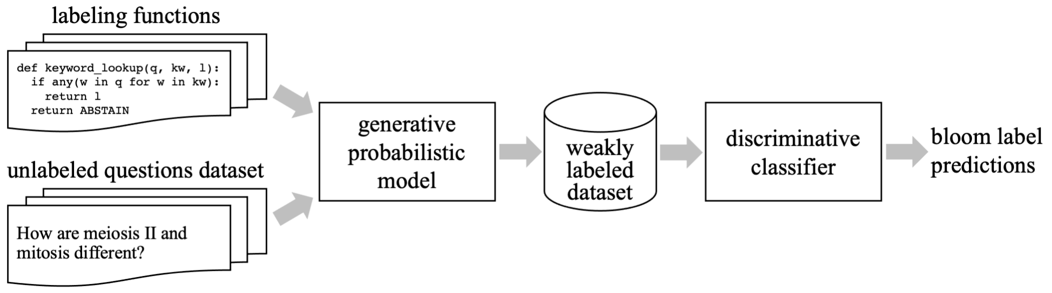

labeling functions
def keyword_lookup(q, kw, l):
  if any(w in q for w in kw):
    return l
  return ABSTAIN
unlabeled questions dataset
How are meiosis II and mitosis different?
generative probabilistic model
weakly labeled dataset
discriminative classifier
bloom label predictions

Figure 5.2 : An illustration of BLACBOARD, our proposed weakly supervised learning framework for Bloom's level classification of questions.

5.3 Methodology

We now describe our BLACBOARD framework. We introduce our novel LFs, explain how to incorporate LFs into a graphical model to infer the weak Bloom's level label for each question, and finally show how to combine the weakly labeled questions dataset with a supervised classifier. As mentioned in Section 5.2, in this work we tackle the *binary* Bloom's level classification problem, where one class LOCS includes Bloom's level 1 and the other class HOCS includes Bloom's level 2–6. Extension to all 6 Bloom's levels is left for future work.

5.3.1 Human Expert-Inspired Labeling Functions (LFs)

An LF $f_j(\boldsymbol{x}_i) : \mathbb{R}^D \rightarrow |\boldsymbol{y}| \cup \{\emptyset\}$ assigns a Bloom's level label to each question. $\boldsymbol{x}_i \in \mathbb{R}^D$ is the D-th dimensional vector representation of question i. $\boldsymbol{y} = \{y_i\}_{i=1}^N$ is the set of labels and $y_i \in \{0, 1\}$ is the label for each of the N question. We assume LOCS and HOCS are class 0 and 1, respectively. $j \in \{1, \dots, L\}$ indexes the LFs. $|\cdot|$ is the cardinality of $\boldsymbol{y}$, which in our case is 2 because $\boldsymbol{y}$ is binary. We include the empty set as a potential output of an LF, because an LF can abstain, i.e., give no label to a question.

To effectively design these critical LFs, we conducted semi-structured interviews of

Table 5.1 : Description of labeling functions (LFs). “Bloom 1 – 6 kw” collapses 6 similar LFs.

LF	**Description**	**Output**
short	Check the number of words in the question	LOCS if # characters less than 75, $\emptyset$ otherwise
why	Check whether the word “why” is in the question	HOCS if the word “why” is in question, $\emptyset$ otherwise
Bloom 1–6 kw	Each checks whether keywords (kw) specific to each Bloom’s level is in the question	LOCS if the keyword is in question, $\emptyset$ otherwise
glossary	Check whether glossary terms is in the question	HOCS if the more than 3 terms are in question, $\emptyset$ otherwise
readability	Computes a question’s Flesch readability score	HOCS if the score < 50, $\emptyset$ otherwise

3 education experts investigating how they utilize Bloom’s taxonomy in their general pedagogy while creating tests, authoring questions, and labeling existing questions. We used the findings from these interviews to design 11 simple labeling functions that represent Bloom’s level characteristics and domain knowledge, which we believe to be useful for determining the appropriate Bloom’s level for a given question. Table 5.1 describes all of our LFs, which we categorize into 3 groups. The first group (“short”, “why”, “readability”) focuses on question properties. Our intuition is that HOCS questions tend to be longer, less readable, and ask more “why” questions. The second group (“Bloom 1 – 6 kw”) looks for keywords (kw) indicative of each Bloom’s level. We collect these keywords from teachers’ rubrics and instructions for writing questions at a specific Bloom’s level. The third group (“glossary”) looks for keywords specific to *subject domains*, i.e., biology. Our intuition is that HOCS questions tend to contain more subject domain keywords, which potentially reflects increased question complexity and demands higher level skills. We collect these keywords from the textbooks’ glossaries.

5.3.2 Graphical Model for Weak Label Inference

With the LFs, each data point now has a set of noisy labels. However, to train a classifier, each data point must have a label. Therefore, we must learn the most likely (weak) label given a set of labels for each data point. To do so, we leverage a generative model following [274], which we include here for completeness. Concretely, let $\boldsymbol{y} \in \mathbb{R}^N$ be the ground-truth labels and $\Upsilon \in (\boldsymbol{y} \cup \{\emptyset\})^{N \times M}$ be the weak label matrix obtained from the LFs where N is the number of data points and L is the number of LFs. Then, we model the joint distribution of the weak and the true labels for all data points using the following generative model:

$$p_\theta(\Upsilon, \boldsymbol{y}) = A_\theta^{-1} \exp\left(\sum_{i=1}^{M} \theta^\top f_i(\Upsilon, y_i)\right) ,$$

where A_θ is a normalizing constant. $f_i(\cdot)$ is a function that combines three LF properties including labeling propensity f^{pro}, accuracy f^{acc} and correlation f^{cor}:

$$f_{i,j}^{\text{pro}}(\Upsilon, \boldsymbol{y}) = \mathbf{1}\{\Upsilon_{i,j} \neq \emptyset\} , \tag{5.1}$$

$$f_{i,j}^{\text{acc}}(\Upsilon, \boldsymbol{y}) = \mathbf{1}\{\Upsilon_{i,j} = y_i\} , \tag{5.2}$$

$$f_{i,j,k}^{\text{cor}}(\Upsilon, \boldsymbol{y}) = \mathbf{1}\{\Upsilon_{i,j} = \Upsilon_{j,k}\} , \tag{5.3}$$

where θ is the model parameters and $\mathbf{1}\{\cdot\}$ is an indicator function. These three properties are important for understanding LFs' effectiveness and thus are incorporated into the generative model. Notably, computing LF properties using Eqs. 5.1 and 5.3 do not require any ground-truth data. We will leverage these unsupervised LF analytics in Section 5.4.3 and validate LF effectiveness. Because we assume ground-truth labels $\boldsymbol{y}$ are unavailable in our setting, we optimize the model and learn the model parameters $\widehat{\theta}$ using the marginal log likelihood which eliminates $\boldsymbol{y}$. Concretely, the optimization objective is

$$\widehat{\theta} = \arg\min_{\theta} \; -\log \sum_{\boldsymbol{y}} p_{\theta}(\Upsilon, \boldsymbol{y}).$$

More details on labeling correlation computation and model optimization are available in [274].

5.3.3 Bloom's Level Classifier

To obtain the final Bloom classification results, we leverage a classifier trained on the weakly labeled dataset in which each data point has a label inferred by the generative model using our LFs. Note that we may simply use the inferred labels as the final Bloom taxonomy label for each question without training a supervised classifier. However, a classifier brings more modeling capability and is beneficial for classification performance. In Section 5.4.2, we empirically confirm the advantage of additionally training a classifier on the weak labels.

5.4 Experiments

We now empirically show the power of BLACBOARD that uses questions without any Bloom's level labels for Bloom's level classification.[1] We first compare BLACBOARD with the selected LFs to supervised learning methods that use fully labeled data. We then present LF analytics that our framework enables and that help us understand the effectiveness of each LF. Notably, this LF evaluation step is unsupervised, i.e., without access to the ground-truth labels.

[1]A demonstration and associated code of BLACBOARD are available at `https://github.com/manningkyle304/edu-research-demo`

5.4.1 Dataset

We use a new, closed-source, large-scale, real-world dataset from OpenStax[2] of 17,719 multiple-choice questions that are actively used in practice and have expert-tagged Bloom's levels ranging from level 1 to 6. The dataset includes questions from multiple subjects, including natural sciences (biology, physics) and social sciences (economics, history, sociology), and is representative of high-school and college-level courses. These properties make our dataset the largest and most diverse one to be used for Bloom's level classification to our knowledge. For our binary Bloom's level classification problem, we reassign the Bloom labels, resulting in 11,190 LOCS questions and 6,529 HOCS questions. A naïve majority classifier gives a classification accuracy of 63.15%, which serves as one of our baselines. We encode each question into a numeric feature vector using TF-IDF that is commonly used in text mining and information retrieval [285].[3]

5.4.2 Comparing BLACBOARD to Fully Supervised Methods

In this experiment, we compare BLACBOARD, which uses entirely unlabeled data, against supervised learning methods that use fully labeled data. This experiment will demonstrate the capability of WSL in effectively performing classification tasks in the absence of human-provided ground-truth labels. We randomly split the data into 80% training and 20% test sets. For BLACBOARD, we learn the weak labels for all questions in the training set and then train a classifier on the weakly labeled training set. For supervised learning, we simply train a classifier on the training set with ground-truth labels. To verify the results in different settings, we use a variety of classifiers for both BLACBOARD and supervised learning,

[2] `https://openstax.org`

[3] We also experimented with other featurization methods, but the results were similar to TF-IDF. We thus use TF-IDF for all experiments in this work.

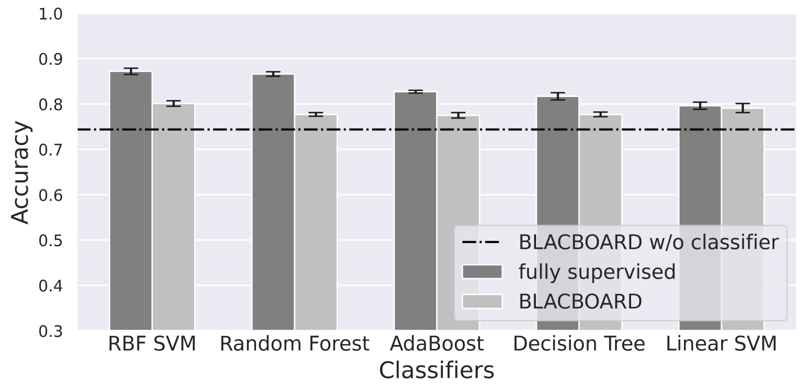


Figure 5.3 : Quantitative Bloom's level classification results comparing BLACBOARD to fully supervised methods. We see that for all classifiers used, BLACBOARD achieves classification accuracies very close to supervised methods using *fully unlabeled* data.

including linear support vector machine (SVM), radial Basis function (RBF) SVM, random forest, adaboost, and decision trees [106, 21]. We perform each experiment 5 times and report the average accuracies on the test set.

Figure 5.3 visualizes the average test accuracies (with standard deviation) comparing BLACBOARD to fully supervised learning methods for each classifier. We observe that BLACBOARD achieves classification performance close to fully supervised methods. In particular, for linear SVM, BLACBOARD achieves statistically the same performance as its fully supervised counterpart. This result showcases that, with entirely unlabeled data and by creatively incorporating domain expertise, BLACBOARD approaches the performance of supervised learning methods using fully labeled data with minimal performance degradation. We also see that removing the classifier in BLACBOARD leads to lower accuracy, implying that a classifier trained on weak labels can improve performance.

Table 5.2 : Labeling function (LF) analysis. We can identify 2 weak LFs, because of low coverage (“why” LF) and low overlap (“Bloom 5 kw” LF).

Labeling Functions	**Coverage**	**Overlap**	**Conflict**
short	0.435	0.326	0.111
why	0.007	0.006	0.003
Bloom 1 kw	0.321	0.308	0.133
Bloom 2 kw	0.218	0.211	0.163
Bloom 3 kw	0.286	0.250	0.185
Bloom 4 kw	0.166	0.163	0.101
Bloom 5 kw	0.142	0.006	0.003
Bloom 6 kw	0.142	0.142	0.091
glossary	0.104	0.085	0.053
readability	0.329	0.271	0.197

5.4.3 Unsupervised Labeling Function Analysis

In this experiment, we show how we can perform LF analysis in an unsupervised manner, i.e., without the ground-truth labels. This analysis reveals insights about the effectiveness of each LF and helps us retain or discard certain LFs.

Metrics.

Recall that the generative model in BLACBOARD leverages three LF properties including propensity, accuracy, and correlation. We now leverage these properties to define 3 types of statistics for each LF. The first statistic is **coverage**, which computes the number of questions that an LF assigns a label, i.e., $\Upsilon_{ij} \neq \emptyset$ for LF f_j. The second statistic is *overlap*, which computes the portion of questions with at least 2 weak labels. The third statistic is *conflict*, which computes a portion of questions for which at least 1 other LF yields a label different from the LF under examination.

Results, interpretation and significance.

Table 5.2 reports the analysis results, averaged over 5 random runs. We first see that the conflict scores are low for all LFs. This is an encouraging signal because each LF agrees with the other LFs in general. The opposite situation, in which LFs tend to give contrasting labels, would cause much trouble for the graphical model to infer the most likely weak label. Therefore, having a sizable number of LFs that do not conflict much suggests that all LFs are reasonable. However, some LFs have low coverage (e.g., "why" LF) and low overlap (e.g., "why" and "Bloom 5 kw" LFs). An LF with low coverage and overlap suggests that this LF influences only a very small fraction of all data points and contributes little to improving the modeling capacity. For example, for the "why" LF (indexed by j), its corresponding row in the weak label matrix Υ_j would have mostly -1 representing abstain (e.g., no label). This row thus has limited influence for the graphical model inference.

In this work, we choose to keep all LFs because conflicts are low. Even for the LFs with low coverage and overlap, they have lower conflicts and thus do not appear to cause negative effects on the weak label inference process. Through these analyses, we show that by examining the LF statistics in an unsupervised manner (e.g., without using any ground-truth labels), we can verify the validity of our LFs and identify LFs that have limited contribution to the weak label inference process.

Post-hoc analysis with ground-truth labels.

To illustrate the conclusions from our unsupervised LF analysis using coverage, overlap, and conflict statistics, we additionally perform a post-hoc, leave-one-LF-out experiment. Specifically, we remove one LF and use the remaining LFs to train the generative model in BLACBOARD. We perform this training step for every LF using the training data (again without ground-truth labels) and report the labeling accuracy on the test set. This

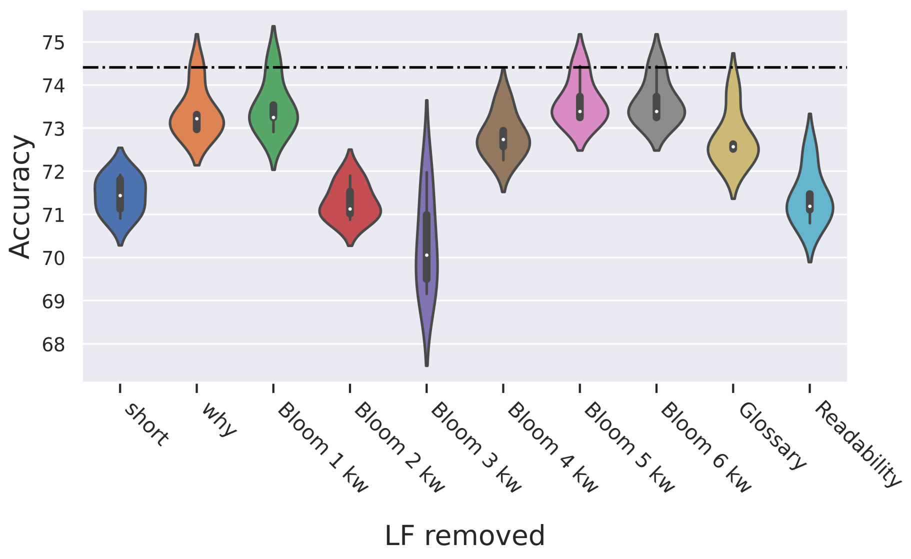


Figure 5.4 : Post-hoc leave-one-LF-out analysis results. We show the average and density of the accuracies for the experiment with each excluded LF. The horizontal line is the best accuracy with all LFs. We can see that none of the LFs, if removed, statistically improve the labeling accuracy, suggesting that each individual LF contributes to the labeling inferring capability and all should be kept.

accuracy is computed using BLACBOARD's inferred labels and the ground-truth labels and is thus a post-hoc analysis because we assume ground-truth labels are not available in practice. Nevertheless, this post-hoc analysis reveals the effectiveness of each individual LF. Specifically, if the test accuracy improves *without* a particular LF, then this LF does not contribute to improving BLACBOARD's label inferring capability and thus should be removed. If the opposite situation happens, then this LF is useful and should be retained.

Figure 5.4 shows the post-hoc analysis results for each LF, averaged over 5 runs with different train-test splits. The horizontal line is the best labeling accuracy with all LFs. We can see that most of the LFs, if removed, significantly decrease the accuracy, suggesting that these are important LFs and should be kept. Some LFs, such as the "knowledge keywords" (Bloom 1 kw), sometimes improve performance if removed. However, the improvement is not statistically significant; in most of the 5 runs, the accuracy decreases when these LFs are removed. Thus, all LFs are useful in BLACBOARD and none should be removed. This result is consistent with and confirms the conclusion in the preceding unsupervised LF analysis.

5.5 Conclusions and Future Work

In this chapter, we have introduced BLACBOARD, a WSL framework for Bloom's level classification. To the best of our knowledge, this is the first work to investigate WSL for Bloom's level classification. Our framework, unlike existing supervised methods for Bloom's level classification, requires no labeled dataset. Instead, it incorporates instructional and domain expertise into modeling to create weak labels for classification. We report promising preliminary results on a large, real-world question dataset, demonstrating that, compared to conventional fully supervised methods, BLACBOARD suffers little to no decline in performance.

The modular framework of our weak supervision approach coupled with our new procedures to perform unsupervised model diagnostics enables iterative and intuitive adjustments for improvements. In the future, we intend to extend our framework from binary to full 6-level Bloom's level labeling. One promising avenue of research is to investigate more sophisticated LFs based on linguistic and heuristic characteristics of Bloom's levels. More advanced models that leverage recently developed deep probabilistic methods can also contribute to improving the weak label inference capability. The essential value of a WSL approach is that it does not rely on massive, high-quality, labeled data, therefore resulting in a scalable solution to a previously unscalable problem. Our present work forecasts the exciting promise of transferring the WSL approach for solving a wide range of problems in Artificial Intelligence in Education beyond Bloom's level classification including question generation [358], educational conversational agents [65], forum posts sentiment analysis [271], and knowledge graph construction [267].

Chapter 6

Educational Question Mining At Scale: Prediction, Analysis, and Personalization

6.1 Introduction

Online education platforms are transforming education by democratizing access to high-quality educational resources and personalizing learning experiences. A central instrument in today's online education scenarios is assessment questions (referred to as "questions" henceforth), which help teachers evaluate the students' abilities and help students reinforce the knowledge they are learning [149, 150, 58, 163, 159]. Questions are particularly important in *online* education, because teachers have more limited interactions with students; students answer records to questions serve as one of the few ways for teachers to interact with and understand their students [327]. With the world reeling from the impact of the Covid-19 pandemic, there has been a rapid and massive increase in the number of students learning online [18, 286, 339]. Questions as assessment and learning instruments have thus become even more prominent.

A key challenge to best utilize these questions is how to choose the most suitable ones for students. Only high quality and suitably difficult questions are beneficial for learning [22]; as a result, understanding these two properties of questions help guide the choice of questions. While in traditional classroom settings, manually examining and selecting questions to attend to the learning status of each individual student remains the best practice, this labor- and time-intensive procedure clearly does not scale to large-scale online education scenarios in which the number of questions and students can be massive. We thus desire an efficient tool for question analysis at scale and capable of computing question analytics including quality and difficulty and automatically selecting questions for students. The analytics will serve as side information that helps teachers and students select appropriate questions for their pedagogical and educational needs. The automatic question selection process enables personalized learning and adaptive testing experience when the number of students greatly exceeds teachers' capacities [39, 384]. Both analytics and personalization are important to realize the values of massive questions in online education scenarios.

To address the above challenges, we aim to develop an AI solution for large-scale online educational question mining, providing both insights for question quality and difficulty and strategies for choosing questions for each student. This task involves 3 challenges. First, the number of questions, students, and answer records is extremely large and processing such data may be computationally expensive. Second, online educational data is highly incomplete because each student can only answer a small fraction of all available questions. Third, we need to design quantitative metrics and strategies to accurately identify question quality and to adaptively choose questions for each student. Overall, we need a solution that is efficient, handles highly sparse data, and automatically acquires educationally meaningful and actionable insights about questions.

6.1.1 Contributions

In this work, we collect a large real-world online educational dataset in the form of students' answers to multiple-choice questions and develop a machine learning framework to quantify question quality and difficulty and provide personalized question selection strategy. We briefly summarize our framework below.

- We leverage the partial variational auto-encoder (p-VAE) [206, 205] to efficiently handle the highly sparse educational data at a large scale. p-VAE models existing students' answers and predicts the potential answer to unseen questions in a probabilistic manner.

- We propose a novel information-theoretic metric to quantify the quality of each question based on p-VAE. We also define a metric to quantify question difficulty.
- We propose a novel information-theoretic strategy to sequentially select questions for each student. Our strategy chooses a sequence of questions which best identifies the student's learning status.
- We experimentally validate our framework on a new, large educational dataset and demonstrate state-of-the-art performances of our framework over various baselines.

6.2 Dataset

We analyze data from Eedi,[1] a renowned real-world online education platform used by over 100,000 thousand students and 25,000 teachers in over 16,000 schools. Eedi offers crowd-sourced, multiple-choice diagnostic questions to students from primary to high school (roughly between 7 and 18 years old). Each question has 4 answer choices and only one of them is correct. Currently, the platform focuses mainly on math questions. Figure 6.1 shows an example question from the platform. We use data collected from the 2018 – 2019 school year.

We organize the data in a matrix form where each row represents a student and each column represents a question. Each entry contains a number that represents whether the student has answered a question correctly (i.e., 0 represents the wrong answer and

[1] `https://eedi.com`

1 represents the correct answer). Each student has only answered a tiny fraction of all questions and hence the matrix is extremely sparse. We thus removed questions that contain less than 50 answers and students who have answered less than 50 questions. Besides, when a student has multiple answer records to the same question, we keep the latest answer record.

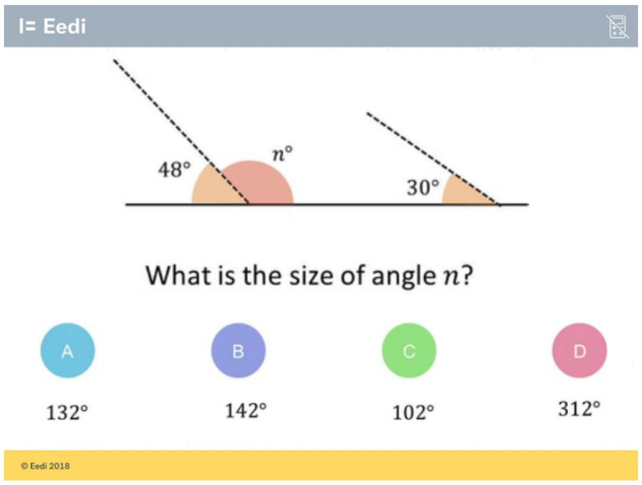
The above preprocessing steps lead to a final data matrix that consists of more than 17 million students' answer records with 123,889 students (rows) and 27,613 questions (columns), making it one of the largest educational datasets to date compared to a number of existing ones [109, 310]. Additionally, each question is linked to one or more topics that describe the skills [335, 56] that the question intends to assess. We have open-sourced this dataset [359] and it is available at `https://eedi.com/projects/neurips-education-challenge`.

6.3 Method

The extreme sparsity and massive quantity of our dataset brings challenges to analyze both the question quality because each student only answers a small fraction of potentially non-overlapping questions. To gain insights into such real-world educational data, we first need a model that predicts the missing data with uncertainty estimation. The missing data in our case is students' answers to unseen questions. With such a model, we can design different metrics to quantify question quality and difficulty.

The first step is formulated as the following probabilistic missing data imputation (matrix completion) problem. We have a data matrix $\boldsymbol{X}$ of size N by M, where N is the total number of students and M is the total number of questions. Each entry x_{ij} is binary which indicates whether student i has answered question j correctly.[2] The data matrix is only partially observed; we denote the observed part of the data matrix as $\boldsymbol{X}_O$. Then, we would then like to accurately predict the missing entries in a *probabilistic* manner which enables the design

[2]x_{ij} can also be categorical which is the answer in a multiple choice question that a student selects.



What is the size of angle n?

Figure 6.1 : An example question from our new dataset.

of various metrics for question quality, difficulty, and selection strategy. Thus, we use the partial variational auto-encoder (p-VAE) [206], which is the state-of-the-art method for the above imputation task.

The second step is to quantify question quality and difficulty and adaptively select questions for students. Specifically, we define a difficulty measure using the full data matrix completed by p-VAE. We quantify question quality by measuring the value of information that each question carries using an information theoretical metric. Using Similar ideas, we sequentially select questions for each student based on a notion of information gain. Such "information" computation is made possible and efficient by our utilization of the p-VAE in the first step above. We present these two steps in detail in the remainder of this section.

6.3.1 Partial Variational Auto-encoder (p-VAE) for Student Answer Prediction

p-VAE [205] is a deep latent variable model that extends traditional VAEs [157, 277, 396] to handle missing data as in such education applications. VAEs assume that the responses

$\boldsymbol{x}_i$ of student i is generated from a latent variable $\mathbf{z}_i$:

$$\begin{aligned} p_\theta(\boldsymbol{X}) = \prod_{i=1}^{N} p_\theta(\boldsymbol{x}_i) &= \prod_{i=1}^{N} \int p_\theta(\boldsymbol{x}_i|\mathbf{z}_i)\, p_\theta(\mathbf{z}_i)\, d\mathbf{z}_i \\ &= \prod_{i=1}^{N} \int \prod_{j=1}^{M} p_\theta(x_{ij}|\mathbf{z}_i)\, p_\theta(\mathbf{z}_i)\, d\mathbf{z}_i \,, \end{aligned}$$

where x_{ij} is the i-th student's answer to the j-th question. We use a deep neural network for the generative model $p_\theta(x_{ij}|\mathbf{z}_i)$ because of its expressive power. Of course, $\boldsymbol{x}_i$ contains missing entries because each student i only answers a small fraction of all questions. Unfortunately, VAEs can only model fully observed data. To model partially observed data, p-VAE extends traditional VAEs by exploiting the fact that, given $\boldsymbol{z}_i$, $\boldsymbol{x}_i$ is fully factorized. Thus, the unobserved data entries can be predicted given the inferred $\mathbf{z}_i$'s. Concretely, p-VAE optimizes the following *partial* evidence lower bound (ELBO):

$$\begin{aligned} &\log p(\boldsymbol{X}_O) \geq \log p(\boldsymbol{X}_O) - D_{\mathrm{KL}}(q(\mathbf{z}|\boldsymbol{X}_O)\|p(\mathbf{z}|\boldsymbol{X}_O)) \\ &= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\boldsymbol{X}_O)}[\log p(\boldsymbol{X}_O|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\boldsymbol{X}_O)] \\ &\equiv \mathcal{L}_{\text{partial}} \,, \end{aligned}$$

which is in the same form as the ELBO for VAE but only over the observed part of the data. Because $\boldsymbol{X}$ is binary, we use Bernoulli distribution as the likelihood function. However, we can also choose to use students' actual answers (A, B, C, or D) as $\boldsymbol{X}$ where the likelihood function becomes categorical. Investigation of categorical data format is left for future work.

The challenge is to approximate the posterior of $\mathbf{z}_i$'s using a partial observed data vector. p-VAE uses a set-based inference network $q_\phi(\mathbf{z}_i|\boldsymbol{x}_{O_i})$, where $\boldsymbol{x}_{O_i}$ is the observed subset of answers for student i [392, 265]. $q_\phi(\mathbf{z}_i|\boldsymbol{x}_{Oi})$ is assumed to be Gaussian; Concretely, the

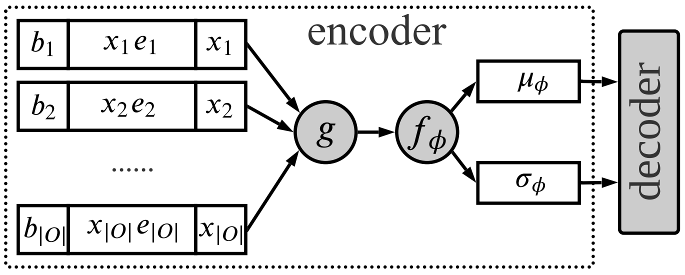


Figure 6.2 : Illustration of the p-VAE model architecture.

mean and variance of the posterior of the latent variable is inferenced as

$$[\mu_\phi(\boldsymbol{x}_O), \sigma_\phi(\boldsymbol{x}_O)] = f_\phi(g(\boldsymbol{s}_1, \dots, \boldsymbol{s}_{ij}, \dots, \boldsymbol{s}_{|O|}))\,, \tag{6.1}$$

where we have dropped the student index i for notation simplicity. $\boldsymbol{s}_{ij}$ is the observed answer value augmented by its location embedding, which is learned; $g(\cdot)$ is a permutation invariant transformation such as summation which outputs a fix sized vector; and $f_\phi : \mathbb{R}^M \rightarrow \mathbb{R}^K$ is a regular feedforward neural network. In this work, we set $\boldsymbol{s}_{ij} = [x_{ij}, x_{ij}\boldsymbol{e}_j, b_j]$ where Figure 6.2 illustrates the network architecture of p-VAE.

Note that, in p-VAE, some parameters have natural interpretations. For example, the per-question parameters $[\mathrm{e}_j, b_j]$ can be collectively interpreted as a *question embedding* for each question j. The per-student latent parameter $\mathbf{z}_i$ can be interpreted as a *student embedding* for each student i. Furthermore, p-VAE quantifies the uncertainty of the student embedding $\mathbf{z}_i$ in the form of an (approximate) posterior distribution. This enables us to define information-theoretic metrics and strategies for question difficulty, question quality, and question selection, which we describe next.

6.3.2 Question Difficulty Quantification

For a group of questions answered by the same group of students, the difficulty level of the questions can be quantified by the incorrect rate of all students' answers. However, for real-world online education data, every question is answered only by a small fraction of students and by different subsets of students with different educational backgrounds. Thus, directly comparing the difficulty levels of the question from observational data is not accurate because an easy question, which may be answered by only less-skilled students, may be shown to be difficult if only observational data is used. Thanks to p-VAE, we can predict whether a student can correctly answer an unseen question. We achieves this by first predicting students' responses to all unanswered questions and then defining the difficulty level of question j as $\sum_i^N \frac{p(x_{ij}=0)}{N} \approx \frac{1}{N}\sum_i^N \hat{x}_{ij}$ where $\hat{x}_{ij}$ denotes if the student has answered the question correctly. Higher value implies that more students are predicted to answer this question correctly and that this is an easier question.

6.3.3 Question Quality Quantification

Working closely with education experts, we found that high-quality questions are considered to be those that best differentiate student abilities. When a question is simple, almost all students will answer it correctly. When a question is badly formulated, all students will provide incorrect answers or random guesses. In any of these cases, the question neither helps the teacher gain insights about the students' abilities nor helps students learn well. Thus, high-quality questions are the ones that can differentiate the students' abilities.

We thus formulate the following information theoretic objective to quantify the quality of question j:

$$R(j) = \mathbb{E}_{x_{ij} \sim p_\theta(x_{ij})} \Big[\mathrm{D}_{\mathrm{KL}}[p_\phi(\mathbf{z}_i|x_{ij}) \, || \, p(\mathbf{z})] \Big] \, , \tag{6.2}$$

$$\approx \frac{1}{S} \sum_{i=1}^{S} \mathrm{D}_{\mathrm{KL}} \left[q_\phi(\mathbf{z}|x_{ij}) | p(\mathbf{z}) \right] \, , \tag{6.3}$$

where we have used Monte Carlo integration and replaced $p_\phi(\mathbf{z}_i|x_{ij})$ with $q_\phi(\mathbf{z}_i|x_{ij})$ for practical and efficient computation [205, 97]. j is the question index, x_{ij} is the i-th student's answer to the j-th question, which can be either binary indicating whether the student has answered it correctly or categorical which is the student's answer choice for this question. $\mathbf{z}$ is the latent embedding of students. The i-th student ability can be determined by the student's possible performance on all questions, which can be inferred from $\mathbf{z}_i$.

This objective measures the information gain of estimating the student ability by conditioning on the answer to question j. When R is large, the question is more informative on differentiating the student ability reflected in the student embedding $\mathbf{z}$, and thus it is considered as high-quality.

6.3.4 Personalized Question Selection

In practical online education scenarios, it is of great interest to adaptively select a small sequence of questions for students. Appropriately choosing these questions allows effective and efficient evaluation of students' abilities at scale.

We formulate the problem of personalized question selection as a Bayesian experiment design problem in an information theoretic manner. Specifically, we are interested in selecting a sequence of questions that is most informative in revealing the student's current state of learning. Inspired by [205, 97], we formulate the following selection strategy which is similar to Eq. 6.3:

$$R_i(j, \boldsymbol{x}_O) = \mathbb{E}_{x_j \sim p(x_j|\boldsymbol{x}_O)} D_{\mathrm{KL}}[p(\mathbf{z}_j|\boldsymbol{x}_O, x_j)||p(\mathbf{z}_j|\boldsymbol{x}_O)] \,. \tag{6.4}$$

Intuitively, this selection strategy selects a question j that provides the most information, defined in KL divergence, on the student knowledge state summarized in $\mathbf{z}_i$. This objective is different from the one used in [205] as we do not have a particular target variable. Note that we have omitted the student index subscript i in Eq. 6.4 for succinctness. Similar to the previous quality metric computation, we use Monte Carlo integration to approximate Eq. 6.4.

The above selection strategy enables selecting a sequence of personalized questions for each student in the following manner (which is different from Eq. 6.3 which can only select a single question). First, we initialize $\boldsymbol{x}_O = \emptyset$. Then, we compute the information reward according to Eq. 6.4 for each question and select question j outside the prediction targets (i.e., $x_j \notin \boldsymbol{x}_\psi$) with the maximum $\widehat{R}$ as the next one for student i. Finally, we set $\boldsymbol{x}_O \leftarrow \boldsymbol{x}_O \bigcup x_j$ and repeat the previous steps until we reach the desired number of selected questions.

6.4 Experiments

In this section, we demonstrate the applicability of our framework on the real-world educational dataset that we have introduced in the dataset section for student answer prediction, question difficulty, and quality quantification, and personalized question selection.

Table 6.1 : Imputation performances of various methods. p-VAE remains a very strong competitor and slightly outperforms all baselines that we consider.

Method	**Accuracy** $\uparrow$	**Mean Abs. Err.** $\downarrow$
Random	0.534	0.471
IRT	0.735	0.359
SVD	0.734	0.358
SVD++	0.737	0.352
Co-clustering	0.731	0.351
NMF	0.737	0.382
p-VAE (ours)	**0.739**	**0.343**

6.4.1 Student Answer Prediction

Setup. We split the students (rows of the data matrix) into the train, validation, and test sets with an 80:10:10 ratio. Therefore, students that are in the test set are never seen in the training set. We train the model on the train set for 50 epochs using Adam optimizer [156] with a learning rate of 0.001. We train p-VAE on binary students' answer records (correct or incorrect answers). To evaluate imputation performance, we supply the trained p-VAE model a subset of the test set as input and compute the model's prediction accuracy and mean absolute error (MAE) on the rest of the test set.

Results. Table 6.1 shows the accuracy of p-VAE trained on binary answer records comparing to various baselines, including random imputation, the Rasch model [273], SVD, SVD++ [162], Co-clustering [90] and Negative Matrix Factorization (NMF) [204]. Note that SVD can also be interpreted as a multivariate Rasch model [38] which is another classic method in educational data mining. We did not compare with vanilla VAEs [157] because, as mentioned previously, they cannot handle an incomplete input data matrix.

We can observe that p-VAE achieves state-of-the-art performance on the dataset, slightly outperforming all baselines. In addition, p-VAE is the only method that not only accurately

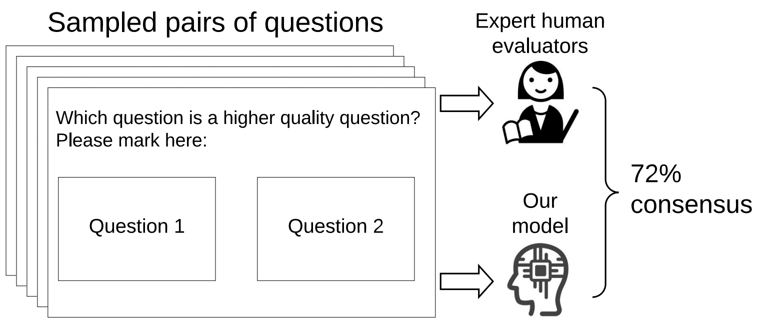


Figure 6.3 : Illustration of question quality evaluation interface for the human evaluator. Our quality metric achieves a maximum of 72% agreement with human evaluators.

predicts students' responses but also computes a posterior distribution over student embeddings which enables the computation of question quality and personalized question selection. The remaining baselines either do not perform as well as p-VAE or do not compute any uncertainty information. There exist Bayesian versions of some baselines, such as Bayesian IRT [317] or Bayesian sparse factor analysis [173] models. Unfortunately, due to the high computational cost, these traditional Bayesian models do not scale to datasets as big as the one that we consider in this work. Thanks to the amortized inference, p-VAE is capable of efficient Bayesian inference at scale.

Implications. Accurately predicting students' answers and estimate uncertainty lies at the core of educational data mining and adaptive testing. p-VAE achieves both with state-of-the-art performances which is a necessary prerequisite to the computation involved in the remaining experiments.

6.4.2 Question Difficulty Quantification

Setup. With the complete data matrix imputed by p-VAE, we compute question difficulty by taking the average of all students' answers including observed and predicted answers. We resort to human evaluation to compare our framework's rankings. We ask the evaluator to provide a full difficulty ranking for all topics. We then compute the difficulty ranking

Table 6.2 : Question quality ranking agreement between various methods and each evaluator (T1 through T5). Our metric achieves the best agreement with every evaluator.

Method	**T1**	**T2**	**T3**	**T4**	**T5**
Random	0.4	0.56	0.44	0.44	0.44
Entropy	0.68	0.64	0.64	0.62	0.64
Ours	**0.72**	**0.64**	**0.68**	**0.62**	**0.72**

Table 6.3 : Spearman Correlation coefficients for question topic difficulty rankings between human expert and model prediction.

Method	**Spearman Correlation**
random ordering	0.060
majority imputation	0.076
using observation	0.395
p-VAE imputation	**0.738**

using our framework and compute the Spearman correlation coefficient as a measure of the level of agreement between the model's and the expert's difficulty rankings.

Results. Table 6.3 shows the Spearman correlation coefficients comparing expert's topic and scheme rankings, respectively, to our framework's and two other baselines' rankings. The baselines include random ordering, using majority imputation to fill the data matrix, and using the observed data alone. We see that our framework's ranking closely matches the human expert's ranking while baselines do not produce rankings that are any close to the expert's ranking.

6.4.3 Question Quality Quantification

Setup. We compute question quality according to Eq. 6.3 and compare the pairwise rankings of question quality to those provided by human evaluators. Our 5 evaluators are all highly respected math teachers who have no prior information about this work. We resort to

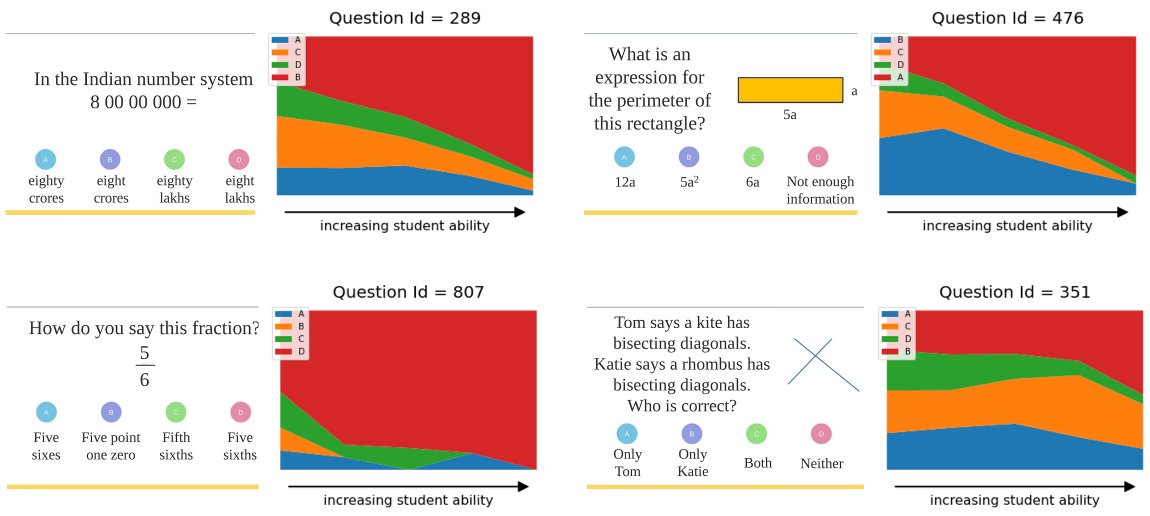


Figure 6.4 : Two examples of high-quality questions (top row) and two examples of low-quality questions (bottom row) determined by our framework. For each pair, the left image shows the actual question, and the right image shows the stacked portion plot indicating the percentage of students who answered A, B, C, or D, where the top portion (red) always indicates the percentage of students who answered correctly. **Top row**: high-quality questions differentiate students' abilities. **Bottom left**: this low-quality question is too easy; students tend to answer it correctly despite their ability. **Bottom right**: these low-quality questions are either too easy or too difficult; the majority of the students tend to answer them either correctly or incorrectly despite their ability.

pairwise comparison, i.e., we give all 5 evaluators a pair of questions and ask them to give a preference on which question is of higher quality. We then compute the number of times our model agrees with each evaluator's choice of quality. Because this evaluation requires significant domain knowledge and because there are more than ten thousand questions and thus many more possible pairs, we only sample 80 pairs from a subset of our full dataset. Although limited, the evaluation of these samples provides preliminary evidence on whether the proposed quality metric agrees with domain experts. Figure 6.3 illustrates the interface that the evaluator sees when they provide the question quality labels. Each evaluator performed this task in isolation without knowledge of the other evaluators' labels.

We also consider two baseline metrics for question quality. The first metric randomly

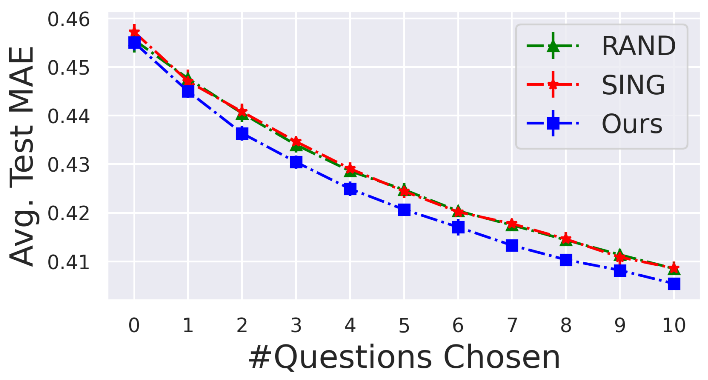


Figure 6.5 : Our personalized question selection strategy outperforms baseline strategies for each question selected.

assigns a ranking to each question. The second metric computes the entropy of a question where the probability is the portion of correct or incorrect answers among all answer records for this question.

Results. Table 6.2 shows the agreement between the models' question quality rankings and those provided by each of the 5 evaluators. We can see that our proposed quality metric achieves the highest agreement with human judgment. Also note that the entropy baseline, which is also inspired by information theory, although outperformed by our metric, is significantly better than random rankings. This preliminary result implies that further development of better question quality metrics via similar information-theoretic approaches has the potential to accurately capture human judgment of question quality and is a promising direction to pursue.

To visualize the high and low-quality questions that our framework selects, we show in Figure 6.4 two examples of high-quality questions (top row) and two examples of low-quality questions (bottom row) determined by our framework. For each pair, the left image shows the actual question, and the right image shows the stacked portion plot. The stacked portion plot shows the percentage of students in different ability ranges who have answered

Table 6.4 : Example of the topics of the 10 questions that our framework chooses for 2 students.

Questions	**Student #1**	**Student #2**
Q1	Squares, Cubes, etc.	Perimeter and Area
Q2	Simplifying Expressions by Collecting Like Terms	Line Symmetry
Q3	Squares, Cubes, etc	Missing Lengths
Q4	Mental Multiplication and Division	Measuring Angles
Q5	Mental Multiplication and Division	Line Symmetry
Q6	Factors and Highest Common Factor	Basic Angel Facts
Q7	Factors and Highest Common Factor	Angles
Q8	Factors and Highest Common Factor	Measuring Angles
Q9	Simplifying Expressions by Collecting Like Terms	Perimeter and Area
Q10	Simplifying Expressions by Collecting Like Terms	Transformations

the question correctly. (the correct answer choice is always at the top, i.e., the red color part of the plot, and the remaining colors are the remaining three incorrect answer choices). The stacked portion plot is produced using the observed students' answer choices to the questions (i.e., A, B, C, or D choices).

In addition to the question content itself, we can gain some insights by examining and comparing the stacked portion plots. For example, We can see that high-quality questions better test the variability in students' abilities because fewer students with a lower ability score can answer them correctly, whereas more students with a higher ability score can answer them correctly. This phenomenon is not present in lower quality questions, where most of the students, regardless of their ability score, tend to answer them either correctly or incorrectly.

Implications. Our difficulty and quality metrics efficiently and accurately provide data-driven analytics on questions which may help teachers in designing quizzes, homework sets, and exams that best suit their classes.

6.4.4 Personalized Question Selection

Setup. To quantitatively evaluate the performance of our selection strategy, we proceed as follows. We first sample a subset of students from the dataset and sample 10% of each student's answer records as the prediction targets. For each student, we then sequentially choose 10 questions as prescribed in Eq. 6.4, reveal their answers, and predict the student's answers using these 10 revealed answers through p-VAE. We report the average mean absolute error over all the sampled answer records at each of the 10 steps in the question selection process. We consider two baselines, a random selection strategy (RAND) and a single global optimal strategy (SING) which is similar to our proposed strategy but the reward is averaged over all students. Therefore, this strategy only selects one single sequence of questions for all students and cannot achieve personalization.

Results. Figure 6.5 reports the average MAE comparing our proposed strategy with RAND and SING after 10 runs. Our strategy achieves lower MAE at every step of the question selection process, demonstrating its effectiveness in choosing a sequence of questions that minimizes prediction error.

To demonstrate that our strategy is personalized for each student, we show in figure 6.6 the sequence of questions for 10 students. The x-axis is the step (number) of the question selected and the y-axis is the question ID. Each color represents the selected question sequence for one student. We can see that our strategy picks a very different set of questions for these 10 students. We also show the topics of the selected questions for two students in Table 6.4 to gain some understanding of how our strategy chooses questions. We can see that the questions chosen for these two students are very different: the questions for the first students emphasize a few recurring topics while the questions for the second students cover a diverse set of topics.

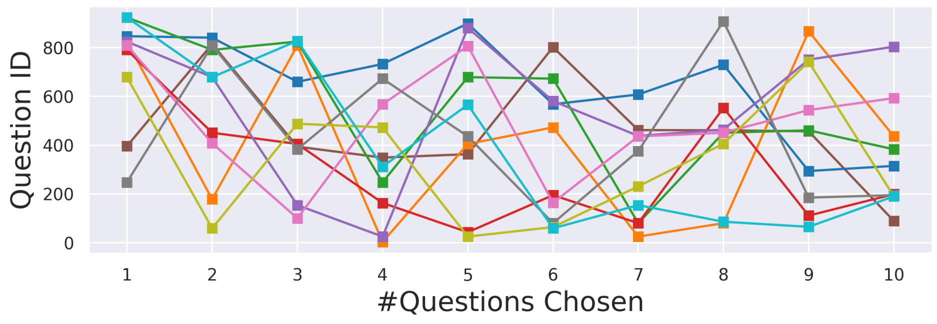


Figure 6.6 : Illustration of the questions that our framework selects for 10 students. Each student gets a personalized sequence of questions.

Implications. Our question selection strategy provides an efficient and automatic method to sequentially select questions for students. This can be used in large-scale learning and adaptive testing scenarios in which teachers cannot attend to each individual student but still would like to have a quick way to assess each student's skills.

6.5 Related Work

AI in Education. There is a vast and rapidly expanding literature on using AI for advancing education. A sample of examples includes knowledge tracing [175, 273, 341, 173, 234, 258]; various automation including grading [363, 174], feedback generation [374] and quiz question generation [361]; and understanding students' online and offline behaviors such as collaboration [362] and cheating [181].

Most related to our work is prior literature on acquiring educational insights by analyzing students' answer records to questions. Notable examples include the Rasch model [273] that outputs a scalar question difficulty and a scalar student ability; the sparse factor analysis (SPARFA) model [175] which extends the Rasch model to output multi-dimensional question and student analytics; the time-varying SPARFA model [173] capable of processing time-varying students' answer records which is a more realistic scenario. These models, however, are limited to their high computational complexity and thus are difficult to apply to large scale educational data analysis. Our work complements prior work in that we develop a framework capable of efficiently analyzing large educational data sets while at the same time extracting educationally meaningful insights.

Missing Data Imputation. The missing Data Imputation method is used in many real-world applications, such as recommender systems. Many existing approaches for recommender systems rely on linear methods because of their efficiency and scalability [283, 250, 312]. More recent literature introduces nonlinear models using deep learning for improved model capacity, notably with variational autoencoders [49, 147, 188, 236, 347]. However, many of these models do not handle missing data without some ad-hoc modifications to the data, i.e., zero imputation as in [166, 192, 291]. Such an ad-hoc way of replacing missing values in the data matrix significantly changes the original data distribution and can negatively impact the imputation results. In contrast to the above methods, p-VAE leverages amortized Bayesian inference with a special model architecture to efficiently handle missing data and quantify uncertainty, which is ideal for our application.

Our work builds on [206, 205, 97, 219], all of which take advantage of the desirable properties of p-VAE for recommender systems and information acquisition framework, respectively. Our work further advances such prior work in that we extend, to the first of our knowledge, p-VAE to the application of education data mining with a novel information criterion to extract educationally meaningful insights.

6.6 Conclusions

In this work, we develop a framework to analyze questions in online education platforms on a large scale. Our framework combines the recently proposed partial variational auto-encoder (p-VAE) for efficiently processing large scale, partially observed educational datasets, and novel metrics and strategy for automatically producing a suite of meaningful and actionable insights about quiz questions. We demonstrate the applicability of our framework on a real-world educational dataset, showcasing the rich and interpretable information including question difficulty, question quality, and personalization that our framework obtains from millions of students' answer records to multiple-choice questions.

Our framework is highly flexible, which enables further improvements and extensions to obtain richer educational insights. For example, one extension is to customize the information-theoretic metrics for extracting various other information of interest. Another extension is to adapt the p-VAE model for time-series data, where we can work with a more realistic yet challenging scenario that students' states of knowledge change over time. To facilitate and encourage future research, we have open sourced our dataset in the form of a competition for AI in education; see `https://eedi.com/projects/neurips-education-challenge` for more details.

Chapter 7

Open-Ended Knowledge Tracing for Computer Science Education

7.1 Introduction

Knowledge tracing (KT) [60] refers to the problem of estimating student mastery of concepts/skills/knowledge components from their responses to questions and using these estimates to predict their future performance. KT methods play a key role in many of today's large-scale online learning platforms to automatically estimate the knowledge levels of a large number of students and provide each of them with personalized feedback and recommendation, leading to improved learning outcomes [278]. KT methods consist of two essential components. First, a *knowledge estimation* (KE) component, i.e.,

$$\mathbf{h}_{t+1} = \text{KE}((\mathbf{p}_1, \mathbf{x}_1), \ldots, (\mathbf{p}_m, \mathbf{x}_t)), \tag{7.1}$$

estimates a student's current knowledge state $\mathbf{h}_{t+1}$ using questions ($\mathbf{p}$) and responses ($\mathbf{x}$) from previous (discrete) time steps for this student. Second, a *response prediction* (RP) component predicts the student's response to the next question (or future questions), i.e., $\mathbf{x}_{t+1} \sim \text{RP}(\mathbf{h}_{t+1}, \mathbf{p}_{t+1})$. Section 7.4 contains a detailed overview of existing KT methods and how the question, responses, and knowledge state variables are represented.

One key limitation of almost all existing KT methods is that they only analyze and predict *binary-valued* student responses to questions, i.e., the *correctness* of the response.

That is, the RP is typically a simple binary classifier. As a result, one can broadly apply KT methods to any question as long as student responses are graded. However, this approach loses important information regarding student mastery, since it does not make use of the *content* of questions and student responses, especially for open-ended questions. Past work has shown that students' open-ended responses to such questions contain useful information on their knowledge states, e.g., having a "buggy rule" [28], exhibiting misconceptions [79, 80, 307], or a general lack of knowledge [9]; this information is highly salient for instructors but cannot be captured by response correctness alone.

Generative language models such as GPT [31] provide an opportunity to fully exploit the rich information contained in open-ended student responses in various domains for the purposes of KT. In this work, we focus on computer science education, where short programming questions require students to write code chunks that satisfy the question's requirements. The program synthesis capabilities of variants of pre-trained neural language models such as CodeX [43] enable the generation of short chunks of code from natural language instructions, which we can leverage for open-ended response prediction. However, two key challenges make this task difficult: First, as students learn through practice, their knowledge on different programming concepts is *dynamic*; students can often learn and correct their errors given instructor-provided feedback or even error messages generated by the compiler. Therefore, *we need new KE models that can effectively trace time-varying student knowledge throughout their learning process.* Second, student-generated code is often *incorrect and exhibits various errors*; there may also exist multiple correct responses that capture different lines of thinking among students. This intricacy is not covered by program synthesis models, since their goal is to generate correct code and they are usually trained on code written by skilled programmers. Therefore, *we need new RP models that can generate student-written (possibly erroneous) code that reflects their (often imperfect)* *knowledge of programming concepts.*

7.1.1 Contributions

In this work, we present the first attempt at analyzing and predicting exact, open-ended student responses, specifically for programming questions in computer science education.[1] Our contributions can be summarized as follows:

- We define the **open-ended knowledge tracing (OKT)** framework, a novel KT framework for open-ended student responses, and a new KT task, exact student response prediction. We ground OKT in the domain of computer science education for student code submission analysis and prediction but emphasize that *OKT can be broadly applicable to a wide range of subjects that involve open-ended questions*.
- We develop an initial solution to the OKT task, a **knowledge-guided** code generation method. Our method combines KE components in existing binary-valued KT methods with code generation models, casting the OKT task as a *dynamic controllable generation* problem where the control, i.e., time-varying student knowledge states, are also learned.
- Through **extensive experiments on a real-world student code dataset**, we explore the effectiveness of OKT in reflecting variations of student code and especially errors in its knowledge state estimates. We explore the effectiveness of our solution in making reasonably accurate predictions of student-submitted code. We also discuss how these OKT capabilities can help computer science instructors and outline several new research directions.

[1]Find our code at `https://github.com/lucy66666/OKT`

7.2 OKT for Computer Science Education

We now define the OKT framework and detail specific model design choices in the domain of computer science education, where we focus on analyzing students' code submissions to programming questions. Figure 7.1 illustrates the three key components of OKT: knowledge representation (KR), KE, and response generation (RG), the last of which is the key difference between OKT and existing KT methods. Our key technical challenges are (i) how to represent programming questions and student code submissions (KR, Section 7.2.1) and use them to estimate student knowledge states (KE, Section 7.2.2); (ii) how to combine knowledge states with the question prompt to generate student code (RG, Section 7.2.3); and (iii) how to efficiently perform optimization to train the OKT model components (Section 7.2.4).

7.2.1 Knowledge Representation (KR)

The purpose of the KR component is to convert the prompt/statement of questions that students respond to and their corresponding code submissions to continuous representations. Our KR component is significantly different from existing binary-valued KT methods that ignore question/response content and one-hot encode them using question/concept IDs and response correctness.

Question Representation. We adopt the popular GPT-2 model[2] [268] for prompt representation: Given a question prompt $\mathbf{p}$, GPT-2 tokenizes it into a sequence of M word tokens, where each token has an embedding $\bar{\mathbf{p}}_m \in \mathbb{R}^K$. For GPT-2, the dimension of these embeddings is $K = 768$. This procedure produces a sequence of token embeddings $\{\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2, \ldots, \bar{\mathbf{p}}_M\}$. We then average the embeddings of each prompt token to get our prompt

[2]One can use any language model; we choose GPT-2 since our RG component for student code is also built on GPT-2.

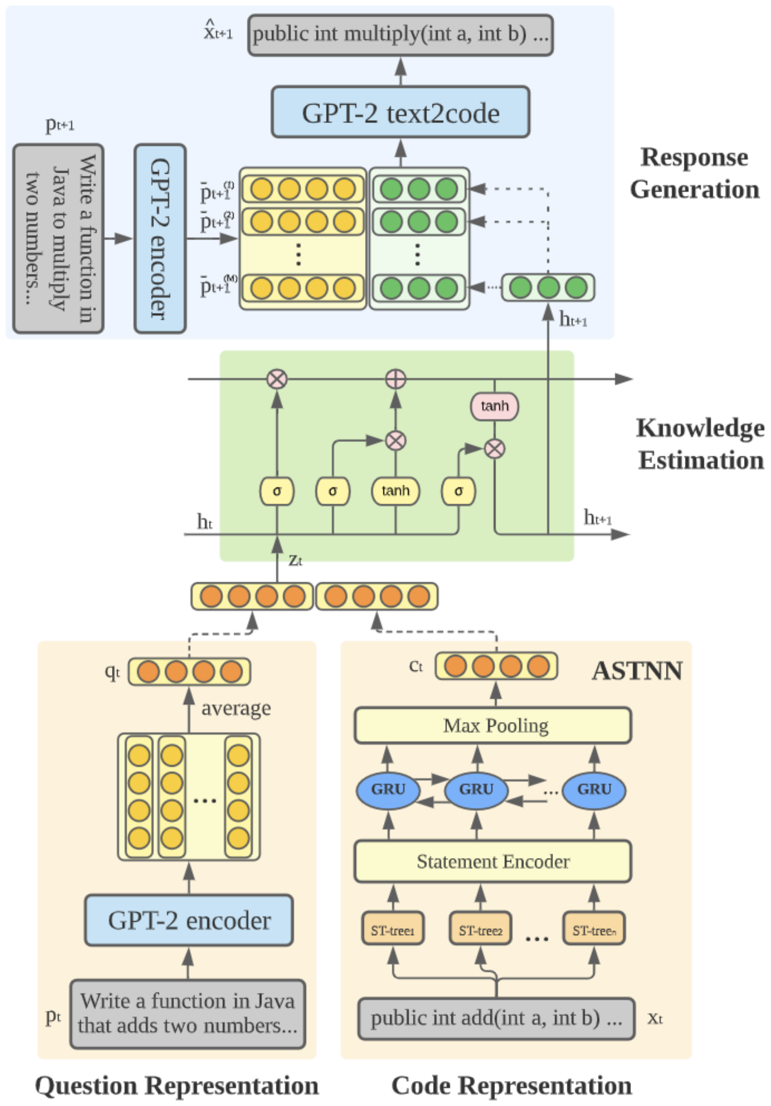


Figure 7.1 : Open-ended knowledge tracing (OKT) block diagram. We update the student's current knowledge state $\mathbf{h}_{t+1}$ using the last question $\mathbf{p}_t$ and actual student code $\mathbf{x}_t$. We then combine it with the next question statement $\mathbf{p}_{t+1}$ to generate our prediction of the actual student code $\hat{\mathbf{x}}_{t+1}$.

embedding $\mathbf{q} = \sum_{m=1}^{M} \frac{\bar{\mathbf{p}}_m}{M}$, where the average is computed element-wise on vectors.

Code Representation. In order to preserve both semantic and syntactic properties of programming code in embedding vectors, we utilize ASTNN [398], a popular tool for code representation. We first parse student-submitted code into an abstract syntax tree (AST). We then split each full AST into a sequence of non-overlapping statement trees (ST-trees) through preorder traversal. Each ST-tree contains a statement node as the root and its corresponding AST nodes as children. We then pass the ST-trees through a recurrent statement encoder to obtain embedding vectors and use a bidirectional gated recurrent unit network [53] to capture the naturalness of the statements and further enhance the capability of the recurrent layer. Eventually, we apply a max-pooling layer to capture the most important semantics for each dimension of the embedding. We denote this entire process as $\mathbf{c} = \text{ASTNN}(\mathbf{x})$ where $\mathbf{x}$ is student-submitted code and $\mathbf{c}$ is its code embedding vector, which we use as input to the KE component. We refer readers to [398] for more details on ASTNN.

7.2.2 Knowledge Estimation (KE)

The purpose of the KE component is to turn a student's past question/code information into estimates of their current knowledge state. Following DKT [258], a popular existing KT method, we use a long short-term memory (LSTM) model [117] to update a student's current knowledge state, $\mathbf{h}_{t+1}$, given their previous response at the last time step. We use the output of the KR component, i.e., question prompt and code embeddings, as the input to the KE component as $\mathbf{h}_{t+1} = \text{LSTM}(\mathbf{h}_t, \mathbf{q}_t, \mathbf{c}_t)$ and use it as input to the RG component to generate predicted student code submissions. In principle, we can use any existing binary-valued KT method as OKT's KE component. We validate in our experiments (Section 7.3) that OKT is compatible with two other popular KT methods, DKVMN [399] and AKT [92], that are based on external memory and attention networks.

7.2.3 Response Generation (RG)

The purpose of RG, OKT's core component, is to **predict open-ended responses**, i.e., generate predicted student code, which makes OKT significantly different from existing binary-valued KT methods with binary classifiers of response correctness. We fine-tune a base GPT-2 generative model into a text-to-code model P_Θ with parameter Θ on code data (see Section 7.2.5 for details). We choose language models over other code generation approaches since their text-to-code generation pipeline suits OKT well.

Our key technical challenge is how to use knowledge states as *control* in the code generation model to guide personalized code predictions for each student. Given the current question prompt, $\mathbf{p}_{t+1}$, and its sequence of M token embeddings $\{\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2, \ldots, \bar{\mathbf{p}}_M\}$, where $\bar{\mathbf{p}}_m \in \mathbb{R}^K$ (we drop the time step index t in prompt tokens for clarity), our approach for injecting student knowledge states into the code generation model is to replace raw token embeddings with *knowledge-guided* embeddings using an *alignment* function, i.e., $\mathbf{p}_m = f(\bar{\mathbf{p}}_m, \mathbf{h}_{t+1})$ for $m = 1, \ldots, M$. Therefore, the GPT-2 input embeddings are

$$\{\mathbf{p}_1, \ldots, \mathbf{p}_M\} = \{f(\bar{\mathbf{p}}_1, \mathbf{h}_{t+1}), \ldots, f(\bar{\mathbf{p}}_M, \mathbf{h}_{t+1})\}.$$

Intuitively, this (possibly learnable) alignment function aligns the space of knowledge states with the space of textual embeddings for the question prompt. Thus, knowledge states are responsible for predicting different code submitted to the same programming question by different students.

We explore four different alignment functions to combine knowledge states with question prompt token embeddings:

- Addition, i.e., $\mathbf{p}_m = \bar{\mathbf{p}}_m + \mathbf{h}_{t+1}$.
- Averaging, i.e., $\mathbf{p}_m = (\bar{\mathbf{p}}_m + \mathbf{h}_{t+1})/2$.

- Weighted addition, i.e., using a learnable weight for knowledge states, $\mathbf{p}_m = \bar{\mathbf{p}}_m + \alpha \cdot \mathbf{h}_{t+1}$.
- Linear combination, i.e., applying a learnable affine transformation to the knowledge states before adding it to token embeddings, $\mathbf{p}_m = \bar{\mathbf{p}}_m + \mathbf{A}\mathbf{h}_{t+1} + \mathbf{b}$.

The latter two functions are learnable with parameters $\alpha \in \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{D \times K}$, and $\mathbf{b} \in \mathbb{R}^K$. Therefore, the predicted student code (with N code tokens), $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, is generated in an autoregressive manner by the RP component given the knowledge-guided question prompt token embeddings $\{\mathbf{p}_1, \ldots, \mathbf{p}_M\}$.

7.2.4 Optimization

During the training process, we jointly optimize the parameters of the KE and RG components of OKT; in essense, we are learning *both* a controllable generation model for student responses *and* the control itself, which is the student's time-varying knowledge state. We keep the knowledge representation encoders E_1 and E_2 fixed. The objective for one student code submission is given by

$$\begin{aligned} Loss = \textstyle\sum_{n=1}^{N} - \log P_\Theta\big(x_n | \, \{\mathbf{p}_1, \ldots, \mathbf{p}_M\}, \\ \{x_{n'}\}_{n'=1}^{n-1}\big), \end{aligned} \tag{7.2}$$

where Θ denotes the set of parameters in the RP component, including both the GPT-2 text-to-code model parameters and learnable parameters in the alignment function $f(\cdot)$. The final training objective is the sum of this loss over all code submissions made by all students.

We also design an efficient training setup for OKT. For existing neural network-based KT methods, at each training step, we use a batch of student (question, response) sequences to compute the correctness prediction loss across all time steps and all students in the batch.

We cannot use this training method since OKT's loss for one student is the sum of code prediction losses over all time steps, whereas the loss at each time step is itself the sum of a sequence of cross entropy losses for code token predictions. As a result, if we use the training setup for existing KT methods, at each training step, we need to call the response generator for a total of $T \times B$ times where T is the number of time steps and B is the batch size, which will significantly slow down training. Instead of batching over students, we use a batch of (student, time step) pairs. Then, at each training step, we first apply the knowledge update component in OKT to compute the knowledge states for students in the batch, extract the knowledge states corresponding to the sampled time steps in the batch, and then feed them into the response generator. This setup enables efficient training for OKT.

7.2.5 Pre-training Models

Before training OKT, we pre-train its KE component using the binary-valued correctness prediction loss with question and code embeddings as input, following [213, 412]. Since we cannot directly use CodeX [43] due to our need to adjust the input embeddings with student knowledge states, we pre-train a text-to-code pipeline by fine-tuning a standard GPT-2 model on the Funcom dataset [180], which contains 2.1 million Java code snippets and their textual descriptions.

7.3 Experiments

We now present a series of experiments to explore the capabilities of OKT. We first introduce the dataset, various quantitative metrics on which we evaluate various methods, and detail quantitative results. We then qualitatively illustrate that OKT (i) learns a meaningful latent student knowledge space and (ii) generates predicted student code that capture their coding patterns and error types.

Dataset. We use the dataset from the CSEDM Data Challenge, henceforth referred to as the **CSEDM dataset**.[3] To our knowledge, this is the only college-level, publicly-available dataset *with students' actual code submissions*; a concurrent work [305] uses the Hour of Code dataset, which has some similarities with this dataset but only has two questions. The CSEDM dataset contains 246 college students' 46,825 full submissions on each of the 50 programming questions over the course of an entire semester. The dataset contains rich textual information on question prompt and students code submissions as well as other relevant metadata such as the programming concepts involved in each question and all error messages returned by the compiler. See Section B.1 in the Appendix for detailed data statistics and preprocessing steps.

Evaluation Metrics. In the context of predicting students code submissions, we need a variety of different metrics to fully understand the effectiveness of OKT. We thus use two types of evaluation metrics. First, we need metrics that can measure OKT's ability to predict student code on the test set after training. For this purpose, we use two metrics, including **CodeBLEU** [275], a variant of the classic BLEU metric adapted to code that measures the similarity between predicted code and actual student code. The other metric is the average **test loss** across code tokens computed using OKT methods with the lowest validation loss. Second, we need metrics that can measure the diversity of predicted student code since we do not want OKT to simply memorize frequent student code in the training data. For this purpose, we use the **dist-**N metric [185] that computes the ratio of unique N-grams in the predictions over all N-grams. We choose $N = 1$ in this work since uni-gram setting is more compatible with the limited coding vocabulary size. We note that predicting whether a student code submission passes test cases is another important task for OKT evaluation;

[3]Challenge: `https://sites.google.com/ncsu.edu/csedm-dc-2021/`. The dataset is called "Code-Workout data Spring 2019" in Datashop (`pslcdatashop.web.cmu.edu`).

however, since test cases are not included in the CSEDM dataset, we cannot conduct this evaluation and leave it for future work.

Methods for Comparison. Since exact student code prediction is a novel task, there are **no existing baselines** that we can compare against. We thus compare among variants of OKT to demonstrate that it is highly flexible and extensible. First, we test three different existing binary-valued KT methods, DKT, DKVMN, and AKT, as the KE component; one can apply any existing binary-valued KT method as the KE component that is suitable. As a strong baseline, we also test a version of OKT without KE and use the question prompt and code embeddings from the previous time step as additional input to the text-to-code RG component. Second, we compare different alignment functions between the knowledge and question prompt embedding spaces listed in Section 7.2.3. Third, we compare several training settings, including pre-training the KE and RG components and using a multi-task training objective by adding the binary-valued response correctness prediction loss to the code generation loss in Eq. 7.2, following [7].

Experimental Setup. Following typical settings in the KT literature, our goal is to predict the code a student submits to a question at the next time step t, $\mathbf{x}_{t+1}$, given their question prompts and code submissions in all previous time steps, i.e., $(\mathbf{p}_1, \mathbf{x}_1), \ldots, (\mathbf{p}_t, \mathbf{x}_t)$. We use two experimental settings in our experiments that capture different aspects of OKT: First, we analyze only the first submission to each question, ignoring later attempts. In this setting, knowledge states mostly capture a student's overall mastery of programming concepts. Second, we analyze all code submissions from each student, including multiple consecutive attempts at the same question. In this setting, knowledge states capture not only a student's programming concept mastery but also their debugging skills. We choose not to study only the final attempt since most students were able to submit correct code in the

Table 7.1 : OKT results comparing different KT models as the KE component of OKT. AKT slightly outperforms DKVMN while DKT performs best under both settings.

setting	KT model	CodeBLEU ↑	Dist-1 ↑	Test Loss ↓
first submission	**DKT**	0.690	0.422	0.178
	AKT	0.581	0.401	0.193
	DKVMN	0.580	0.388	0.196
	None	0.518	0.426	0.215
all submissions	**DKT**	0.726	0.403	0.111
	AKT	0.632	0.396	0.125
	DKVMN	0.570	0.399	0.135
	None	0.471	0.385	0.151

end. See Section B.2 of the Appendix for detailed experimental settings. Additionally, we perform another experiment on predicting student code submissions to new questions that are unseen during training; see Section 7.3.4 for details.

7.3.1 Quantitative Results

Table 7.1 shows the quantitative results evaluating OKT on the CSEDM dataset comparing DKT, AKT, and DKVMN as the KE component, averaged over all students and time steps. Overall, we observe that our initial OKT method performs reasonably well; as a reference, the CodeBLEU value for the examples in Table 7.3 are 0.8 and 0.65, respectively. Using existing binary-valued KT methods as the KE component significantly outperforms the baseline that relies on a standard text-to-code generation pipeline without this component, which suggests that KT is a key component in student-generated code prediction. Across the two experimental settings, analyzing first submissions leads to higher test loss and lower CodeBLEU score than analyzing all submissions, while performance on the Dist-1 metric does not vary much. These results can be explained by our observation that students rarely make substantial changes to their code across different submissions, often making minor tweaks; therefore, predicting a later code submission given the previous submissions becomes an easier task than predicting the first submission to a new question. Since these metrics are computed over all questions, we break down OKT's performance across questions in Section B.3 of the Appendix; performance varies significantly across questions (between 0.55 and 0.85 on CodeBLEU). This observation suggests that there is considerable room for improvement on the task of exact student code prediction since they have many nuanced variations, which we further illustrate in the qualitative experiments below.

We also see that using using DKT as the KE component of OKT significantly outperforms using AKT and DKVMN on all metrics in both experimental settings, while using AKT also outperforms DKVMN. These results suggest that DKT is more effective than AKT or DKVMN as the KE component of OKT, which also reported in [412] for standard binary-valued KT on programming exercises, likely because DKT relies on a simple and robust LSTM model. In contrast, AKT and DKVMN have complicated model architectures and may require further parameter tuning and/or more training data in the context of OKT; typical binary-valued KT datasets are much larger in scale (up to $\sim$10M responses [55]).

Table 7.2 shows the quantitative results comparing different OKT designs and training settings with DKT as the KE component on first submissions. First, we see that aligning the knowledge state space with the prompt token embedding space with a learnable linear function is the most effective (with p-value of 0.01 for CodeBLEU), although other alignment functions are only slightly worse. Developing better alignment functions may further improve performance, which we leave for future work. Second, we see that pre-training the KE and RG components result in limited improvement in OKT's performance. This result suggests that there are significant differences between (i) the nature of the binary-valued KT task and OKT's exact code prediction task and (ii) code written by professionals and by students who are still learning programming. Third, we see that a multi-task OKT

Table 7.2 : Linearly combining knowledge states and the prompt token embeddings, pre-training both KE and RG components, and using a multi-task loss lead to best OKT performance.

		CodeBLEU ↑	**Dist-1 ↑**	**Test Loss ↓**
Alignment	add	0.681 ± 0.003	0.423 ± 0.004	0.179 ± 0.006
	average	0.680 ± 0.003	0.425 ± 0.003	0.179 ± 0.006
	weight	0.684 ± 0.008	0.422 ± 0.004	0.182 ± 0.007
	linear	0.696 ± 0.005	0.425 ± 0.004	0.178 ± 0.006
Pre-train LSTM	yes	0.681 ± 0.003	0.423 ± 0.004	0.179 ± 0.006
	no	0.678 ± 0.003	0.425 ± 0.002	0.180 ± 0.004
Pre-train GPT	yes	0.702 ± 0.004	0.423 ± 0.003	0.174 ± 0.003
	no	0.678 ± 0.005	0.415 ± 0.004	0.219 ± 0.006
Multi-task	yes	0.706 ± 0.002	0.423 ± 0.002	0.362 ± 0.008
	no	0.664 ± 0.019	0.426 ± 0.008	0.198 ± 0.009

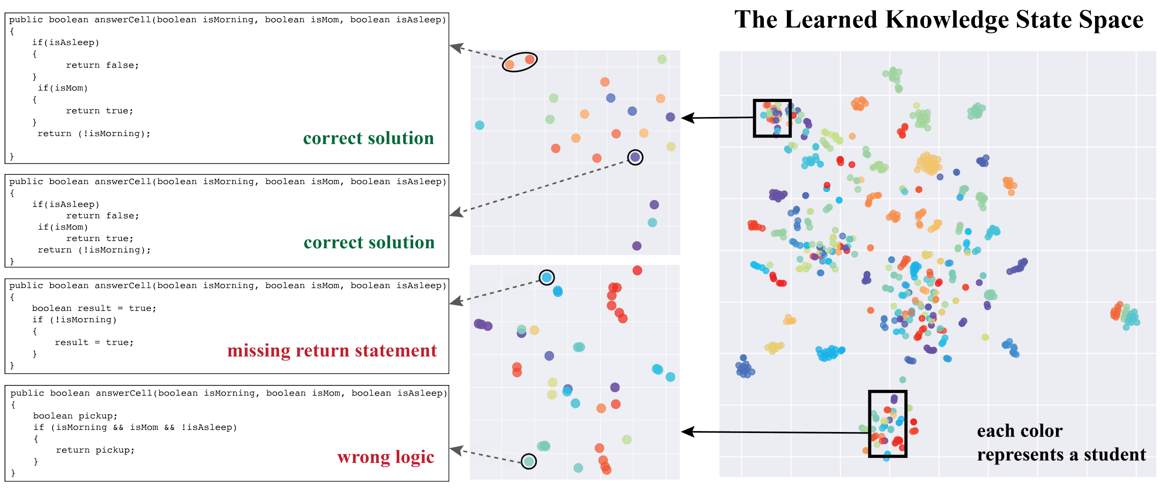
training objective improves both code prediction performance and model robustness in our experiments. (with p-value of 0.018) This result suggests that multi-task learning with multiple objectives helps us learn better representations of the data, i.e., student knowledge state representations, in OKT.

7.3.2 Interpreting Learned Knowledge States

We now use a case study to show that the knowledge state space learned by OKT captures the variation in the content and structure of student code. Figure 7.2 visualizes the learned knowledge states, projected to a 2-D space via t-SNE [331], for the following question:

`Write a function in Java that implements the following logic: Your cell phone rings. Return true if you should answer it. Normally you answer, except in the morning you only answer if it is your mom calling. In all cases, if you are asleep, you do not answer.`

The right part of Figure 7.2 shows the knowledge states of all students when they respond



```
public boolean answerCell(boolean isMorning, boolean isMom, boolean isAsleep)
{
    if(isAsleep)
    {
          return false;
    }
     if(isMom)
    {
          return true;
    }
     return (!isMorning);

}
```

```
public boolean answerCell(boolean isMorning, boolean isMom, boolean isAsleep)
{
    if(isAsleep)
          return false;
     if(isMom)
          return true;
     return (!isMorning);
}
```

```
public boolean answerCell(boolean isMorning, boolean isMom, boolean isAsleep)
{
    boolean result = true;
    if (!isMorning)
    {
        result = true;
    }
}
```

```
public boolean answerCell(boolean isMorning, boolean isMom, boolean isAsleep)
{
    boolean pickup;
    if (isMorning && isMom && !isAsleep)
    {
        return pickup;
    }
}
```

Figure 7.2 : Visualization of latent student knowledge states (best viewed in color; each color corresponds to one student) and corresponding actual code. Knowledge states reflect the variation in student-generated code.

to this question, where each dot represents the submission at a time step (a student may have multiple submissions at multiple time steps) and each color represents a student. We see that there are distinct clusters in these knowledge states that correspond to different student code. To further demonstrate this observation, we zoom in into two areas in the knowledge state space, shown in the two small plots on the left part of Figure 7.2 together with the corresponding actual student code submissions. We clearly see that the codes within each cluster share similar structural and syntactic properties and that codes from different clusters differ significantly. See Section B.4 for a case study on how OKT's knowledge state space captures student code revisions across multiple submissions. These results suggest that the OKT-learned knowledge state space *aligns* with actual student code submissions.

In Figure 7.3, we compare the learned knowledge state space for OKT against that for existing KT methods. We see that binary-valued DKT learns knowledge states that belong to a few highly overlapping groups with little difference within each group. The KT method in [213] that uses code embeddings only as *input* to binary-valued KT learns a slightly

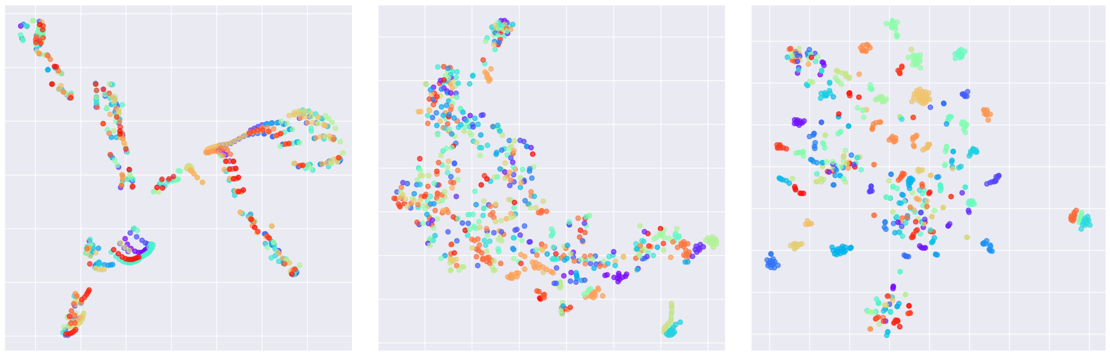


Figure 7.3 : Comparison of the knowledge state spaces learned by DKT (left), DKT with code embeddings as input [213] (middle), and OKT (right). OKT learns a knowledge space with distinct clusters that capture variations in actual student code.

more disentangled knowledge state space. In contrast, OKT's knowledge state space is highly informative with obvious clusters that correspond to actual student code. Overall, these results demonstrate that the knowledge state space learned by OKT captures important aspects of programming knowledge for each student. Therefore, OKT has potential in student and instructor-facing tasks such as hint generation and predicting when a student gets stuck and needs help. We can use OKT in a human-in-the-loop process for student modeling: First, OKT can identify clusters among student responses in an *unsupervised* way. Then, instructors and domain experts can *supervise* OKT by providing fine-grained concept or error labels on these clusters to further interpret the latent knowledge state space.

7.3.3 Knowledge-aware Prediction of Students' Code Submissions

We now use a case study to demonstrate OKT's ability to predict student-submitted code. Similar to most existing text-to-code models [129, 203], exact prediction of the actual student code is very difficult. However, OKT can still be effective in capturing coding styles and even predicting some error types with the help of the learned knowledge states. Table 7.3 shows the predicted code vs. actual student code for two questions. For the top

Table 7.3 : OKT generated code vs. actual student code for two questions (differences highlighted in red boxes).

predicted code	actual student code

`public String zipZap(String str)`
`{`
`  for (int i = 0; i < str.length()- 2; i++)`
`    {`
`    if (str.charAt(i) == 'z' &&`
`        str.charAt(i + 2) == 'p')`
`    {`
`     str.replace("", str.substring(i + 1));`
`    }`
`  }`
`  return str;`
`}`
	`public String zipZap(String str)`
	`{`
	`  for (int i = 0; i < str.length() - 2; i++)`
	`  {`
	`    if (str.charAt(i) == 'z' &&`
	`        str.charAt(i + 2) == 'p')`
	`    {`
	`      str.replace("0", str.substring(i + 1));`
	`      return str;`
	`    }`
	`  }`
	`  return str;`
	`}`

`public boolean evenlySpaced(int a, int b, int c)`
`{`
`    int diffone = b - a;`
`    int difftwo = c - b;`
`    if (diffone = difftwo) {`
`        return true;`
`    }`
`    else {`
`        return false;`
`    }`
`}`
	`public boolean evenlySpaced(int a, int b, int c)`
	`{`
	`    int diffone = b - a;`
	`    int difftwo = c - b;`
	`    boolean question = false;`
	`    if (diffone == difftwo) {`
	`        question = true;`
	`        return question;`
	`    }`
	`    else {`
	`        return question;`
	`    }`
	`}`

example, we see that our generation model is able to predict the student's code structure, capturing their use of *for* loops (instead of another popular choice of *while* loops). In the bottom example, we see that while code prediction for this question is less accurate than for the first question, OKT can still capture the main logic and most important parts of the student's actual code. These examples show that OKT can capture both code structure and knowledge gaps on programming concepts for individual students and even predict their possible errors; this capability has much more potential for student and instructor support than standard binary-valued KT methods.

7.3.4 Generalizing to Unseen Questions

One important limitation of binary-valued KT methods is that they cannot really generalize to new questions; if a question is not present during training, these methods can only predict

Table 7.4 : OKT's generalization performance to new questions that are unseen during training, using knowledge states from the previous time step, neighboring time steps, and random values.

Method	**CodeBLEU** $\uparrow$	**Dist-1** $\uparrow$
Previous	0.484	0.431
Average	0.504	0.419
Random	0.328	0.452

a student's probability of responding to it correctly using its concept labels (which are often unavailable). On the contrary, OKT's KR and RG components utilize exact question and response content, enabling it to generalize to new questions and predict exact responses to these questions and specific errors. We conduct a preliminary experiment to demonstrate this advantage of OKT: we first remove one question from the dataset (say it occurs at time step t for a student) and then predict the response to this question using the estimated knowledge state $\mathbf{h}_t$. We explore two ways to estimate h_t: i) averaging the knowledge states from neighboring time steps, i.e., $\mathbf{h}_{t-1}$ and $\mathbf{h}_{t+1}$, and ii) using the knowledge state from the previous time step, i.e., $\mathbf{h}_{t-1}$. As a baseline, we also use randomly generated knowledge state vectors to predict the response.

Table 7.4 shows the average results over removing each question, using DKT on first submissions. We use a smaller amount of epochs for this experiment (10 compared to 25 epochs from Table 7.1), which explains some of the significant drop in CodeBLEU scores. Nevertheless, OKT still significantly outperforms the baseline approach with no KE component, with averaging knowledge states from neighboring time steps slightly outperforming using the previous time step. Figure 7.4 visualizes predicted code vs. actual student code embeddings for an unseen question with an average CodeBLEU value of 0.538 over all students. Blue dots correspond to actual student responses and green dots represent RG predicted responses in 2-D, while red dots correspond to pairs of predicted and actual

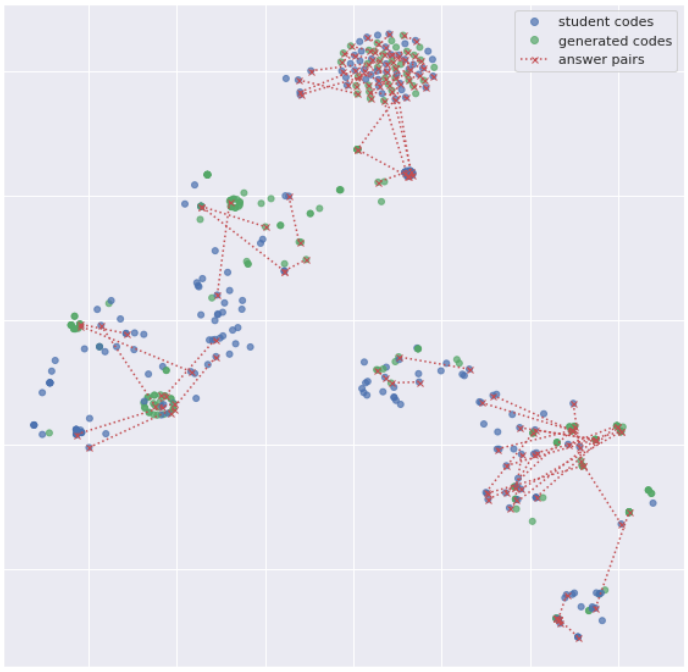


Figure 7.4 : Visualization of actual student code (blue) compared to predicted code (green) for a new question unseen during training. Code pairs that are close in the code embedding space are connected (red).

code that are highly similar (76 out of 225). We clearly see that OKT is able to capture the majority of student code variations on this new question from their responses to other questions and left no parts of the code embedding space unaccounted for. OKT's capability of generalizing to new questions can potentially be used to provide feedback to teachers plan homeworks, by predicting typical errors in programming questions that students in their class may exhibit, before assigning them.

7.4 Related Work

Knowledge Tracing. Existing methods for binary-valued KT can be broadly grouped by how they represent the student knowledge level variable, $\mathbf{h}$, in Eq. 7.1. For example, classic Bayesian knowledge tracing methods [153, 247, 391] treat student knowledge as a binary-valued latent variable. The KE and RP components are noisy binary channels, resulting in excellent interpretability of the model parameters. Factor analysis-based methods [37, 54, 253] use features and latent ability parameters to model student knowledge. The RP component in these methods relies on item response theory models [330]. More recently, deep learning-based KT methods [92, 244, 258, 300, 399] treat student knowledge as hidden states in neural networks. The KE component often relies on variants of recurrent neural networks [117], resulting in models that excel at future performance prediction but have limited interpretability.

Student responses, i.e., $\mathbf{x}$ in Eq. 7.1, are almost always treated as a binary-valued scalar indicating response correctness. Few methods characterize them as non-binary-valued such as option tracing [93], which analyzes the exact option students select on each multiple-choice question, and predict partial analysis [357]. Questions, i.e., $\mathbf{q}$ in Eq. 7.1, are often one-hot encoded, either according to question IDs/concept tags, or in a few cases, represented with graph neural networks using question-concept dependencies [385]. Few existing works use exact question content for $\mathbf{q}$. For example, [197, 352] use pre-trained word embeddings such as word2vec [226] to encode questions in the RP component. Specifically for programming questions, [351, 213, 412] use code representation techniques such as ASTNN [398] and code2vec [5] to convert student code into vectors and use them as input to the KE component.

Program Synthesis and Computer Science Education. Program synthesis from natural language instructions [70] has attracted significant recent interest since pre-trained language models [43] or language model architectures [191] have demonstrated their effectiveness on hard tasks such as solving coding challenge problems [112]. These methods are pre-trained on large datasets containing publicly available code on the internet, which is primarily written by skilled programmers. There is a line of existing work on analyzing student-generated code, most noticeably using the Hour of Code dataset released by Code.org [259, 305, 351], for tasks such as error analysis and automated feedback generation that are meaningful in computer science education settings.

7.5 Discussions

Limitations. Being the first attempt at the task of predicting the exact content of open-ended student responses, OKT has several obvious limitations. First, the ability to predict variation in student responses depends on the fine-tuned language model's ability to generate correct responses given the question statement. Therefore, it is not clear whether OKT can generalize to domains where language models have not been shown to be highly accurate at generating correct open-ended responses. Second, OKT requires a large amount of student coding data, which may limit its applicability to learning platforms in their early stages that do not have a large number of student users. Third, the open-ended response generation process is sequential and can be time-consuming, which may limit OKT's ability to support instructors and students in real-time in real-world computer science education scenarios.

Ethics statement. Our work should be seen as exploratory rather than a finished tool that can readily be deployed in real-world computer science educational scenarios. Since OKT requires training on a large amount of student-generated code, there is a need to systematically study any potential negative biases toward underrepresented student populations. The effectiveness of exact open-ended response prediction in helping instructors adjust their instruction and benefit students remains to be seen, which requires principled evaluation using A/B testing.

7.6 Conclusions and Future Work

In this work, we have proposed a framework for open-ended knowledge tracing (OKT) to track student knowledge acquisition while predicting their full responses to open-ended questions. We have demonstrated how OKT can be applied to the computer science education domain, where we analyze students' code submissions to programming questions. We addressed the key technical challenge of integrating student knowledge representations into code generation methods, e.g., text-to-code models based on fine-tuning GPT-2. Our experiments on real-world computer science student data indicate that OKT has considerable promise for tracking and predicting student mastery and performance.

There are many avenues for future work. First, we can use code standardization techniques [279] to further pre-process student code using semantic equivalence. Second, we can explore the applicability of OKT to other domains such as mathematics, where many pre-trained models for mathematical problem solving have been developed [57, 113, 289] and explore whether students consistently exhibit certain errors [334]. Third, we can develop knowledge tracing models that capture more specific aspects of knowledge, i.e., debugging skills, which is reflected in the *change* in student code across submissions to the same question after receiving automated feedback generated by the compiler or test cases. Fourth, we can further enhance the validity and interpretability of OKT by adding more human supervision, such as adding an additional loss on the test case scores of generated code. We can also use instructor- or expert-provided labels on student errors to make the latent knowledge state space more informative. Finally, we can further evaluate our framework on tasks relevant to instructor feedback, including compilation/runtime error category prediction and test case outcome prediction; see Section B.5 in the Appendix for a detailed discussion.

Chapter 8

VarFA: A Variational Factor Analysis Framework For Efficient Bayesian Learning Analytics

8.1 Introduction

A core task for many practical educational systems is *student modeling*, i.e., estimating students' mastery level on a set of skills or knowledge components (KC) [335, 56]. Such estimates allow in-depth understanding of students' learning status and form the foundation for automatic, intelligent learning interventions. For example, intelligent tutoring systems (ITSs) [302, 99, 8, 240, 109, 269] rely on knowing the students' skill levels in order to effectively recommend individualized learning curriculum and improve students' learning outcomes. In the big data era, student modeling is usually formulated as an educational data mining (EDM) problem [281, 15, 223] where an underlying machine learning (ML) model estimates students' skill mastery levels from students' learning records, i.e., their answers to assessment questions.

Many student modeling methods have been proposed in prior literature. A fruitful line of research for student modeling follows the *factor analysis (FA)* approach. FA models usually assume that an unknown, potentially multi-dimensional student parameter, in which each dimension is associated with a certain skill, explains how a student answers questions and is to be estimated. Popular and successful FA models include item response theory (IRT) [330], multi-dimensional IRT [3], learning factor analysis (LFA) [37], performance factor analysis (PFA) [253], DASH (short for difficulty, ability, and student history) [195, 232], DAS3H (short for difficulty, ability, skill and student skill history) [54], knowledge tracing machines (KTM) [341], and so on. Recently, more complex student models based on deep neural networks (DNN) have also been proposed [258, 399, 227]. Nevertheless, thanks to their simplicity, effectiveness and robustness, FA models remain widely adopted and investigated for practical EDM tasks. Moreover, there is evidence that simple FA models could even outperform DNN models for student modeling in terms of predicting students' answers [369]. Because of FA models' competitive performance and its elegant mathematical form, we focus on FA-based student models in this work.

Most of the aforementioned FA models compute a single point estimate of skill levels for each student. Often, however, it is not enough to obtain mere point estimates of students' skill levels; knowing the model's uncertainty in its estimation is crucial because it potentially helps improve the model's performance and improve both students' and instructors' experience with educational systems. For example, in ITS, an underlying model can use the uncertainty information in its estimation of students' skill level to automatically decide that its recommendations based on highly uncertain estimations are unreliable and instead notify a human instructor to evaluate the students' performance. This enables collaboration between ITS and instructors to create a more effective learning environment. In adaptive testing systems [39, 384], knowing the uncertainty in model's estimation could help the model intelligently pick the next test items to most effectively reduce its uncertainty about estimated students' skill levels. This will help to potentially reduce the number of items needed to have a confident, accurate estimation of the students' skill mastery level, saving time for both students to take the test and instructors to have a good assessment of the student's skills.

All the above applications require the model to "know what it does not know," i.e., to quantify the uncertainty of its estimation. Achieving this does not necessary changes the model. rather, we need a different inference algorithm for inferring not only a point estimate of student's skill level from observed data (i.e., students' answer records to questions) but also uncertainty in the estimations.

Fortunately, there exist methods that both compute point estimates of students' skill levels and quantifies the uncertainty of those estimates. These methods usually follow the Bayesian inference paradigm, where each student's unknown skill levels are treated as random variables drawn from a *posterior distribution*. Thus, one can use the *credible interval* to quantify the model's uncertainty on the student's estimated skill level. A classical method for Bayesian inference is Monte Carlo Markov Chain (MCMC) sampling [89] which has been widely used in many other disciplines [94] other than EDM. In the context of student modeling with FA models, existing works such as [175, 81, 248, 224] have applied MCMC methods to obtain credible intervals.

Unfortunately, classic Bayesian inference methods suffer from extensive computational complexity. For example, each computation step in MCMC involves a time-consuming evaluation of the posterior distribution. Making matters worse, MCMC typically takes many more steps to converge than non-Bayesian inference methods. As a concrete illustration, in [175], the Bayesian inference method (10 minutes) is about 100 times slower than the other non-Bayesian inference method based on stochastic gradient descent (6 seconds). The high computational cost prevents Bayesian inference methods from mass-deployment in large-scale, real-time educational systems where timely feedback is critical for learning [74] and tens of thousands of data points need to be processed in seconds or less instead of minutes or hours. It is thus highly desirable to accelerate Bayesian inference so that one can quantify model's uncertainty in its estimation as efficiently as non-Bayesian inference methods.

Contributions. In this work, we propose VarFA, a novel framework based on *variational inference* (VI) to perform efficient, scalable Bayesian inference for FA models. The key idea is to approximate the true posterior distribution, whose costly computation slows down Bayesian inference, with a *variational distribution*. We will see in Section 8.3 that, with this approximation, we turn Bayesian inference into an optimization problem where we can use the same efficient inference algorithms as in non-Bayesian inference methods. Moreover, the variational distribution is very flexible and we have full control specifying it, allowing us to freely use the latest development in machine learning, e.g., deep neural networks (DNNs), to design the variational distribution that closely approximates the true posterior. Thus, we also regard our work as a first step in applying DNNs to FA models for student modeling, achieving efficient Bayesian inference (enabled by DNNs) without losing interpretability (brough by FA models). We demonstrate the efficacy of our framework on both synthetic and real data sets, showcasing that VarFA substantially accelerates classic Bayesian inference for FA models with no compromise on performance.

The remainder of this chapter is organized as follows. Section 8.2 introduces the problem setup in FA and reviews the important FA models and their inference methods, in particular Bayesian inference. Section 8.3 explains our VarFA framework in detail. Section 8.4 presents extensive experimental results that substantiate the claimed advantages of our framework. Section 8.5 concludes this chapter and discusses possible extensions to VarFA.

8.2 Background and Related Work

We first set up the problem and review related work. Assume we have a data set $\boldsymbol{Y} \in \mathbb{R}^{N \times Q}$ organized in matrix format where N is the total number of students and Q is the number of questions. This is a binary students' answer record matrix where each entry y_{ij} represents whether student i correctly answered question j. Usually, not all students answer all questions. Thus, $\boldsymbol{Y}$ contains missing values. We use $\{i, j\} \in \Omega_{\text{obs}}$ to denote entries in $\boldsymbol{Y}$, i.e., the i-th student's answer record to the j-th question, that are observed.

We are interested in models capable of inferring each i-th student's skill mastery level that can accurately predict the student's answers given the above data. These models are often evaluated on the prediction accuracy and whether the inferred student skill mastery levels are easily interpretable and educationally meaningful. We now review factor analysis models (FA), one of the most widely adopted and successful methodologies for the student modeling task.

8.2.1 Factor Analysis For Student Modeling

One of the earliest FA model for student modeling is based on the item response theory (IRT) [330]. It usually has the following form

$$\mathbb{P}(y_{ij} = 1) = \sigma(c_i + \mu_j)\,, \tag{8.1}$$

which assumes that each student's answer y_{ij} is independently Bernoulli distributed. The above formula says that the students' answers can be explained by an unknown scalar student skill level factor c_i for each student i and an unknown scalar question difficulty level factor μ_j for each question j. $\sigma(\cdot)$ is a Sigmoid activation function, i.e., $\sigma(x) = 1/(1+\exp(-x))$. The multi-dimensional IRT model (MIRT) [3] extends IRT by using a multi-dimensional vector to represent the student skill levels. More recently, [175] proposed sparse factor analysis model (SPARFA) which extends MIRT by imposing additional assumptions, resulting in improved interpretations of the inferred factors.

Other FA models seek to improve student modeling performance by cleverly incorporate additional auxiliary information. For example, the additive factor model (AFM) [37] incorporates students' accumulative correct answers for a question and the skill tags associated with each question:

$$\mathbb{P}(y_{ij} = 1) = \sigma \left(c_i + \sum_{k=1}^{\kappa} \left(q_{kj}\beta_k + q_{kj}\rho_k b_{ik} \right) \right) , \tag{8.2}$$

where κ is the total number of skills in the data, b_{ik} is the i-th student's total number of correct answers for skill k and $q_{kj} \in \{0, 1\}$ indicates whether the skill k is associated with question j. In particular, q_{kj}'s form matrix $\boldsymbol{Q} \in \mathbb{R}^{K \times Q}$ which is commonly known as the Q-matrix in literature [322]. The unknown, to-be-inferred factors are β_k, a scalar difficulty factor for each skill k, and ρ_k, a scalar learning rate of skill k. The performance factor model (PFM) [253] builds upon AFM that additionally incorporate the total number of a student's incorrect answers to questions associated with a skill. The instructor factor model (IFM) [51] further builds on PFM to incorporate the prior knowledge of whether a student has already mastered a skill. More recently, [341] introduces knowledge tracing machines (KTM) that could flexibly incorporate a number of auxiliary information mentioned above, thus generalizing AFM, PFM and IFM.

Another way to improve student modeling performance is to utilize the so-called *memory*, i.e., using students' historic action data over time. For example, [195, 232] proposed DASH (short for d̲ifficulty, a̲bility, and s̲tudent h̲istory) that incorporates a student's total number of correct answers and total number of attempts for a question in a given time window

$$\begin{aligned}\mathbb{P}(y_{ij} = 1) = \sigma \Bigg(c_i - \mu_j + \sum_{w=0}^{W-1} \Big(& \theta_{2w+1} \log(1 + \rho_{ijw}) \\ & - \theta_{2w+2} \log(1 + \gamma_{ijw}) \Big) \Bigg)\end{aligned} \tag{8.3}$$

where W is the length of the time windows (e.g., number of days), ρ_{ijw} and γ_{ijw} are the

i-th student's total number of correct answers and total number of attempts for question j at time w, respectively. θ's are parameters that captures the effect of correct answers and total attempts. More recently, DAS3H [54] extends DASH to further include skill tags associated with each question which essentially amounts to adding, within the last summation term in Eq. 8.3, another summation over the number of skills.

A general formulation for FA models

Although the above FA models differ in their formulae, modeling assumptions and the available auxiliary data used, we argue that the aforementioned FA models can be unified into a canonical formulation below

$$\mathbb{P}(y_{ij} = 1) = \sigma(\mathbf{c}_i^\top \mathbf{m}_j + \mu_j)\,, \tag{8.4}$$

where $\mathbf{c}_i \in \mathbb{R}^K$, $\mathbf{m}_j \in \mathbb{R}^K$ and $\mu_j \in \mathbb{R}$ are factors whose dimension, interpretations and subscript indices depend on the specific instantiations of the FA model. To illustrate that Eq. 8.4 subsumes the FA models mentioned above, we demonstrate, as examples, how to turn SPARFA and AFM into the form in Eq. 8.4 and how to reinterpret the parameters in the reformulation. The equivalence between Eq. 8.4 and the original SPARFA formula is immediate and we can interpret the parameters in Eq. 8.4 as follows to recover SPARFA: K is the number of *latent skills* that group the actual skill tags in the data set into meaningful coarse clusters; $\mathbf{c}_i$ represents the i-th student's skill level on the latent skills; $\mathbf{m}_j$ represents the strength of association of the j-th question with the latent skills which is assumed to be nonnegative and sparse for improved interpretation; and μ_j represents the difficulty of question j. All three factors in SPARFA are assumed to be unknown.

Similarly, for AFM, we can perform the following change of variables and indices to obtain Eq. 8.4. We first change the indices $\mathbf{m}_j$ to $\mathbf{m}_{ij}$ and μ_j to μ_i, and remove the index in $\mathbf{c}_i$ to $\mathbf{c}$. Then, we simply set $\mathbf{c} = [\beta_1, ..., \beta_\kappa, \rho_1, ..., \rho_\kappa]^\top$, $\mathbf{m}_{ij} = [q_{1j}, ..., q_{\kappa j}, q_{1j}b_{ik}, ..., q_{\kappa j}b_{i\kappa}]^\top$ and $\mu_i = \alpha_i$ to obtain Eq. 8.4. We can interpret the parameters in the reformulation as follows to recover AFM: $\kappa = K/2$ is the total number of skill tags in the data; $\mathbf{c}$ represents the unknown skill information including skill difficulty and learning rate; $\mathbf{m}_{ij}$ summarizes known auxiliary information for the i-th student and j-th question; μ_i represents the unknown i-th student's skill mastery level. Tthe other FA models can be cast into Eq. 8.4 in a similar fashion. Note that, if the observed data $\boldsymbol{Y}$ is not binary but rather continuous or categorical, we can simply use a Gaussian or categorical distribution for the observed data and change the Sigmoid activation $\sigma(\cdot)$ to some other activation functions accordingly. In this way, we still retain the general FA model in Eq. 8.4. We will use this canonical form to facilitate discussions in the rest of this chapter.

8.2.2 Inference Methods for FA Models

We first introduce the inference objective and briefly review maximum log likelihood estimation method. Then, we review existing works that uses Bayesian inference for FA and highlight their high computational complexity, paving the way to VarFA, our proposed efficient Bayesian inference framework based on variational inference.

The factors in Eq. 8.4 may contain known factors that need no inference. Therefore, for convenience of notation, let $[\nu, \theta, \psi]$ be a partition of the factors $\mathbf{c}_i$, $\mathbf{m}_j$ and μ_j where ν contains the unknown students' skill level factors to be estimated, θ contains the remaining unknown factors and ψ contains the known factors. For example, for AFM, $\nu = \{\mu_1, ..., \mu_N\}$, $\theta = \mathbf{c}$ and $\psi = \{\mathbf{m}_{11}, ..., \mathbf{m}_{NQ}\}$. For SPARFA, $\nu = \{\mathbf{c}_1, ..., \mathbf{c}_N\}$, $\theta = \{\mathbf{m}_1, ..., \mathbf{m}_Q, \mu_1, ..., \mu_Q\}$ and $\psi = \emptyset$. Further, let the subscripts for these partitions indicate the corresponding latent factors in FA; i.e., for SPARFA, $\theta_i = [\mathbf{m}_i, \mu_i]$ and $\nu_i = \mathbf{c}_i$.

Since ψ summarizes known factors, we will omit it from the mathematical expositions in the remainder of the chapter.

The inference objective in FA is then to obtain a good estimate of the unknown factors, represented by θ and ν, with respect to some loss function $\mathcal{L}$, usually the marginal data log likelihood. There are two ways to infer the unknown factors.

Maximum Likelihood Estimation

The first way is through the maximum likelihood estimate (MLE) which obtains a point estimate of the unknown factors

$$\widehat{\theta}, \widehat{\nu} = \underset{\theta,\nu}{\operatorname{argmin}} \left(- \sum_{i,j \in \Omega_{\text{obs}}} \log p(y_{ij}; \theta, \nu) \right) + \lambda \mathcal{R}(\theta, \nu) \, . \tag{8.5}$$

Recall that Ω_{obs} indicates which student i has provided an answer to question j. $\mathcal{R}(\theta, \nu)$ is a regularization term, e.g., $\ell 2$ regularization on the factors and λ is a hyper-parameter that controls the strength of the regularization. Most of the works in FA use this method of inference [37, 253, 51, 341, 54, 195, 232]. The advantage of MLE is that the above optimization objective allows the use of fast inference algorithms, in particular stochastic gradient descent (SGD) and its many variants, making the inference highly efficient.

Maximum A Posteriori Estimation

The second way is through maximum a posteriori (MAP) estimation through Bayesian inference. This is the focus of our work. Recall that we wish to obtain not only a point estimate but also credible interval, which MAP allows while MLE does not.

Bayesian inference methods treat the unknown parameters ν and θ as random variables drawn from some distribution. We are thus interested in inferring the *posterior* distribution of ν and θ. Using the Bayes rule, we have that

$$p(\nu|\boldsymbol{Y},\theta) = \frac{p(\boldsymbol{Y},\nu,\theta)}{p(\boldsymbol{Y})} \tag{8.6}$$

$$= \frac{p(\boldsymbol{Y}|\nu,\theta)\,p(\nu)\,p(\theta)}{\iint p(\boldsymbol{Y}|\nu,\theta)\,p(\nu)\,p(\theta)d\nu d\theta}. \tag{8.7}$$

We can similarly derive the posterior for the other unknown factor θ, although in this work we focus on Bayesian inference for the unknown student skill level parameter ν. In Eq. 8.6 to Eq. 8.7 we have applied standard Bayes rule. Note that, because ψ contains known factors and thus does not enter the Bayes rule equation, we have omitted it from the above equations. Once we estimate the posterior distribution for ν from data, we can obtain both point estimates and credible intervals by respectively taking the mean and the standard deviation of the posterior distributions. A number of prior works have proposed to use Bayesian inference for factor analysis, mostly developed for the IRT model [82, 228]. More recently, Bayesian methods are developed for more complex models. For example, [175] proposed SPARFA-B, a specialized MCMC algorithm that accounts for SPARFA's sparsity and nonnegativity assumptions.

However, Bayesian inference suffers from high computational cost despite its desired capability to quantify uncertainty. The challenge stems from the difficulty to evaluate the denominator in the posterior distribution in Eq. 8.6 and 8.7 which involves an integration over a potentially multi-dimensional variable. This term usually cannot be computed in close form, thus we usually cannot get an analytical formula for the posterior distribution. Monte Carlo Markov Chain (MCMC) method gets around this difficulty by using analytically tractable proposal distributions to approximate the true posterior and sequentially updating the proposal distribution in a manner reminiscent of gradient descent. MCMC methods also enjoy the theoretical advantage that, when left running for long enough, the proposal distribution is guaranteed to converge to the true posterior distribution [89]. However, in practice, it might take very long time for MCMC to converge to the point that running MCMC is no longer practical when data set is very large. The high computational complexity of MCMC is apparently impractical for educational applications where timely feedback is of critical importance [74].

8.3 VarFA: A Variational Inference Factor Analysis Framework

We are ready to introduce VarFA, our variational inference (VI) factor analysis framework for efficient Bayesian learning analytics. The core idea follows the variational principle, i.e., we use a parametric variational distribution to approximate the true posterior distribution. VarFA is highly flexible and efficient, making it suitable for large scale Bayesian inference for FA models in the context of educational data mining.

In this current work, we focus on obtaining credible interval for the student skill mastery factor ν as a first step of VarFA because, recall from Section 8.1, this factor is often of more practical interest than the other unknown factors. Therefore, currently VarFA is a hybrid MAP and MLE inference method: we perform VI on the unknown student factor ν and MLE on all other unknown factors θ. We consider this hybrid nature of VarFA in its current formulation as a novel feature because classic MLE or MAP estimation are not capable of performing Bayesian inference on a subset of the unknown factors. Extension to VarFA to full Bayesian inference for all unknown factors is part of an ongoing research; see 8.5 for more discussions.

8.3.1 VarFA Details

Now, we explain in detail how to apply variational inference for FA models for efficient Bayesian inference. Because the posterior distribution is intractable to compute (recall

Eq. 8.6 and 8.7 and the related discussion), we approximate the true posterior distribution for ν with a parametric variational distribution

$$p(\nu|\boldsymbol{Y}, \theta) \approx q_\phi(\nu|\boldsymbol{Y}) = \prod_{i=1}^{N} q_\phi(\nu_i|\boldsymbol{y}_i)\,, \tag{8.8}$$

where ϕ is a collection of learnable parameters that parametrize the variational distribution and $\boldsymbol{y}_i$ is all the answer records by student i. Notably, we have removed the dependency of the variational distribution on ψ and θ so that the variational distribution is solely controlled by the variational parameter ϕ. Thus, the design of the variational distribution is highly flexible. All we need to do is to specify a class of distributions and design a function parametrized by ϕ to output the parameters of q_ϕ. Common in prior literature is to use a Gaussian with diagonal covariance for q_ϕ:

$$q_\phi(\nu|\boldsymbol{y}_i) = \mathcal{N}(\boldsymbol{u}_j, \mathrm{diag}(\boldsymbol{v}_j))\,, \tag{8.9}$$

where its mean and variance $[\boldsymbol{u}_j^\top, \boldsymbol{v}_j^\top]^\top = f_\phi(\boldsymbol{y}_i)$. We can use arbitrarily complex functions such as a deep neural network for f_ϕ as long as they are differentiable; more details in Section 8.3.2. With the above approximation, Bayesian inference turns into an optimization problem under the variational principle, where we now optimize a lower bound, known as the evidence lower bound (ELBO) [21], of the marginal data log likelihood. We derive a novel ELBO objective for variational inference applied to FA models in the EDM setting, summarized in the proposition below.

Proposition 1

In VarFA, the ELBO objective for an FA model in the form of Eq. 8.4 is

$$\begin{aligned}\mathcal{L}_{\text{ELBO}}(\phi,\theta) = \sum_{i,j\in\Omega_{\text{obs}}} \Big(&- D_{\text{KL}}[q_\phi(\nu_i|\boldsymbol{y}_i)\|p(\nu_i)] \\ &+ \mathbb{E}_{\nu_i\sim q_\phi(\nu_i|\boldsymbol{y}_i)}[\log p_{\theta_j}(y_{ij}|\nu_i)]\Big)\end{aligned} \tag{8.10}$$

where D_{KL} *is the Kullback–Leibler (KL) divergence [208] between two distributions.*

Proof 1 We start with the marginal data log likelihood which is the objective we want to maximize and introduce the random variables ν_i*'s:*

$$\begin{aligned}\log p(\boldsymbol{Y}) &= \sum_{i,j\in\Omega_{\text{obs}}} \log p_{\theta_j}(y_{ij}) \\ &= \sum_{i,j\in\Omega_{\text{obs}}} \log \int p_{\theta_j}(y_{ij},\nu_j)\, d\nu_i \\ &= \sum_{i,j\in\Omega_{\text{obs}}} \log \int q_\phi(\nu_i|\boldsymbol{y}_i)\frac{p_{\theta_j}(y_{ij},\nu_j)}{q_\phi(\nu_i|\boldsymbol{y}_i)}\, d\nu_i \\ &\overset{(i)}{\geq} \sum_{i,j\in\Omega_{\text{obs}}} \mathbb{E}_{\nu_i\sim q_\phi(\nu_i|\boldsymbol{y}_i)}\left[\log\frac{p_{\theta_j}(y_{ij},\nu_j)}{q_\phi(\nu_i|\boldsymbol{y}_i)}\right] \\ &\geq \sum_{i,j\in\Omega_{\text{obs}}} -\mathbb{E}_{\nu_i}\left[\log\frac{q_\phi(\nu_i|\boldsymbol{y}_i)}{p(\nu_j)}\right] + \mathbb{E}_{\nu_i}\left[\log p_{\theta_j}(y_{ij}|\nu_j)\right] \\ &= \mathcal{L}_{\text{ELBO}}(\phi,\theta)\end{aligned}$$

where step (i) follows from Jensen's inequality [133]. □

The ELBO objective differs from those used in existing literature in that, in our case, because the data matrix is only partially observed, the summation is only over $\{i,j\}\in\Omega_{\text{obs}}$, i.e., the observed entries in the data matrix. In contrast, in prior literature, the summation in the ELBO objective is over all all entries in the data matrix.

We form the following optimization objective to estimate ϕ and θ:

$$\widehat{\theta}, \widehat{\phi} = \underset{\theta,\phi}{\operatorname{argmin}} \; -\mathcal{L}_{\text{ELBO}}(\phi, \theta) + \lambda \mathcal{R}(\theta)\,, \tag{8.11}$$

where $\mathcal{R}(\theta)$ is a regularization term. That is, we perform VI on the student factor ν and MLE inference on the remaining factors θ. More explicitly, to perform VI on ν, we simply compute the mean and standard deviation of q_ϕ using f_ϕ with the learnt variational parameters ϕ.

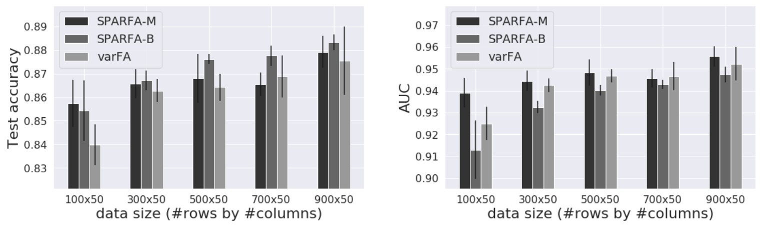
8.3.2 Why is VarFA efficient?

VarFA supports efficient stochastic optimization. By formulating Bayesian inference as an optimization problem via the ELBO objective, VarFA allows efficient optimization algorithms such as SGD, with a few additional tricks. In particular, we require all terms in $\mathcal{L}(\theta, \phi)$ to be differentiable with respect to θ and ϕ which enable gradient computation necessary for SGD. This requirement is easily satisfied with a few moderate assumptions. Specifically, we let the variational distribution q_ϕ and the prior distribution $p(\nu_i)$ for each i to be Gaussian (See Eq. 8.9; we use a standard Gaussian for the prior distribution $p(\nu_i)$, i.e., $p(\nu_i) = \mathcal{N}(0, \boldsymbol{I})$) following existing literature [157, 276]. These two assumptions are not particularly limiting especially given the vast number of successful applications of VI that rely on the same assumptions [405, 218, 303, 61, 44, 115, 154]. Thanks to the Gaussian assumption, for the first term in $\mathcal{L}(\theta, \phi)$, we can easily compute the KL divergence term analytically in closed form. For the second term in $\mathcal{L}(\theta, \phi)$, we can use the so-called reparametrization trick, i.e., $\nu_i = \boldsymbol{u}_i + \boldsymbol{v}_i \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, which allows a low variance estimation of the gradient of the second term in $\mathcal{L}(\theta, \phi)$ with respect to ϕ. See Section 2.3 in [157], Section 2.3 in [158] and Section 3 in [276] for more details on stochastic gradient computation in VI. As a result, our framework can be efficiently implemented using a number of open source, automatic differentiation packages such as Tensorflow [1] and PyTorch [252].

VarFA supports amortized inference. Classic Bayesian inference methods such as MCMC infer each parameter ν_i for each student. As the number of students increases, the number of parameters that MCMC needs to infer also increases, which may not be scalable in large-scale data settings. In contrast, unlike classic Bayesian inference methods, VI is *amortized*. It does *not* infer each parameter ν_i. Rather, it estimates a *single* set of variational parameter ϕ responsible for inferring *all* ν_i's. In this way, once we have trained FA models using VarFA and obtain the variational parameter ϕ, we can easily infer ν_i by simply computing q_ϕ, even for new students, without invoking any additional optimization or inference procedures.

8.3.3 Dealing with missing entries

Note from Eq. 8.10 that the variational distribution q_ϕ for each ν_i is conditioned on $\boldsymbol{y}_i \in \mathbb{R}^Q$, i.e., an entire row in $\boldsymbol{Y}$. However, in practice, the data matrix $\boldsymbol{Y}$ is often only partially observed, i.e., some students only answer a subset of questions. Then, $\boldsymbol{y}_i$'s will likely contain missing values which computation cannot be performed on. To work around this issue, we use "zero imputation", a simple strategy that transforms the missing values to 0, following prior work on applying VI in the context of recommender systems [192, 284, 291, 236] that demonstrated the effectiveness of this strategy despite its simplicity. We note that there exist more elaborate ways to deal with missing entries, such as designing specialized function f_ϕ for the variational distribution [205, 97, 206]. We leave the investigation of more effectively dealing with missing entries to future work.



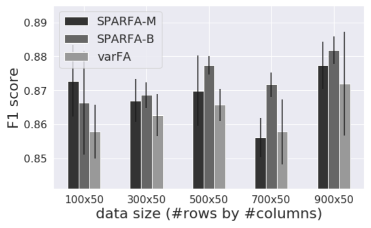


Figure 8.1 : Performance of the SPARFA-M, SPARFA-B, and VarFA algorithms on the synthetic data set with different data sizes. Plots from left to right show comparison on accuracy (ACC), area under curve (AUC) and F1 metrics, respectively. Higher is better for all metrics. VarFA performs similarly to SPARFA-M and SPARFA-B.

8.3.4 Remarks

Applicability of VarFA. The VarFA framework is general and flexible and can be applied to a wide array of FA models. The recipe for applying VarFA to an FA model of choice is as follows: 1) formulate the FA model into the canonical formulation as in Eq. 8.4; 2) partition the factors into student factor ν, the remaining unknown factors θ and known factors ψ; 3) perform VI on ν and MLE estimation on θ, following Section. 8.3.1.

Relation to variational auto-encoders. Our proposed framework can be regarded as a standard variational auto-encoder (VAE) but with the decoder implemented not as a neural network but by the FA model. Because the decoder is constraint to a FA model, it is more interpretable than a neural network.

Relation to other efficient Bayesian factor analysis methods. We acknowledge that VI applied to general FA models have been proposed in existing literature [91, 404]. However, to our knowledge, little prior work have applied VI to educational FA models nor investigated its effectiveness. One concurrent work applied VI to IRT [373]. Our VarFA framework applies generally to a number of other FA models by following the recipe in the preceding paragraph, including IRT. Thus, our work complements existing literature in providing promising results in applying VI for FA models in the context of educational data mining.

8.4 Experiments

We demonstrate the efficacy of VarFA variational inference framework using the sparse factor analysis model (SPARFA) as the underlying FA model. This choice of SPARFA as the FA model to investigate is motivated by its mathematical generality: it assumes no auxiliary information is available and all three factors $\mathbf{c}_i$'s, $\mathbf{m}_j$'s and μ_j's need to be estimated, which makes the inference problem more challenging. [175] provides two inference algorithms including SPARFA-M (based on MLE) and SPARFA-B (based on MAP).

We conduct experiments on both synthetic and real data sets. Using synthetic data sets, we compare VarFA to both SPARFA-M and SPARFA-B. We demonstrate that 1) VarFA predicts students' answers as accurately as SPARFA-M and SPARFA-B; 2) VarFA is almost $100\times$ faster than SPARFA-B. Using real data sets, we compare VarFA to SPARFA-M. We demonstrate that 1) VarFA predicts students' answers more accurately than SPARFA-M; 2) VarFA can output the same insights as SPARFA-M, including point estimate of students' skill levels and questions' associations with skill tags; 3) VarFA can additionally output meaningful uncertainty quantification for student skill levels, which SPARFA-M is incapable of, without sacrifice to computational efficiency. Note that SPARFA-B can also compute uncertainty for small data sets but fails for large data sets due to scalability issues and thus we do not compare to SPARFA-B for real data sets.

Specifically for SPARFA, using the notation convention in Section 8.2.2, the question-skill association factors $\mathbf{m}_j$'s and the question difficulty factors μ_j are collected in $\theta = \{\mathbf{m}_1, ..., \mathbf{m}_Q, \mu_1, ..., \mu_Q\}$. The student skill level factors $\mathbf{c}_j$'s are collected in $\nu = \{\mathbf{c}_1, ..., \mathbf{c}_N\}$. Because the factors $\mathbf{m}_j$'s are unknown, the "skills" in SPARFA are *latent* (referred to as "latent skills" in subsequent discussions) and are not attached to any

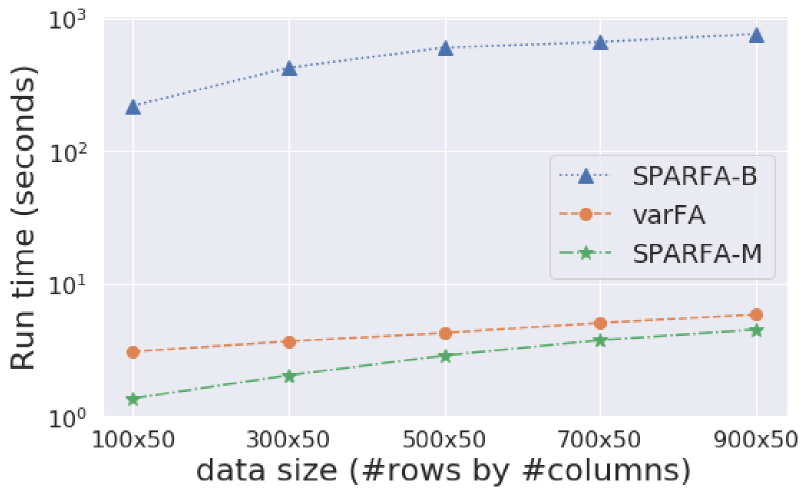


Figure 8.2 : Training run time comparing VarFA, SPARFA-M, and SPARFA-B on synthetic data sets of varying sizes. VarFA performs approximate Bayesian inference almost 100x faster than SPARFA-B and is close to the run time of SPARFA-M. Thus, VarFA enables practical and scalable Bayesian inference for very large data sets.

specific interpretation. However, as we will see in Section 8.4.2, assuming the skill tags are available as auxiliary information, we can associate the estimated latent skills with the provided skills tags using the same approach proposed in [175]. For the regularization term in Eq. 8.11, because the factors $\mathbf{m}_j$'s are assumed to be sparse and nonnegative, we use $\ell 1$ regularization for $\mathbf{m}_j$'s. When performing MLE for $\mathbf{m}_j$'s, we use the proximal gradient algorithm for optimization problem with nonnegative and sparsity requirements; see Section 3.2 in [175] for more details. For the other unknown factor μ_j's in θ, we apply standard $\ell 2$ regularization.

8.4.1 Synthetic Data Experiments

Setup

Data set generation. We generate data set according to the canonical FA model specified in Eq. 8.4, taking into consideration the additional sparsity and nonnegativity assumptions

Table 8.1 : Summary statistics of pre-processed real data sets.

data set	**#students**	**#questions**	**%observed**
assistment	392	747	12.93%
algebra	697	782	13.02%
bridge	913	1242	11.42%

in SPARFA. Specifically, We sample the factors μ_j's and $\mathbf{c}_i$'s from i.i.d. standard isotropic Gaussian distributions with variance 1 (or identity matrix for $\mathbf{c}_i$'s) and mean sampled from a uniform distribution in range $[-1, 1]$. To simulate sparse and nonnegative factors $\mathbf{m}_j$'s, we first sample an auxiliary variable $s_{kj} \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(\pi)$, where π controls the sparsity, and then sample $m_{kj} \overset{\text{i.i.d.}}{\sim} \text{Exponential}(1)$ when $s_{kj} = 1$ or setting $m_{kj} = 0$ when $s_{kj} = 0$ for each j and k. In all synthetic data experiments, we use $\pi = 0.3$ and set the true number of latent skills to $K = 5$.

Experimental settings. We vary the size of the data set and use 5 different data matrix sizes: 100×50, 300×50, 500×50, 700×50 and 900×50. We select a data missing rate of 50%, i.e., we randomly choose 50% of all entries in the data matrix as training set and the rest as test set. Experimental results for each of the above data sizes are averaged over 5 runs where we randomize over the train/test data split. We train SPARFA with both SPARFA-M and VarFA using the Adam optimizer [156] with learning rate $= 0.05$ for 100 epochs. Regularization hyper-parameters are chosen by grid search for each experiment. For VarFA, we use a simple 3-layer neural network for the function f_ϕ in the variational distribution. We use the hyper-parameter settings for SPARFA-B following Section 4.2 in [175].

Evaluation metrics. We evaluate the inference algorithms on their ability to recover (predict) the missing entries in the data matrix given the observed entries. We term this criterion "student answer prediction". Since our data matrix is binary, using prediction accuracy (ACC) alone does not accurately reflect the performance of the inference algorithms under comparison. Thus, we use area under the receiver operating characteristic curve (AUC) and F1 score in addition to ACC, as standard in evaluating binary predictions [107].

Results

Fig. 8.1 shows bar plots that compare the student answer prediction performance of VarFA with SPARFA-M and SPARFA-B on all three evaluation metrics. We can observe that all three methods perform similarly and that SPARFA-M and SPARFA-B show no statistically significant advantage over VarFA. We further showcase VarFA's scalability by comparing its training run-time with SPARFA-M and SPARFA-B. The results are shown in Fig. 8.2. VarFA is significantly faster than SPARFA-B and is almost as fast as SPARFA-M.

In summary, with VarFA, we obtain posteriors that allow uncertainty quantification with roughly the *same* computation complexity of computing point estimates and *without* compromising prediction performance.

8.4.2 Real Data Experiments

Setup

Data sets and pre-processing steps. We perform experiments on three large-scale, publicly available, real educational data sets including ASSISTments 2009-2010 (Assistment) [109], Algebra I 2006-2007 (algebra) [310] and Bridge to Algebra 2006-2007 (bridge) [311]. The details of the data sets, including data format and data collection procedure can be found in the preceding references. We remove students and questions that have too few answer records from the data sets to reduce the sparsity of the data sets. Specifically, we keep students and questions that have no less than 30, 35 and 40 answer records for Assistment, Algebra and Bridge data sets, respectively. In each data set, each student may provide more than one answer record for each question. Therefore, we also remove student-question answer records except for the first one. Table 8.1 presents the summary statistics of the resulting pre-processed data matrix for each data set.

Experimental settings and evaluation metrics. Optimizer, learning rate, number of epochs, neural network architecture and evaluation metrics are the same as in synthetic data experiments. For real data sets, we use 8 latent skills instead of 5. Other choices of the number of latent skills might result in better performance. However, since we are comparing different inference algorithms for the *same* model, it is a fair to compare the inference algorithms on the same model with the same number of latent dimensions. We use a 80:20 data split, i.e., we randomly sample 80% and 20% of the observed entries in each data set, without replacement, as training and test sets, respectively. Regularization hyper-parameters are selected by grid search and are different for each data set. We only compare VarFA with SPARFA-M because SPARFA-B does not scale to such large data sets.

Results: Performance Comparison

Table 8.2 shows the average performance on the test set of each data set comparing VarFA and SPARFA-M for all three data sets and additionally run time. We can see that VarFA achieves slightly better student answer prediction on most data sets and on most metrics. Interestingly, recall that, in the preceding synthetic data set experiment, SPARFA-M performed better than VarFA most of the time. A possible explanation is that, for synthetic data sets, the underlying data generation process matches the SPARFA model, whereas for real data

Table 8.2 : Student answer prediction erformance comapring VarFA to SPARFA-M on Assistment, Algebra and Bridge data sets. $\uparrow$ and $\downarrow$ denote higher and lower is better, respectively. VarFA performs better than SPARFA-M on all three data sets and evaluation metrics most of the time. Additionally, VarFA's run time is very close to SPARFA-M.

(a) Assistment

Metric	**Algorithm**	
	SPARFA-M	VarFA
ACC $\uparrow$	0.7074±0.0044	**0.7101**±0.0048
AUC $\uparrow$	0.756±0.048	**0.7635**±0.0036
F1 $\uparrow$	0.7746±0.0029	**0.7765**±0.0014
Run time (s) $\downarrow$	**5.3319**±0.2774	6.9167±0.1074

(b) Algebra

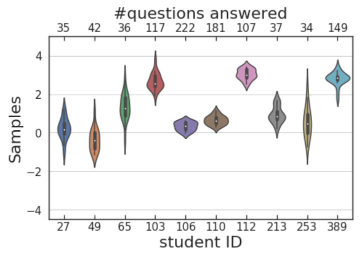
Metric	**Algorithm**	
	SPARFA-M	VarFA
ACC $\uparrow$	0.7735±0.0037	**0.7774**±0.0031
AUC $\uparrow$	0.8137±0.003	**0.8245**±0.002
F1 $\uparrow$	0.8465±0.0021	**0.8486**±0.001
Run time (s) $\downarrow$	**8.464**±0.4568	10.3335±0.4435

(c) Bridge

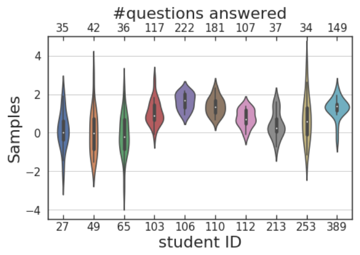
Metric	**Algorithm**	
	SPARFA-M	VarFA
ACC $\uparrow$	**0.8492**±0.0016	0.8468±0.0016
AUC $\uparrow$	0.837±0.0024	**0.8419**±0.0028
F1 $\uparrow$	**0.9121**±0.0005	0.912±0.0009
Run time (s) $\downarrow$	**15.6048**±0.7314	15.8558±1.046

sets, the underlying data generation process is unknown. VarFA's better performance than SPARFA-M for real data sets implies that VarFA is more robust to the unknown underlying data generation process. As a result, VarFA may be more applicable than SPARFA-M in real-world situations.

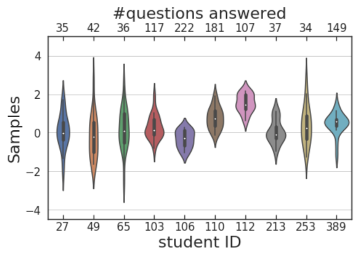
Table 8.2 also shows the run time comparison between VarFA and SPARFA-M; see the last row in each sub-table. We see that both inference algorithms have very similar run time,



(a) 3rd latent concept



(b) 4th latent concept



(c) 7th latent concept

Figure 8.3 : Violin plot showing the mean and standard deviation of the estimated skill mastery levels on 10 selected students on the 3rd, 4th and 7th latent skills that VarFA computes. In each sub-figure, bottom and top axises respectively shows student IDs and top axis shows the number of questions each student answered. The more questions a student answers, the tighter the credible interval. (Best viewed in color.)

showing that VarFA is applicable for very large data sets. Notably, VarFA achieves this efficiency while also performing Bayesian inference on the student knowledge level factor.

Results: Bayesian Inference With VarFA

We now illustrate VarFA's capability of outputting credible intervals using the Assistment data set. Fig. 8.3 presents violin plots that show the sampled student latent skill levels for a random subset of 10 students. Plots 8.3a, 8.3b and 8.3c shows the inferred students ability for the 3rd, 4th and 7th latent skill dimension. In each plot, the bottom axis shows the student ID and the top axis shows the total number of questions answered by the corresponding student. For each student, the horizontal width of the violin represents the density of the samples; the skinnier the violin, the more widespread the samples are, implying the model's less certainty on its estimations.

Results in Fig. 8.3 confirms our intuition that the more questions a student answers, the more certain the model is about its estimation. For example, students with ID 106, 110 and 389 answered 222, 181 and 149 questions, respectively, and the credible intervals of their

Table 8.3 : Illustration of the estimated latent skills with the their top 3 most strongly associated skill tags in the Assistment data set. The percentage in the parenthesis shows the association probability (summed to 1 for each latent skill). We see that the tagged skills associated with each estimated latent skill form intuitive and interpretable groups.

Latent Skill 1	**Latent Skill 3**
Division Fractions (29.1%)	Conversion of Fraction Decimals Percents (7.3%)
Least Common Multiple (18.1%)	Addition and Subtraction Positive Decimals (6.8%)
Write Linear Equation from Ordered Pairs (17.8%)	Probability of a Single Event (5.7%)
Latent Skill 4	**Latent Skill 7**
Pattern Finding (17.4%)	Volume Sphere (13.4%)
Histogram as Table or Graph (11.3%)	Volume Cylinder (10.4%)
Percent Of (10.5%)	Surface Area Rectangular Prism (10.2%)

ability estimation is quite small. In contrast, students with ID 27, 49 and 65 answered far less questions and the credible intervals of their ability estimation is quite large. This result implies that VarFA outputs sensible and interpretable credible intervals. As mentioned in Section 8.1, such uncertainty quantification may benefit a number of educational applications such as improving adaptive testing algorithms. Note that VarFA is able to compute such credible interval as fast as SPARFA-M, making VarFA potentially useful for even real-time educational systems. In contrast, SPARFA-M is not capable of computing credible intervals. Although we may still obtain confidence interval as uncertainty quantification with SPARFA-M via bootstrapping, i.e., train with SPARFA-M multiple times with random subsets of the data set and then use the point estimates from different random runs to compute confidence intervals. However, this method suffers from two immediate drawbacks: 1) it is slow because we need to run the model multiple times and 2) different runs result in permutation in the estimated factors and thus the averaged estimations loss meaning. Given that VarFA is as efficient as SPARFA-M and the previous drawbacks of SPARFA-M, we recommend VarFA over bootstrapping with SPARFA-M for quantifying model's uncertainty in practice.

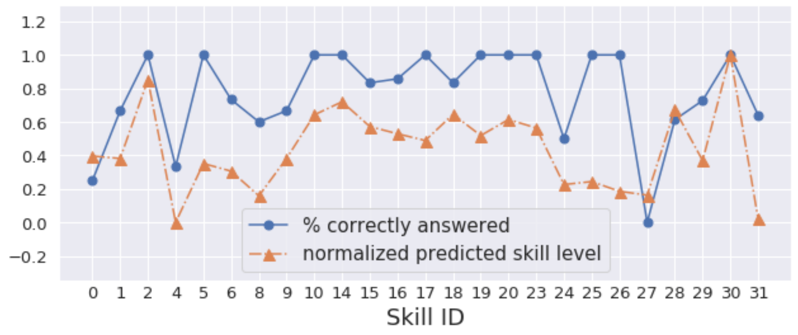


Figure 8.4 : Comparison between the estimated skill mastery levels using VarFA's predictions and using empirical observations for student with ID 110. Even though the two curves show different numeric values, they nevertheless demonstrate similar trends, showing that the predictions reasonably match our intuition about student's skill mastery levels.

Results: Post-Processing for Improved Interpretability

SPARFA assumes that each student factor ν_i identifies a multi-dimensional skill level on a number of "latent" skills (recall that we use 8 latent skills in our experiments). As mentioned earlier, these latent skills are not interpretable without the aid of additional information. To improve interpretability, [175] proposed that, when the skill tags for each question is available in the data set, we can associate each latent skill with skill tags via a simple matrix factorization. Then, we can compute each students' mastery levels on the actual skill tags. We refer readers to Sections 5.1 and 5.2 in [175] for more technical details. Although the above method to improve interpretability has already been proposed, [175] only presented results on private data sets, whereas here we presents results on publicly available data set with code, making our results more transparent and reproducible.

We again use the Assistment data set for illustration. We compute the association of skill tags in the data set with each of the latent skills and show 4 of the latent skills with their top 3 most strongly associated skill tags. We can see that each latent skill roughly identify the same group of skill tags. For example, latent skill 4 clusters skill tags on statistics and probability while latent skill 7 clusters skill tags on geometry. Thus, by simple post-processing, we obtain an interpretation of the latent skills by associating them with known skill tags in the data.

We can similarly obtain VarFA's estimations of the students' mastery levels on each skill tags through the above process. In Fig. 8.4, we compare the predicted mastery level for each skill tag (only for the questions this student answered) with the percent of correct answers for that skill tag. Blue curve shows the empirical student's mastery level on a skill tag by computing the percentage of correctly answered questions belonging to a particular skill tag. Orange curve shows VarFA's estimated student mastery level on a skill tag, normalized to range $[0, 1]$. We can see that, when the student gave more correct answers to questions of a particular skill tag, such as skill tag ID 2, 10, 14 and 30, VarFA also predicts a higher ability score for these skill tags. When the student gave more incorrect answers to questions of a particular skill tag, such as skill tag ID 0, 4 and 27, VarFA also predicts a lower ability score for those skill tags. Although the correspondence is not perfect, VarFA's predicted student's mastery levels match our intuition about student's abilities (using the observed correct answer ratio as an empirical estimate) reasonably well. Thus, by post-processing, we can interpret VarFA's estimated students' latent skill mastery levels using the easily understandable skill tags.

8.5 Conclusions and Future Work

We have presented VarFA, a variational inference factor analysis framework to perform efficient Bayesian inference for learning analytics. VarFA is general and can be applied to a wide array of FA models. We have demonstrated the effectiveness of our VarFA using the sparse factor analysis (SPARFA) model as a case study. We have shown that VarFA can very efficiently output interpretable, educationally meaningful information, in particular credible intervals, much faster than classic Bayesian inference methods. Thus, VarFA has potential application in many educational data mining scenarios where efficient credible interval computation is desired, i.e., in adaptive testing and adaptive learning systems. We have also provided open-source code to reproduce our results and facilitate further research efforts.

We outline three possible future research directions and extensions. First, VarFA currently performs Bayesian inference on the student skill mastery level factor. We are working on extending the framework to perform full Bayesian inference for all unknown factors. Extensions to some models such as IRT is straightforward, as studied in [373], because all factors can be reasonably assumed to be Gaussian. However, some other models involve additional modeling assumptions, making it challenging to design distributions that satisfy these assumptions. For example, in SPARFA, one of the factors is assumed to be nonnegative and sparse [175]. No standard distribution fulfills these requirements. We are exploring *mixture* distributions, in particular spike and slab models [128] for Bayesian variable selection, and methods to combine them with variational inference, following [324, 325].

Second, other methods exist that accelerate classic Bayesian inference. Recent works in large-scale Bayesian inference proposed *approximate* MCMC methods that scale to large data sets [402, 207, 309]. Some of these methods apply variational inference to perform the approximation [67, 103]. We are investigating extending VarFA to support approximate MCMC methods.

Finally, the goal of VarFA is to improve real-world educational systems and ultimately improve learning. Therefore, it is necessary to evaluate VarFA beyond synthetic and benchmark data sets. We plan to integrate VarFA into an existing educational system and conduct a case study where students interact with the system in real-time to understand the VarFA's educational implications in the wild.

Chapter 9

A Max-Affine Spline Perspective of Recurrent Neural Networks

9.1 Introduction

Recurrent neural networks (RNNs) are a powerful class of models for processing sequential inputs and a basic building block for more advanced models that have found success in challenging problems involving sequential data, including sequence classification (e.g., sentiment analysis [308] , sequence generation (e.g., machine translation [12]), speech recognition [100], and image captioning [212]. Despite their success, however, our understanding of how RNNs work remains limited. For instance, an attractive theoretical result is the universal approximation property that states that an RNN can approximate an arbitrary function [290, 304, 104]. These classical theoretical results have been obtained primarily from the dynamical system [304, 290] and measure theory [104] perspectives. These theories provide approximation error bounds but unfortunately limited guidance on applying RNNs and understanding their performance and behavior in practice.

In this work, we provide a new angle for understanding RNNs using *max-affine spline operators* (MASOs) [210, 105] from approximation theory. The piecewise affine approximations made by compositions of MASOs provide a new and useful framework to study neural networks. For example, [17] have provided a detailed analysis in the context of feedforward networks. Here, we go one step further and find new insights and interpretations from the MASO perspective for RNNs. We will see that the input space partitioning and matched filtering links developed in [17] extend to RNNs and yield interesting insights into their inner workings. Moreover, the MASO formulation of RNNs enables us to theoretically justify the use of a random initial hidden state to improve RNN performance.

For concreteness, we focus our analysis on a specific class of simple RNNs [77] with *piecewise affine and convex nonlinearities* such as the ReLU [95]. RNNs with such nonlinearities have recently gained considerable attention due to their ability to combat the *exploding gradient problem*; with proper initialization [179, 320] and clever parametrization of the recurrent weight [10, 372, 142, 125, 225, 110], these RNNs achieve performance on par with more complex ones such as LSTMs. Below is a summary of our key contributions.

Contribution 1. We prove that an RNN with piecewise affine and convex nonlinearities can be rewritten as a composition of MASOs, making it a piecewise affine spline operator with an elegant analytical form (Section 9.3).

Contribution 2. We leverage the partitioning of piecewise affine spline operators to analyze the input space partitioning that an RNN implicitly performs. We show that an RNN calculates a new, high-dimensional representation (the partition code) of the input sequence that captures informative underlying characteristics of the input. We also provide a new perspective on RNN dynamics by visualizing the evolution of the RNN input space partitioning through time (Section 9.4).

Contribution 3. We show the piecewise affine mapping in an RNN associated with a given input sequence corresponds to an input-dependent template, from which we can interpret the RNN as performing greedy template matching (matched filtering) at every RNN cell (Section 9.5).

Contribution 4. We rigorously prove that using a random (rather than zero) initial hidden state in an RNN corresponds to an explicit regularizer that can mollify exploding gradients. We show empirically that such a regularization improves RNN performance (to

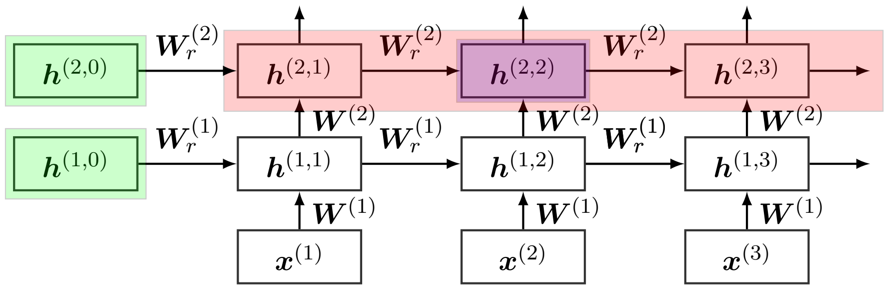


Figure 9.1 : Visualization of an RNN that highlights a cell (purple), a layer (red) and the initial hidden state of each layer (green). (Best viewed in color.)

state-of-the-art) on four datasets of different modalities (Section 9.6).

9.2 Background

Recurrent Neural Networks (RNNs). A simple RNN unit [77] per layer ℓ and time step t, referred to as a "cell," performs the following recursive computation

$$\boldsymbol{h}^{(\ell,t)} = \sigma\left(\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1,t)} + \boldsymbol{W}_r^{(\ell)}\boldsymbol{h}^{(\ell,t-1)} + \boldsymbol{b}^{(\ell)}\right), \tag{9.1}$$

where $\boldsymbol{h}^{(\ell,t)}$ is the hidden state at layer ℓ and time step t, $\boldsymbol{h}^{(0,t)} := \boldsymbol{x}^{(t)}$ which is the input sequence, σ is an activation function and $\boldsymbol{W}^{(\ell)}$, $\boldsymbol{W}_r^{(\ell)}$, and $\boldsymbol{b}^{(\ell)}$ are time-invariant parameters at layer ℓ. $\boldsymbol{h}^{(\ell,0)}$ is the initial hidden state at layer ℓ which needs to be set to some value beforehand to start the RNN recursive computation. Unrolling the RNN through time gives an intuitive view of the RNN dynamics, which we visualize in Figure 9.1. The output of the overall RNN is typically an affine transformation of the hidden state of the last layer L at time step t

$$\boldsymbol{z}^{(t)} = \boldsymbol{W}\boldsymbol{h}^{(L,t)} + \boldsymbol{b}\,. \tag{9.2}$$

In the special case where the RNN has only one output at the end of processing the entire input sequence, the RNN output is an affine transformation of the hidden state at the last time step, i.e., $\boldsymbol{z}^{(T)} = \boldsymbol{W}\boldsymbol{h}^{(L,T)} + \boldsymbol{b}$.

Max-Affine Spline Operators (MASOs). A max-affine spline operator (MASO) is piecewise affine and convex with respect to each output dimension $k = 1, \dots, K$. It is defined as a parametric function $S : \mathbb{R}^D \to \mathbb{R}^K$ with parameters $\boldsymbol{A} \in \mathbb{R}^{K\times R\times D}$ and $\boldsymbol{B} \in \mathbb{R}^{K\times R}$. A MASO leverages K independent max-affine splines [210], each with R partition regions. Its output for output dimension k is produced via

$$[\boldsymbol{y}]_k = [S(\boldsymbol{x})]_k = \max_{r=1,\dots,R} \left\{ \langle [\boldsymbol{A}]_{k,r,\cdot}\,, \boldsymbol{x}\rangle + [\boldsymbol{B}]_{k,r} \right\}, \tag{9.3}$$

where $\boldsymbol{x} \in \mathbb{R}^D$ and $\boldsymbol{y} \in \mathbb{R}^K$ are dummy variables that respectively denote the input and output of the MASO S and $\langle\cdot,\cdot\rangle$ denotes inner product. The three subscripts of the "slope" parameter $[\boldsymbol{A}]_{k,r,d}$ correspond to output k, partition region r, and input signal index d. The two subscripts of the "bias" parameter $[\boldsymbol{B}]_{k,r}$ correspond to output k and partition region r.

We highlight two important and interrelated MASO properties relevant to the discussions throughout the work. First, a MASO performs implicit input space partitioning, which is made explicit by rewriting (9.3) as

$$[\boldsymbol{y}]_k = \sum_{r=1}^{R} [Q]_{k,r} (\langle [\boldsymbol{A}]_{k,r,\cdot}\,, \boldsymbol{x}\rangle + [\boldsymbol{B}]_{k,r})\,, \tag{9.4}$$

where $Q \in \mathbb{R}^{K\times R}$ is a partition selection matrix[1] calculated as

$$[Q]_{k,r} = \mathbb{1}(r = [\boldsymbol{r}^*]_k)\,, \quad \text{where } [\boldsymbol{r}^*]_k = \arg\max_{r=1,\cdots,R} \langle [\boldsymbol{A}]_{k,r,\cdot}\,, \boldsymbol{x}\rangle + [\boldsymbol{B}]_{k,r}\,. \tag{9.5}$$

[1]Prior work denotes the partition selection matrix as T. But in the context of RNNs, T usually denotes the length of the input sequence. Thus we denote this matrix as Q in this work to avoid notation conflicts.

Namely, Q contains K stacked one-hot row vectors, each of which selects the $[\boldsymbol{r}^*]_k^{\mathrm{th}}$ partition of the input space that maximizes (9.4) for output dimension k. As a consequence, knowing Q is equivalent to knowing the partition of an input $\boldsymbol{x}$ that the MASO implicitly computes. We will use this property in Section 9.4 to provide new insights into RNN dynamics.

Second, given the partition $\boldsymbol{r}^*$ that an input belongs to, as determined by (9.5), the output of the MASO of dimension k from (9.3) reduces to a simple affine transformation of the input

$$[\boldsymbol{y}]_k = [A]_{k,\cdot}\boldsymbol{x} + [B]_k \ , \quad \text{where } [A]_{k,\cdot} = [\boldsymbol{A}]_{k,[\boldsymbol{r}^*]_k} \text{ and } [B]_{k,\cdot} = [\boldsymbol{B}]_{k,[\boldsymbol{r}^*]_k} \ . \tag{9.6}$$

Here, the selected affine parameters $A \in \mathbb{R}^{K\times D}$ and $B \in \mathbb{R}^K$ are specific to the input's partition region $[\boldsymbol{r}^*]_k$, which are simply the $[\boldsymbol{r}^*]_k^{\mathrm{th}}$ slice and $[\boldsymbol{r}^*]_k^{\mathrm{th}}$ column of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively, for output dimension k. We emphasize that A and B are input-dependent; different inputs $\boldsymbol{x}$ induce different A and B.[2] We will use this property in Section 9.5 to link RNNs to matched filterbanks.

9.3 RNNs as Piecewise Affine Spline Operators

We now leverage the MASO framework to rewrite, interpret, and analyze RNNs. We focus on RNNs with piecewise affine and convex nonlinearities in order to derive rigorous analytical results. The analysis of RNNs with other nonlinearities is left for future work.

We first derive the MASO formula for an RNN cell (9.1) and then extend to one layer of a time-unrolled RNN and finally to a multi-layer, time-unrolled RNN. Let $\boldsymbol{z}^{(\ell,t)} = \left[\boldsymbol{h}^{(\ell-1,t)^\top}, \boldsymbol{h}^{(\ell,t-1)^\top}\right]^\top$ be the input to an RNN cell that is the concatenation of the current

[2]The notation for the affine spline parameters A and B in [17, 16] are $A[\boldsymbol{x}]$ and $B[\boldsymbol{x}]$, respectively, in order to highlight their input dependency. In this work, we drop the input dependency when writing these affine parameters to simplify the notation, and we use brackets to exclusively denote indexing or concatenation.

input $\boldsymbol{h}^{(\ell-1,t)}$ and the previous hidden state $\boldsymbol{h}^{(\ell,t-1)}$. Then we have the following result, which is a straightforward extension of Proposition 4 in [17].

Proposition 2

An RNN cell of the form (9.1) is a MASO with

$$\boldsymbol{h}^{(\ell,t)} = A^{(\ell,t)}\boldsymbol{z}^{(\ell,t)} + B^{(\ell,t)}\,, \tag{9.7}$$

where $A^{(\ell,t)} = A_\sigma^{(\ell,t)}[\boldsymbol{W}^{(\ell)}, \boldsymbol{W}_r^{(\ell)}]$ *and* $B^{(t)} = A_\sigma^{(\ell,t)}\boldsymbol{b}^{(\ell)}$ *are the affine parameters and* $A_\sigma^{(\ell,t)}$ *is the affine parameter corresponding to the piecewise affine and convex nonlinearity* $\sigma(\cdot)$ *that depends on the cell input* $\boldsymbol{z}^{(\ell,t)}$.

We now derive an explicit affine formula for a time-unrolled RNN at layer ℓ. Let $\boldsymbol{h}^{(\ell-1)} = \left[\boldsymbol{h}^{(\ell-1,1)^\top}, \cdots, \boldsymbol{h}^{(\ell-1,T)^\top}\right]^\top$ be the entire input sequence to the RNN at layer ℓ, and let $\boldsymbol{h}^{(\ell)} = \left[\boldsymbol{h}^{(\ell,1)^\top}, \cdots, \boldsymbol{h}^{(\ell,T)^\top}\right]^\top$ be all the hidden states that are output at layer ℓ. After some algebra and simplification, we arrive at the following result.

Theorem 1

The ℓ^{th} *layer of an RNN is a piecewise affine spline operator defined as*

$$\begin{pmatrix} \boldsymbol{h}^{(\ell,T)} \\ \vdots \\ \boldsymbol{h}^{(\ell,1)} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathcal{A}_{T:T}^{(\ell)} \dots \mathcal{A}_{1:T}^{(\ell)} \\ \vdots \quad \ddots \quad \vdots \\ \boldsymbol{0} \quad \dots \mathcal{A}_{1:1}^{(\ell)} \end{pmatrix}}_{\text{upper triangular}} \underbrace{\begin{pmatrix} A_\sigma^{(\ell,T)}\boldsymbol{W}^{(\ell)} \dots & \boldsymbol{0} \\ \vdots \quad \ddots & \vdots \\ \boldsymbol{0} & \dots A_\sigma^{(\ell,1)}\boldsymbol{W}^{(\ell)} \end{pmatrix}}_{\text{diagonal}} \begin{pmatrix} \boldsymbol{h}^{(\ell-1,T)} \\ \vdots \\ \boldsymbol{h}^{(\ell-1,1)} \end{pmatrix}$$

$$+ \begin{pmatrix} \sum_{t=T}^{1} \mathcal{A}_{t:T}^{(\ell)} B^{(\ell,t)} + \mathcal{A}_{0:T}^{(\ell)}\boldsymbol{h}^{(\ell,0)} \\ \vdots \\ \mathcal{A}_{1:1}^{(\ell)} B^{(\ell,t)} + \mathcal{A}_{0:1}^{(\ell)}\boldsymbol{h}^{(\ell,0)} \end{pmatrix} = A_{\text{RNN}}^{(\ell)}\boldsymbol{h}^{(\ell-1)} + B_{\text{RNN}}^{(\ell)}\,, \tag{9.8}$$

where $\mathcal{A}^{(\ell)}_{t:T'} = \left(\prod_{s=T'}^{t+1} A_\sigma^{(\ell,s)} \boldsymbol{W}_r^{(\ell)}\right)$ for $t < T'$ and identity otherwise,[3] *$\boldsymbol{h}^{(\ell,0)}$ is the initial hidden state of the RNN at layer ℓ, and $A^{(\ell)}_{\mathrm{RNN}}$ and $B^{(\ell)}_{\mathrm{RNN}}$ are affine parameters that depend on the layer input $\boldsymbol{h}^{(\ell-1)}$ and the initial hidden state $\boldsymbol{h}^{(\ell,0)}$.*

We present the proof for Theorem 1 in Appendix C.7. The key point here is that, by leveraging MASOs, we can represent the time-unrolled RNN as a simple affine transformation of the entire input sequence (9.8). Note that this affine transformation changes depending on the partition region in which the input belongs (recall (9.4) and (9.5)). Note also that the initial hidden state affects the layer output by influencing the affine parameters and contributing a bias term $A^{(\ell)}_{0:t}\boldsymbol{h}^{(\ell,0)}$ to the bias parameter $B^{(\ell)}_{\mathrm{RNN}}$. We study the impact of the initial hidden state in more detail in Section 9.6.

We are now ready to generalize the above result to multi-layer RNNs. Let $\boldsymbol{x} = \left[\boldsymbol{x}^{(1)^\top}, \cdots, \boldsymbol{x}^{(T)^\top}\right]^\top$ be the input sequence to a multi-layer RNN, and let $\boldsymbol{z} = \left[\boldsymbol{z}^{(1)^\top}, \cdots, \boldsymbol{z}^{(T)^\top}\right]^\top$ be the output sequence. We state the following result for the overall mapping of a multi-layer RNN.

Theorem 2

The output of an L-layer RNN is a piecewise affine spline operator defined as

$$\boldsymbol{z} = \mathcal{W}\left(\boldsymbol{h}^{(L)}\right) + \boldsymbol{b} = \mathcal{W}\left(A_{\mathrm{RNN}}\boldsymbol{x} + B_{\mathrm{RNN}}\right) + \boldsymbol{b}\,, \tag{9.9}$$

where $A_{\mathrm{RNN}} = \prod_{\ell=L}^{1} A^{(\ell)}_{\mathrm{RNN}}$ and $B_{\mathrm{RNN}} = \sum_{\ell=1}^{L}\left(\prod_{\ell'=\ell}^{L-1} A^{(\ell')}_{\mathrm{RNN}}\right) B^{(\ell)}_{\mathrm{RNN}}$ are the affine parameters of the L-layer RNN. $\mathcal{W}$ and $\boldsymbol{b}$ are parameters of the fully connected output layer, where $\mathcal{W} = [\boldsymbol{W}, \boldsymbol{W}, \ldots, \boldsymbol{W}]$ when the RNN outputs at every time step and $\mathcal{W} = [\boldsymbol{W}, 0, \ldots, 0]$ when the RNN outputs only at the last time step.

[3] In our context, $\prod_{i=m}^{n} a_i := a_m \cdot a_{m-1} \cdots a_{n+1} \cdot a_n$ for $m > n$ as opposed to the empty product.

Theorem 2 shows that, using MASOs, we have a simple, elegant, and closed-form formula showing that the output of an RNN is computed locally via very simple functions. This result is proved by recursively applying the proof for Theorem 1.

The affine mapping formula (9.9) opens many doors for RNN analyses, because we can shed light on RNNs by applying established matrix results. In the next sections, we provide three analyses that follow this programme. First, we show that RNNs partition the input space and that they develop the partitions through time. Second, we analyze the forms of the affine slope parameter and link RNNs to matched filterbanks. Third, we study the impact of the initial hidden state to rigorously justify the use of randomness in initial hidden state. From this point, for simplicity, we will assume a zero initial hidden state unless otherwise stated.

9.4 Internal Input Space Partitioning in RNNs

The MASO viewpoint enables us to see how an RNN implicitly partitions its input sequence through time, which provides a new perspective of its dynamics. To see this, first recall that, for an RNN cell, the piecewise affine and convex activation nonlinearity partitions each dimension of the cell input $\boldsymbol{z}^{(\ell,t)}$ into R regions (for ReLU, $R = 2$). Knowing the state of the nonlinearity (which region r is activated) is thus equivalent to knowing the partition of the cell input. For a multi-layer RNN composed of many RNN cells (recall Figure 9.1), the input sequence partition can be retrieved by accessing the collection of the states of all of the nonlinearities; each input sequence can be represented by a partition "code" that determines the partition to which it belongs.

Since an RNN processes an input sequence one step at a time, the input space partition is gradually built up and refined through time. As a consequence, when seen through the MASO lens, the forward pass of an RNN is simply developing and refining the partition

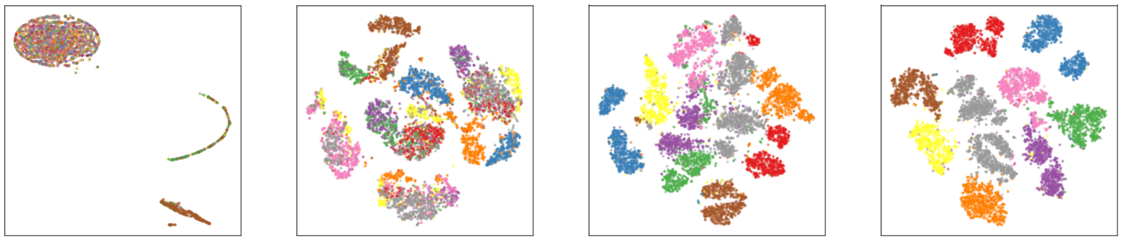


Figure 9.2 : t-SNE [331] visualization of the evolution of the RNN partition codes of input sequences from the MNIST test set. Each color represents one of the ten classes. We see clearly that the RNN gradually develops and refines the partition codes through time to separate the classes.

code of the input sequence. Visualizing the evolution of the partition codes can be potentially beneficial for diagnosing RNNs and understanding their dynamics.

As an example, we demonstrate the evolution of the partition codes of a one-layer ReLU RNN trained on the MNIST dataset, with each image flattened into a 1-dimensional sequence so that input at each time step is a single pixel. Details of the model and experiments are in Appendix C.3. Since the ReLU activation partitions its input space into only 2 regions , we can retrieve the RNN partition codes of the input images simply by binarizing and concatenating all of the hidden states. Figure 9.2 visualizes how the partition codes of MNIST images evolve through time using t-SNE, a distance-preserving dimensionality reduction technique [331]. The figure clearly shows the evolution of the partition codes from hardly any separation between classes of digits to forming more and better separated clusters through time. We can also be assured that the model is well-behaved, since the final partition shows that the images are well clustered based on their labels. Additional visualizations are available in Section C.4.

9.5 RNNs as Matched Filterbanks

The MASO viewpoint enables us to connect RNNs to classical signal processing tools like the matched filter. Indeed, we can directly interpret an RNN as a *matched filterbank*, where the classification decision is informed via the simple inner product between a "template" and the input sequence. To see this, we follow an argument similar to that in Section 9.4. First, note that the slope parameter $A^{(\ell,t)}$ for each RNN cell is a "locally optimal template" because it maximizes each of its output dimensions over the R regions that the nonlinearity induces (recall (9.3) and (9.7)). For a multi-layer RNN composed of many RNN cells, the overall "template" A_{RNN} corresponds to the composition of the optimal templates from each RNN cell, which can be computed simply via $\mathrm{d}\boldsymbol{z}/\mathrm{d}\boldsymbol{x}$ (recall (9.9)).

Thus, we can view an RNN as a matched filterbank whose output is the maximum inner product between the input and the rows of the overall template A_{RNN} [332, 333]. The overall template is also known in the machine learning community as a *salience map*; see [183] for an example of using saliency maps to visualize RNNs. Our new insight here is that a good template produces a larger inner product with the input *regardless of the visual quality of the template*, thus complementing prior work. The template matching view of RNNs thus provides a principled methodology to visualize and diagnose RNNs by examining the inner products between the inputs and the templates.

To illustrate the matched filter interpretation, we train a one-layer ReLU RNN on the polarized Stanford Sentiment Treebank dataset (SST-2) [308], which poses a binary classification problem, and display in Figure 9.3 the templates corresponding to the correct and incorrect classes of an input where the correct class is a negative sentiment. We see that the input has a much larger inner product with the template corresponding to the correct class (left plot) than that corresponding to the incorrect class (right plot), which informs us that the model correctly classifies this input. Additional experimental results are given in

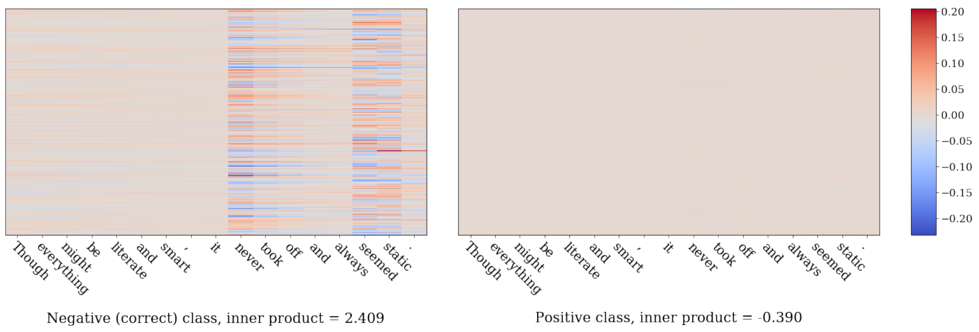


Figure 9.3 : Templates corresponding to the correct (left) and incorrect class (right) of a negative sentiment input from the SST-2 dataset. Each column contains the gradient corresponding to an input word. Quantitatively, we can see that the inner product between input and the correct class template (left) produces a larger value than that between input and the incorrect class template (right).

Appendix C.5.

9.6 Improving RNNs Via Random Initial Hidden State

In this section, we provide a theoretical motivation for the use of a random initial hidden state in RNNs. The initial hidden state needs to be set to some prior value to start the recursion (recall Section 9.2). Little is understood regarding the best choice of initial hidden state other than [414]'s dynamical system argument. Consequently, it is typically simply set to zero. Leveraging the MASO view of RNNs, we now demonstrate that one can improve significantly over a zero initial hidden state by using a *random initial hidden state.* This choice regularizes the affine slope parameter associated with the initial hidden state and mollifies the so-called exploding gradient problem [251].

Random Initial Hidden State as an Explicit Regularization. We first state our theoretical result that using random initial hidden state corresponds to an explicit regularization and then discuss its impact on exploding gradients. Without loss of generality, we focus on one-layer ReLU RNNs. Let N be the number of data points and C the number of classes. Define $\mathcal{A}_h := \mathcal{A}_{1:T} = \prod_{s=T}^{1} A_\sigma^{(s)} \boldsymbol{W}_r^{(\ell)}$ (recall (9.8)).

Theorem 3

Let $\mathcal{L}$ be an RNN loss function, and let $\widetilde{\mathcal{L}}$ represent the modified loss function when the RNN initial hidden state is set to a Gaussian random vector $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_{\boldsymbol{\epsilon}}^2 \boldsymbol{I})$ with small standard deviation $\sigma_{\boldsymbol{\epsilon}}$. Then we have that $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\widetilde{\mathcal{L}}\right] = \mathcal{L} + \mathcal{R}$. For the cross-entropy loss $\mathcal{L}$ with softmax output, $\mathcal{R} = \frac{\sigma_{\boldsymbol{\epsilon}}^2}{2N}\sum_{n=1}^{N}\left\|\operatorname{diag}\left(\left[\frac{\mathrm{d}\widehat{y}_{ni}}{\partial z_{nj}}\right]_{i=j}\right)\mathcal{A}_h\right\|^2$, where $\widehat{y}_{ni}$ is the i^{th} dimension of the softmax output of the n^{th} data point and $i, j \in \{1, \ldots, C\}$ are the class indices. For the mean-squared error loss $\mathcal{L}$, $\mathcal{R} = \frac{\sigma_{\boldsymbol{\epsilon}}^2}{2N}\sum_{n=1}^{N}\|\mathcal{A}_h\|^2$.

We prove this result for the cross-entropy loss in Appendix C.7.2. The standard deviation $\sigma_{\boldsymbol{\epsilon}}$ controls the importance of the regularization term and recovers the case of standard zero initial hidden state when $\sigma_{\boldsymbol{\epsilon}} = 0$.

Connection to the Exploding Gradient Problem. Backpropagation through time (BPTT) is the default RNN training algorithm. Updating the recurrent weight $\boldsymbol{W}_r$ with its gradient using BPTT involves calculating the gradient of the RNN output with respect to the hidden state at each time step $t = 0, \ldots, T$

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{h}^{(t)}} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{z}}\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}\boldsymbol{h}^{(T)}}\left(\prod_{s=T}^{t+1}\frac{\mathrm{d}\boldsymbol{h}^{(T)}}{\mathrm{d}\boldsymbol{h}^{(s)}}\right) = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{z}}\boldsymbol{W}\left(\prod_{s=T}^{t+1}\boldsymbol{A}_\sigma^{(s)}\boldsymbol{W}_r\right). \tag{9.10}$$

When $\|\boldsymbol{A}_\sigma^{(s)}\boldsymbol{W}_r\|_2 > 1$, the product term $\prod_{s=T}^{t+1}\boldsymbol{A}_\sigma^{(s)}\boldsymbol{W}_r$ in (9.10) blows up, which leads to unstable training. This is known as the *exploding gradient problem* [251].

Our key realization is that the gradient of the RNN output with respect to the initial hidden state $\boldsymbol{h}^{(0)}$ features the term $\mathcal{A}_h$ from Theorem 3

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{h}^{(0)}} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{z}}\boldsymbol{W}\left(\prod_{s=T}^{1}\boldsymbol{A}_\sigma^{(s)}\boldsymbol{W}_r\right) = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{z}}\boldsymbol{W}\mathcal{A}_h\,. \tag{9.11}$$

Of all the terms in (9.10), this one involves the most matrix products and hence is the most

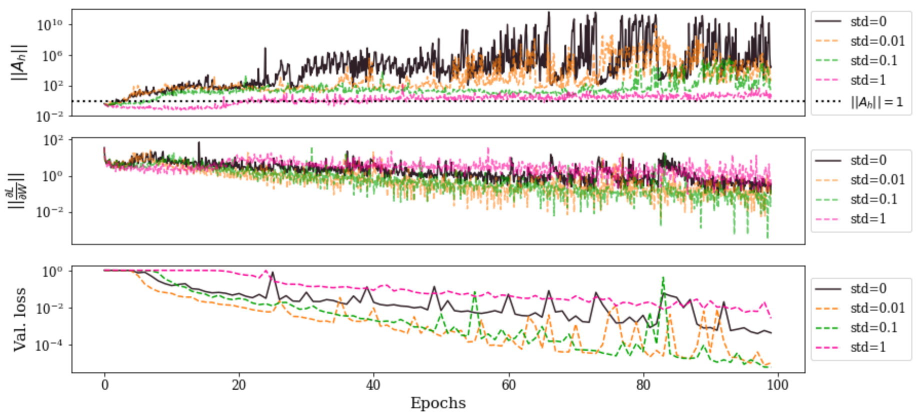


Figure 9.4 : Visualization of the regularization effect of a random initial hidden state on the adding task ($T = 100$). (Top) Norm of $\mathcal{A}_\mathrm{h}$ every 100 iterations; (Middle) norm of the gradient of the recurent weight every 100 iterations; (Bottom) validation loss at every epoch. Each epoch contains 1000 iterations.

erratic. Fortunately, Theorem 3 instructs us that introducing randomness into the initial hidden state effects a regularization on $\mathcal{A}_h$ and hence tamps down the gradient before it can explode. An interesting direction for future work is extending this analysis to every term in (9.10).

Experiments. We now report on the results of a number of experiments that indicate the significant performance gains that can be obtained using a random initial hidden state of properly chosen standard deviation σ_ϵ. Unless otherwise mentioned, in all experiments we use ReLU RNNs with 128-dimensional hidden states and with the recurrent weight matrix $\boldsymbol{W}_r^{(\ell)}$ initialized as an identity matrix [179, 320]. We summarize the experimental results; experimental details and additional results are available in Appendices C.3 and C.6.

Visualizing the Regularizing Effect of a Random Initial Hidden State. We first consider a simulated task of adding 2 sequences of length 100. This is a ternary classification problem with input $\boldsymbol{X} \in \mathbb{R}^{2\times T}$ and target $y \in \{0,1,2\}, y = \sum_i \mathbb{1}_{\boldsymbol{X}_{2i}=1}\boldsymbol{X}_{1i}$. The first row of $\boldsymbol{X}$ contains randomly chosen 0's and 1's; the second row of $\boldsymbol{X}$ contains 1's at 2 randomly chosen indices and 0's everywhere else. Prior work treats this task as a regression task [10];

our regression results are provided in Appendix C.6.1.

In Figure 9.4, we visualize the norm of $\mathcal{A}_h$, the norm of the recurrent weight gradient $\|\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{W}_r}\|$, and the validation loss against training epochs for various random initial state standard deviations. The top two plots clearly demonstrate the effect of the random initial hidden state in regularizing both $\mathcal{A}_h$ and the norm of the recurrent weight gradient, since larger σ_ϵ reduces the magnitudes of both $\mathcal{A}_h$ and $\|\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{W}_r}\|$. Notably, the reduced magnitude of the gradient term $\|\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{W}_r}\|$ empirically demonstrates the mollification of the exploding gradient problem. The bottom plot shows that setting σ_ϵ too large can negatively impact learning. This can be explained as having too much regularization effect. This suggests the question of choosing the best value of σ_ϵ in practice, which we now investigate.

Choosing the Standard Deviation of the Random Initial Hidden State. We examine the effect on performance of different random initial state standard deviations σ_ϵ in RNNs using RMSprop and SGD with varying learning rates. We perform experiments on the MNIST dataset with each image flattened to a length 784 sequence (recall Section 9.4). The full experimental results are included in Appendix C.6.2; here, we report two interesting findings. First, for both optimizers, using a random initial hidden state permits the use of higher learning rates that would lead to an exploding gradient when training without a random initial hidden state. Second, RMSprop is less sensitive to the choice of σ_ϵ than SGD and achieves favorable accuracy even when σ_ϵ is very large (e.g., $\sigma_\epsilon = 5$). This might be due to the gradient smoothing that RMSprop performs during optimization. We therefore recommend the use of RMSprop with a random initial hidden state to improve model performance.

We used RMSprop to train ReLU RNNs of one and two layers with and without random initial hidden state on the MNIST, permuted MNIST[4] and SST-2 datasets. Table 9.1

[4]We apply a fixed permutation to all MNIST images.

Table 9.1 : Classification accuracies on the (permuted) MNIST and SST-2 test sets for various models. A random initial hidden state elevates simple RNNs from also-rans to strong competitors of complex, state-of-the-art models.

Model	**Dataset**		
	MNIST	permuted MNIST	SST-2
RNN, 1 layer, zero initial hidden state	0.970	0.891	0.871
RNN, 1 layer, random initial hidden state	0.981	0.922	0.873
	($\sigma_\epsilon = 0.1$)	($\sigma_\epsilon = 0.01$)	(σ_ϵ=0.1)
RNN, 2 layers, zero initial hidden state	0.969	0.873	0.884
RNN, 2 layers, random initial hidden state	**0.987**	0.927	0.888
	($\sigma_\epsilon = 0.5$)	($\sigma_\epsilon = 0.005$)	(σ_ϵ=0.005)
GRU	0.986	0.888	0.881
LSTM	0.978	0.913	0.849
uRNN [10]	0.951	0.914	–
scoRNN [110]	0.985	**0.966**	–
C-LSTM [408]	–	–	0.878
Tree-LSTM [319]	–	–	0.88
Bi-LSTM+SWN-Lex [323]	–	–	**0.892**

shows the classification accuracies of these models as well as a few state-of-the-art results using complicated models. It is surprising that a random initial hidden state elevates the performance of a simple ReLU RNN to near state-of-the-art performance.

Random Initial Hidden State in Complex RNN Models. Inspired by the results of the previous experiment, we integrated a random initial hidden state into some more complex RNN models. We first evaluate a one-layer gated recurrent unit (GRU) on the MNIST and permuted MNIST datasets, with a random and zero initial hidden state. Although the performance gains are not quite as impressive as those for ReLU RNNs, our results for GRUs still show worthwhile accuract improvements, from 0.986 to 0.987 for MNIST and from 0.888 to 0.904 for permuted MNIST.

We continue our experiments with a more complex, convolutional-recurrent model composed of 4 convolution layers followed by 2 GRU layers [34] and the Bird Audio

Detection Challenge dataset.[5] This binary classification problem aims to detect whether or not an audio recording contains bird songs; see Appendix C.3 for the details. We use the area under the ROC curve (AUC) as the evaluation metric, since the dataset is highly imbalanced. Simply switching from a zero to a random initial hidden state provides a significant boost in the AUC: from 90.5% to 93.4%. These encouraging preliminary results suggest that, while more theoretical and empirical investigations are needed, a random initial hidden state can also boost the performance of complicated RNN models that are not piecewise affine and convex.

9.7 Conclusions and Future Work

We have developed and explored a novel perspective of RNNs in terms of max-affine spline operators (MASOs). RNNs with piecewise affine and convex nonlinearities are piecewise affine spline operators with a simple, elegant analytical form. The connections to input space partitioning (vector quantization) and matched filtering followed immediately. The spline viewpoint also suggested that the typical zero initial hidden state be replaced with a random one that mollifies the exploding gradient problem and improves generalization performance.

There remain abundant promising research directions. First, we can extend the MASO RNN framework following [16] to cover more general networks like gated RNNs (e.g, GRUs, LSTMs) that employ the sigmoid nonlinearity, which is neither piecewise affine nor convex. Second, we can apply recent random matrix theory results [215] to the affine parameter A_{RNN} (e.g., the change of the distribution of its singular values during training) to understand RNN training dynamics.

[5] The leaderboard of benchmarks can be found at `https://goo.gl/TyaFrd`.

Chapter 10

RetMol: Retrieval-based Controllable Molecule Generation

10.1 Introduction

Drug discovery is a complex, multi-objective problem [329]. For a drug to be safe and effective, the molecular entity must interact favorably with the desired target [249], possess favorable physicochemical properties such as solubility [222], and be readily synthesizable [136]. Compounding the challenge is the massive search space (up to 10^{60} molecules [261]). Previous efforts address this challenge via high-throughput virtual screening (HTVS) techniques [346] by searching against existing molecular databases. Combinatorial approaches have also been proposed to enumerate molecules beyond the space of established drug-like molecule datasets. For example, genetic-algorithm (GA) based methods [306, 132, 388] explore potential new drug candidates via heuristics such as hand-crafted rules and random mutations. Although widely adopted in practice, these methods tend to be inefficient and computationally expensive due to the vast chemical search space [118]. The performance of these combinatorial approaches also heavily depends on the quality of generation rules, which often require task-specific engineering expertise and may limit the diversity of the generated molecules.

To this end, recent research focuses on *learning* to controllably synthesize molecules with *generative models* [321, 41, 345]. It usually involves first training an unconditional generative model from millions of existing molecules [370, 127] and then controlling the generative models to synthesize new desired molecules that satisfy one or more property constraints such as high drug-likeness [20] and high synthesizability [78]. There are three main classes of learning-based molecule generation approaches: (i) reinforcement-learning (RL)-based methods [241, 137], (ii) supervised-learning (SL)-based methods [193, 299], and (iii) latent optimization-based methods [371, 64]. RL- and SL-based methods train or fine-tune a pre-trained generative model using the desired properties as reward functions (RL) or using molecules with the desired properties as training data (SL). Such methods require heavy task-specific fine-tuning, making them not easily applicable to a wide range of drug discovery tasks. Latent optimization-based methods, in contrast, learn to find latent codes that correspond to the desired molecules, based on property predictors trained on the generative model's latent space. However, training such latent-space property predictors can be challenging, especially in real-world scenarios where we only have a limited number of active molecules for training [141, 124]. Moreover, such methods usually necessitate a fixed-dimensional latent-variable generative model with a compact and structured latent space [382, 381], making them incompatible with other generative models with a varying-dimensional latent space, such as transformer-based architectures [127].

Our approach. In this work, we aim to overcome the aforementioned challenges of existing works and design a controllable molecule generation method that (i) easily generalizes to various generation tasks; (ii) requires minimal training or fine-tuning; and (iii) operates favorably in data-sparse regimes where active molecules are limited. We summarize our contributions as follows:

[1] We propose a first-of-its-kind retrieval-based framework, termed RetMol, for controllable molecule generation. It uses a small set of exemplar molecules, which may partially satisfy the desired properties, from a retrieval database to guide generation towards satisfying all the desired properties.

[2] We design a retrieval mechanism that retrieves and fuses the exemplar molecules

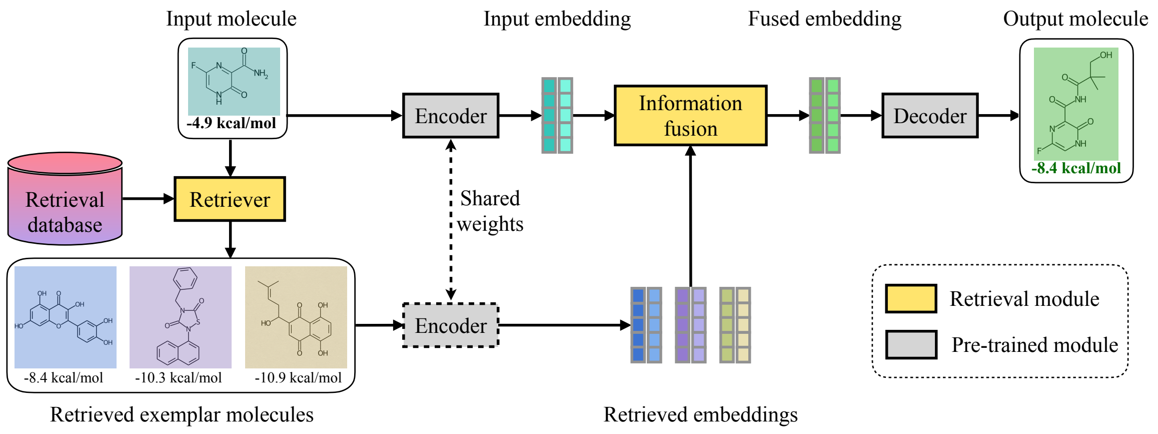


Figure 10.1 : An illustration of RetMol, a retrieval-based framework for controllable molecule generation. The framework incorporates a retrieval module (the molecule retriever and the information fusion) with a pre-trained generative model (the encoder and decoder). The illustration shows an example of optimizing the binding affinity (unit in `kcal/mol`; the lower the better) for an existing potential drug, Favipiravir, for better treating the COVID-19 virus (SARS-CoV-2 main protease, PDB ID: 7L11) under various other design criteria.

with the input molecule, a new self-supervised training with the molecule similarity as a proxy objective, and an iterative refinement process to dynamically update the generated molecules and retrieval database.

[3] We perform extensive evaluation of RetMol on a number of controllable molecule generation tasks ranging from simple molecule property control to challenging real-world drug design for treating the COVID-19 virus, and demonstrate RetMol's superior performance compared to previous methods.

Specifically, as shown in Figure 10.1, the RetMol framework plugs a lightweight retrieval mechanism into a pre-trained, encoder-decoder generative model. For each task, we first construct a *retrieval database* consisting of exemplar molecules that (partially) satisfy the design criteria. Given an input molecule to be optimized, a *retriever* module uses it to retrieve a small number of exemplar molecules from the database, which are then converted into numerical embeddings, along with the input molecule, by the encoder of the pre-trained generative model. Next, an *information fusion* module fuses the embeddings of exemplar molecules with the input embedding to guide the generation (via the decoder in the pre-trained generative model) towards satisfying the desired properties. The fusion module is the only part in RetMol that requires training. For training, we propose a new self-supervised objective (*i.e.*, predicting the nearest neighbor of the input molecule) to update the fusion module, which enables RetMol to generalize to various generation tasks without task-specific training/fine-tuning. For inference, we propose a new inference process via iterative refinement that dynamically updates the generated molecules and retrieval database, which leads to improved generation quality and enables RetMol to extrapolate well beyond the retrieval database.

RetMol enjoys several advantages. First, it requires only a handful of exemplar molecules to achieve strong controllable generation performance without task-specific fine-tuning. This makes it particularly appealing in real-world use cases where training or fine-tuning a task-specific model is challenging. Second, RetMol is flexible and easily adaptable. It is compatible with a range of pre-trained generative models, including both fixed-dimensional and varying-dimensional latent-variable models, and requires training only a lightweight, plug-and-play, task-agnostic retrieval module while freezing the base generative model. Once trained, it can be applied to many drug discovery tasks by simply replacing the retrieval database while keeping all other components unchanged.

In a range of molecule controllable generation scenarios and benchmarks with diverse design criteria, our framework achieves state-of-the-art performance compared to latest methods. For example, on a challenging four-property controllable generation task, our framework improves the success rate over the best method by 4.6% (96.9% vs. 92.3%) with better synthesis novelty and diversity. Using a retrieval database of only 23 molecules, we also demonstrate our real-world applicability on a frontier drug discovery task of optimizing the binding affinity of eight existing, weakly-binding drugs for COVID-19 treatment under multiple design criteria. Compared to the best performing approach, our framework succeeds more often at generating new molecules with the given design criteria (62.5% vs. 37.5% success rate) and generates on average more potent optimized drug molecules (2.84 vs. 1.67 `kcal/mol` average binding affinity improvement over the original drug molecules).

10.2 Methodology: the RetMol Framework

We now detail the various components in RetMol and how we perform training and inference.

Problem setup. We focus on the *multi-property* controllable generation setting in this work. Concretely, let $x \in \mathcal{X}$ be a molecule where $\mathcal{X}$ denotes the set of all molecules, $a_\ell(x) : \mathcal{X} \to \mathbb{R}$ a property predictor indexed by $\ell \in [1, \ldots, L]$, and $\delta_\ell \in \mathbb{R}$ a desired threshold. Then, we formulate multi-property controllable generation as one of three problems below, differing in the control design criteria: (i) a *constraint satisfaction* problem, where we identify a set of new molecules such that $\{x \in \mathcal{X} \,|\, a_\ell(x) \geq \delta_\ell, \forall\, \ell\}$, (ii) an *unconstrained optimization* problem, where we find a set of new molecules such that $x' = \mathrm{argmax}_x\, s(x)$ where $s(x) = \sum_{\ell=1}^{L} w_\ell a_\ell(x)$ with the weighting coefficient w_ℓ, and (iii) a *constrained optimization* problem that combines the objectives in (i) and (ii).

10.2.1 RetMol Components

Encoder-decoder generative model backbone. The pre-trained molecule generative model forms the backbone of RetMol that interfaces between the continuous embedding and raw molecule representations. Specifically, the encoder encodes the incoming molecules into numerical embeddings and the decoder generates new molecules from an embedding, respectively. RetMol is agnostic to the choice of the underlying encoder and decoder architectures, enabling it to work with a variety of generative models and molecule representations. In this work, we consider the SMILES string [364] representation of molecules and the ChemFormer model [127], which a variant of BART [182] trained on the billion-scale ZINC dataset [126] and achieves state-of-the-art generation performance.

Retrieval database. The retrieval database $\mathcal{X}_R$ contains molecules that can potentially serve as exemplar molecules to steer the generation towards the design criteria and is thus vital for controllable generation. The construction of the retrieval database is task-specific: it usually contains molecules that at least partially satisfy the design criteria in a given task. The domain knowledge of what molecules meet the design criteria and how to select partially satisfied molecules can play an important role in our approach. Thus, our approach is essentially a hybrid system that combines the advantages of both the heuristic-based methods and learning-based methods. Also, we find that a database of only a handful of molecules (e.g., as few as 23) can already provide a strong control signal. This makes our approach easily adapted to various tasks by quickly replacing the retrieval database. Furthermore, the retrieval database can be dynamically updated during inference, i.e., newly generated molecules can enrich the retrieval database for better generalization (see Section 10.2.3).

Molecule retriever. While the entire retrieval database can be used during generation, for computational reasons (e.g., memory and efficiency) it is more feasible to select a small portion of the most relevant exemplar molecules to provide a more accurate guidance. We design a simple heuristic-based retriever that retrieves the exemplar molecules most suitable for the given control design criteria. Specifically, we first construct a "feasible" set containing molecules that satisfy all the given constraints, i.e., $\mathcal{X}' = \cap_{\ell=1}^{L} \{x \in \mathcal{X}_R \,|\, a_\ell(x) \geq \delta_\ell\}$. If this set is larger than K, i.e., the number of exemplar molecules that we wish to retrieve, then we select K molecules with the best property scores, i.e., $\mathcal{X}_r = \text{top}_K(\mathcal{X}', s)$, where $s(x)$ is the task-specific weighted average property score, defined in Section 5.3. Otherwise, we construct a relaxed feasible set by removing the constraints one at a time until the relaxed feasible set is larger than K, at which point we select K molecules with the best scores of the most recently removed property constraints. In either case, the retriever retrieves exemplar molecules with more desirable properties than the input and guides the generation towards the given design criteria. We summarize this procedure in Algorithm 2 in Appendix D.1. We find that our simple retriever with $K = 10$ works well across a range of tasks. In general, more sophisticated retriever designs are possible and we leave them as the future work.

Information fusion. This module enables the retrieved exemplar molecules to modify the input molecule towards the targeted design criteria. It achieves this by merging the embeddings of the input and the retrieved exemplar molecules with a lightweight, trainable, standard cross attention mechanism similar to that in [27]. Concretely, the fused embedding $\boldsymbol{e}$ is given by

$$\boldsymbol{e} = f_{\text{CA}}(\boldsymbol{e}_{\text{in}}, \boldsymbol{E}_r; \theta) = \text{Attn}(\text{Query}(\boldsymbol{e}_{\text{in}}), \text{Key}(\boldsymbol{E}_r)) \cdot \text{Value}(\boldsymbol{E}_r) \tag{10.1}$$

where f_{CA} represents the cross attention function with parameters θ, and $\boldsymbol{e}_{\text{in}}$ and $\boldsymbol{E}_r$ are the input embedding and retrieved exemplar embeddings, respectively. The functions Attn, Query, Key, and Value compute the cross attention weights and the query, key, and value matrices, respectively. For our choice of the transformer-based generative model [127], we have that $\boldsymbol{e}_{\text{in}} = \text{Enc}(x_{\text{in}}) \in \mathbb{R}^{L \times D}$, $\boldsymbol{E}_r = [\boldsymbol{e}_r^1, \ldots, \boldsymbol{e}_r^K] \in \mathbb{R}^{(\sum_{k=1}^K L_k) \times D}$, and $\boldsymbol{e}_r^k = \text{Enc}(x_r^k) \in \mathbb{R}^{L_k \times D}$ where L and L_k are the lengths of the tokenized input and the k^{th} retrieved exemplar molecules, respectively, and D is the dimension of each token representation. Intuitively, the Attn function learns to weigh the retrieved exemplar molecules differently such that the more "important" retrieved molecules correspond to the higher weights. The fused embedding $\boldsymbol{e} \in \mathbb{R}^{L \times D}$ thus contains the information of desired properties extracted from the retrieved exemplar molecules, which serves as the input of the decoder to control its generation. More details of this module are available in Appendix D.1.

10.2.2 Training via Predicting the Input Molecule's Nearest Neighbor

The conventional training objective that reconstructs the input molecule (e.g., in Chem-Former [127]) is not appropriate in our case, since perfectly reconstructing the input molecule does not rely on the retrieved exemplar modules (more details in Appendix D.1). To enable RetMol to learn to use the exemplar molecules for controllable generation, we propose a new self-supervised training scheme, where the objective is to predict the *nearest neighbor* of the input molecule:

$$\mathcal{L}(\theta) = \sum_{i=1}^{\mathcal{B}} \mathrm{CE}\Big(\mathrm{Dec}\big(f_{\mathrm{CA}}(\boldsymbol{e}_{\mathrm{in}}^{(i)}, \boldsymbol{E}_r^{(i)}; \theta)\big), x_{\mathrm{1NN}}^{(i)}\Big) . \tag{10.2}$$

where CE is the cross entropy loss function since we use the BART model as our encoder and decoder (see more details in Appendix D.1), x_{1NN} represents the nearest neighbor of the input x_{in}, $\mathcal{B}$ is the batch size, and i indexes the input molecules. The set of retrieved exemplar molecules consists of the remaining $K-1$ nearest neighbors of the input molecule. During training, we freeze the parameters of the pre-trained encoder and decoder. Instead, we only update the parameters in the information fusion module f_{CA}, which makes our training lightweight and efficient. Furthermore, we use the full training dataset as the retrieval database and retrieve exemplar molecules by best similarity with the input molecule. Thus, our training is not task-specific yet forces the fusion module to be involved in the controllable generation with the similarity as a proxy criterion. We will show that during the inference time, the model trained with the above training objective and proxy criterion using similarity is able to generalize to different generation tasks with different design criteria, and performs better compared to training with the conventional reconstruction objective. For the efficient training, we pre-compute all the molecules' embeddings and their pairwise similarities with efficient approximate kNN algorithms [101, 143].

Remarks. The above proposed training strategy that predicts the most similar molecule based on the input molecule and other $K - 1$ similar molecules shares some similarity with masked language model pre-training in NLP [71]. However, there are several key differences: 1) we perform "masking" on a sequence level, i.e., a whole molecule, instead of on a token/word level; 2) our "masking" strategy is to predict a particular signal only (i.e., the most similar retrieved molecule to the input) instead of randomly masked tokens; and 3) our training objective is used to only update the lightweight retrieval module instead of updating the whole backbone generative model.

10.2.3 Inference via Iterative Refinement

We propose an iterative refinement strategy to obtain an improved outcome when controlling generation with implicit guidance derived from the exemplar molecules. The strategy works by replacing the input x_{in} and updating the retrieval database with the newly generated molecules. Such an iterative update is common in many other controllable generation approaches such as GA-based methods [306, 132] and latent optimization-based methods [50, 64]. Furthermore, if the retrieval database $\mathcal{X}_R$ is fixed during inference, our method is constrained by the best property values of molecules in the retrieval database, which greatly limits the generalization ability of our approach and also limits our performance for certain generation scenarios such as unconstrained optimization. Thus, to extrapolate beyond the database, we propose to dynamically update the retrieval database over iterations.

We consider the following iterative refinement process. We first randomly perturb the fused embedding M times by adding Guassian noises and greedily decode one molecule from each perturbed embedding to obtain M generated molecules. We score these molecules according to task-specific design criteria. Then, we replace the input molecule with the best molecule in this set if its score is better than that of the input molecule. At the same time, we add the remaining ones to the retrieval database if they have better scores than the lowest score in the retrieval database. If none of the generated molecules has a score better than the input molecule or the lowest in the retrieval database, then the input molecule and the retrieval database stays the same for the next iteration. We also add a stop condition if the maximum allowable number of iterations is achieved or a successful molecule that satisfies the desired criteria is generated. Algorithm 3 in Appendix D.1 summarizes the procedure.

10.3 Experiments

We conduct four sets of experiments, covering all controllable generation formulations (see "Problem setup" in Section 5.3) with increasing difficulty. For fair comparisons, in each experiment we faithfully follow the same setup as the baselines, including using the same number of optimization iterations and evaluating on the same set of input molecules to be optimized. Below, we summarize the main results and defer the detailed experiment setup and additional results to Appendix D.2 and D.3.

10.3.1 Improving QED and Penalized logP Under Similarity Constraint

These experiments aim to generate new molecules that improve upon the input molecule's intrinsic properties including QED [20] and penalized logP (defined by logP(x)$-$SA(x)) [138] where SA represents synthesizability [78]) under similarity constraints. For both experiments, we use the top 1k molecules with the best property values from the ZINC250k dataset [138] as the retrieval database. In each iteration, we retrieve $K = 20$ exemplar molecules with the best property score that also satisfy the similarity constraint. The

Table 10.1 : Under the similarity constraints, RetMol achieves higher success rate in the constrained QED optimization task and better score improvements in the constrained penalized logP optimization task. Baseline results are reported from [118].

(a) Success rate of generated molecules that satisfy QED $\in [0.9, 1.0]$ under similarity constraint $\delta = 0.4$.

Method	**Success (%)**
MMPA [63]	32.9
JT-VAE [138]	8.8
GCPN [389]	9.4
VSeq2Seq [12]	58.5
VJTNN+GAN [140]	60.6
AtomG2G [139]	73.6
HierG2G [139]	76.9
DESMILES [214]	77.8
QMO [118]	92.8
RetMol	**94.5**

(b) The average penalized logP improvements of generated molecules over inputs under similarity constraint $\delta = \{0.6, 0.4\}$.

Method	**Improvement**	
	$\delta = 0.6$	$\delta = 0.4$
JT-VAE [138]	0.28 ± 0.79	1.03 ± 1.39
GCPN [389]	0.79 ± 0.63	2.49 ± 1.30
MolDQN [410]	1.86 ± 1.21	3.37 ± 1.62
VSeq2Seq [12]	2.33 ± 1.17	3.37 ± 1.75
VJTNN [140]	2.33 ± 1.24	3.55 ± 1.67
HierG2G [139]	2.49 ± 1.09	3.98 ± 1.46
GA [239]	3.44 ± 1.09	5.93 ± 1.41
QMO [118]	3.73 ± 2.85	7.71 ± 5.65
RetMol	$\mathbf{3.78 \pm 3.29}$	$\mathbf{11.55 \pm 11.27}$

remaining configurations exactly follow [118].

QED experiment setup and results. This is a constraint satisfaction problem with the goal of generating new molecules x' such that $a_{\text{sim}}(x', x) \geq \delta = 0.4$ and $a_{\text{QED}}(x') \geq 0.9$ where x is the input molecule, a_{sim} is the Tanimoto similarity function [13], and a_{QED} is the QED predictor. We select 800 molecules with QED in the range $[0.7, 0.8]$ as inputs to be optimized. We measure performance by success rate, i.e., the percentage of input molecules that result in a generated molecule that satisfy both constraints. Table 10.1a shows the success rate of QED optimization by comparing RetMol with various baselines. We can see that RetMol achieves the best success rate, e.g., 94.5% versus 92.8% compared to the best existing approach.

Penalized logP setup and results. This is a constrained optimization problem with the goal to generate new molecules x' to maximize the penalized logP value $a_{\text{plogP}}(x')$ with

Table 10.2 : Success rate, novelty and diversity of generated molecules in the task of optimizing four properties: QED, SA, and two binding affinities to GSK3β and JNK3 estimated by pre-trained models from [137]. Baseline results are reported from [137, 378].

Method	**Success %**	**Novelty**	**Diversity**
JT-VAE [138]	1.3	-	-
GVAE-RL [137]	2.1	-	-
GCPN [389]	4.0	-	-
REINVENT [241]	47.9	0.561	0.621
RationaleRL [137]	74.8	0.568	0.701
MARS [378]	92.3	0.824	0.719
MolEvol [40]	93.0	0.757	0.681
RetMol	**96.9**	**0.862**	**0.732**

similarity constraint $a_{\text{sim}}(x', x) \geq \delta$ where $\delta \in \{0.4, 0.6\}$. We select 800 molecules that have the lowest penalized logP values in the ZINC250k dataset as inputs to be optimized. We measure performance by the relative improvement in penalized logP between the generated and input molecules, averaged over all 800 input molecules. Table 10.1b shows the results comparing RetMol to various baselines. RetMol outperforms the best existing method for both similarity constraint thresholds and, for the $\delta = 0.4$ case, improves upon the best existing method by almost 50%. The large variance in RetMol is due to large penalized logP values of a few generated molecules, and is thus not indicative of poor statistical significance. We provide a detailed analysis of this phenomenon in Appendix D.3.

10.3.2 Optimizing GSK3β and JNK3 Inhibition Under QED and SA Constraints

This experiment aims to generate novel, strong inhibitors for jointly inhibiting both GSK3β (Glycogen synthase kinase 3 beta) and JNK3 (-Jun N-terminal kinase-3) enzymes, which are relevant for potential treatment of Alzheimer's disease [137, 190]. Following [137], we formulate this task as a constraint satisfaction problem with four property constraints: two positivity constraints $a_{\text{GSK3}\beta}(x') \geq 0.5$ and $a_{\text{JNK3}}(x') \geq 0.5$, and two molecule property constraints of QED and SA that $a_{\text{QED}}(x') \geq 0.6$ and $a_{\text{SA}}(x') \leq 4$. Note that $a_{\text{GSK3}\beta}$ and a_{JNK3} are property predictors [137, 190] for GSK3β and JNK3, respectively, and higher values indicate better inhibition against the respective proteins. The retrieval database consists of all the molecules from the CheMBL [88] dataset (approx. 700) that satisfy the above four constraints and we retrieve $K = 10$ exemplar molecules most similar to the input each time. For evaluation, we compute three metrics including success rate, novelty, and diversity according to [137].

Table 10.2 shows that RetMol outperforms all previous methods on the four-property molecule optimization task. In particular, our framework achieves these results without task-specific fine-tuning required for RationaleRL and REINVENT. Besides, RetMol is computationally much more efficient than MARS, which requires 550 iterations model training whereas RetMol requires only 80 iterations (see more details in Appendix D.2.5). MARS also requires test-time, adaptive model training per sampling iteration which further increases the computational overhead during generation.

10.3.3 Guacamol Benchmark Multiple Property Optimization

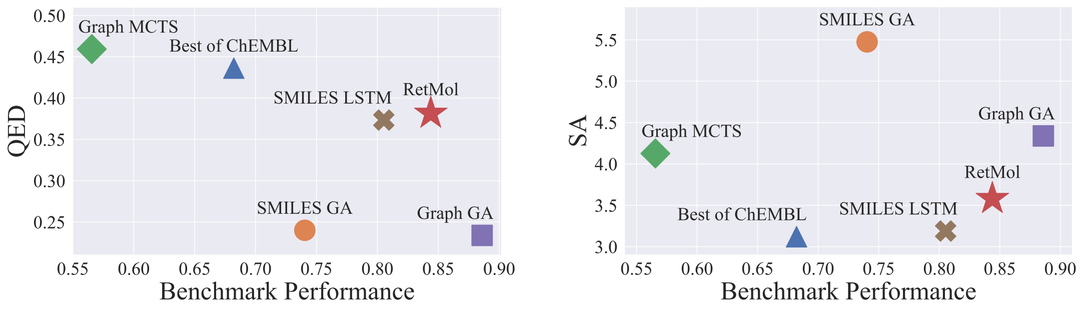


Figure 10.2 : Comparison with the state-of-the-art methods in the multiple property optimization (MPO) tasks on the Guacamol benchmark. **Left**: QED (↑) versus the averaged benchmark performance (↑). **Right**: SA (↓) versus the average benchmark performance (↑). RetMol achieves the best balance between improving the benchmark performance while maintaining the synthesizability (SA) and drug-likeness (QED) of generated molecules.

This experiment evaluates RetMol on the seven multiple property optimization (MPO) tasks, a subset of tasks in the Guacamol benchmark [29]. They are the unconstrained optimization problems to maximize a weighted sum of multiple diverse molecule properties. We choose these MPO tasks because 1) they represent a more challenging subset in the benchmark, as existing methods can achieve almost perfect performance on most of the other tasks [29], and 2) they are the most relevant tasks to our work, e.g., multi-property optimization. The retrieval database consists of 1k molecules with best scores for each task and we retrieve $K = 10$ exemplar molecules with the highest scores each time.

We demonstrate that RetMol achieves the best results along the Pareto frontier of the molecular design space. Figure 10.2 visualizes the benchmark performance averaged over the seven MPO tasks against QED, SA, two metrics that evaluate the drug-likeness and synthesizability of the optimized molecules and that are not part of the Guacamol benchmark's optimization objective. Our framework achieves a nice balance between optimizing benchmark performance and maintaining good QED and SA scores. In contrast, Graph GA [132] achieves the best benchmark performance but suffers from low QED and high SA scores. These results demonstrate the advantage of retrieval-based controllable generation: because the retrieval database usually contains drug-like molecules with desirable properties as high QED and low SA, the generation is guided by these molecules in a good way to not deviate too much from these desirable properties. Moreover, because the retrieval database is updated with newly generated molecules with better benchmark performance, the generation can produce molecules with benchmark score beyond the best in the initial retrieval database. Additional results in Figure 10.2 in Appendix D.3.3 corroborate with those presented above.

Table 10.3 : Quantitative results in the COVID-19 drug optimization task, where we aim to improve selected molecules' binding affinity (estimated via docking [298]) to the SARS-CoV-2 main protease under the QED, SA, and similarity constraints. Under stricter similarity condition, RetMol succeeds in more cases (5/8 versus 3/8). Under milder similarity condition, RetMol achieves higher improvements (2.84 versus 1.67 average binding affinity improvements). Unit of numbers in the table is `kcal/mol` and lower is better.

		$\delta = 0.6$		$\delta = 0.4$	
Input molecule	**Input score**	**RetMol**	**Graph GA** [132]	**RetMol**	**Graph GA** [132]
Favipiravir	-4.93	-6.48	**-7.10**	**-8.70**	-7.10
Bromhexine	-9.64	**-11.48**	-11.20	**-12.65**	-11.83
PX-12	-6.13	**-8.45**	-8.07	**-10.90**	-8.31
Ebselen	-7.31	-	-	**-10.82**	-10.41
Disulfiram	-8.58	**-9.09**	-	**-10.44**	-10.00
Entecavir	-9.00	-	-	**-12.34**	-
Quercetin	-9.25	-	-	**-9.84**	9.81
Kaempferol	-8.45	**-8.54**	-	**-10.35**	10.19
Avg. Improvement	-	**0.78**	0.71	**2.84**	1.67

10.3.4 Optimizing Existing Inhibitors for SARS-CoV-2 Main Protease

To demonstrate our framework's applicability at the frontiers of drug discovery, we apply it to the real-world task of improving the inhibition of existing weak inhibitors against the SARS-CoV-2 main protease (M^{pro}, PDB ID: 7L11), which is a promising target for treating COVID-19 by neutralizing the SARS-CoV-2 virus [87, 397]. Because the the novel nature of this virus, there exist few high-potency inhibitors, making it challenging for learning-based methods that require a sizable training set not yet attainable in this case. However, the few existing inhibitors make excellent candidates for the retrieval database in our framework. We use a set of 23 known inhibitors [141, 124, 118] as the retrieval database and select the 8 weakest inhibitors to M^{pro} as input. We design an optimization task to maximize the binding affinity (estimated via docking [298]; see Appendix D.2 for the detailed procedure) between the generated molecule and M^{pro} while satisfying the following three constraints:

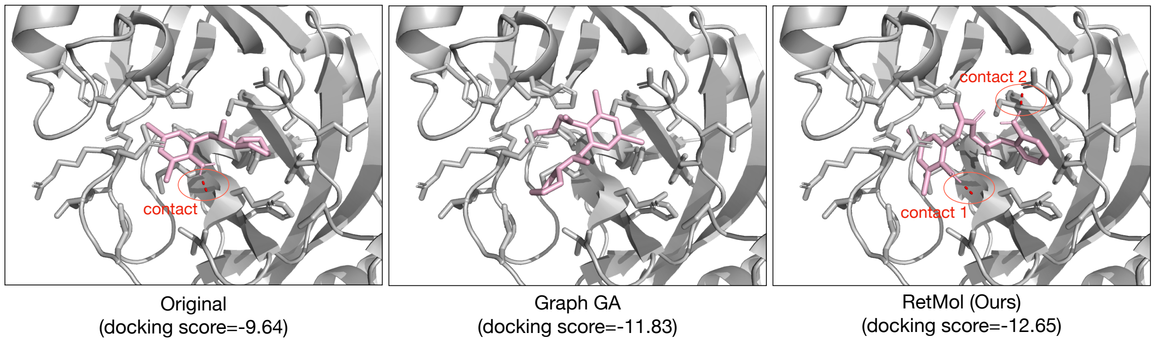


Figure 10.3 : 3D visualizations that compare RetMol with Graph GA in optimizing the original inhibitor, Bromhexine, that binds to the SARS-CoV-2 main protease in the $\delta = 0.6$ case. We can see the optimized inhibitor in RetMol has more polar contacts (red dotted lines) and also more disparate binding modes with the original compound than the Graph GA optimized inhibitor, which aligns with the quantitative results.

$a_{\text{QED}}(x') \geq 0.6$, $a_{\text{SA}}(x') \leq 4$, and $a_{\text{sim}}(x', x) \geq \delta$ where $\delta \in \{0.4, 0.6\}$.

Table 10.3 shows the optimization results of comparing RetMol with graph GA, a competitive baseline. Under stricter ($\delta = 0.6$) similarity constraint, RetMol successfully optimizes the most molecules constraint (5 out of 8) while graph GA fails to optimize input molecules most of the time because it cannot satisfy the constraints or improve upon the input drugs' binding affinity. Under milder ($\delta = 0.4$) similarity constraint, our framework achieves on average higher improvement (2.84 versus 1.67 binding affinity improvements) compared to the baseline. We have also tested QMO [118], and find it unable to generate molecules that satisfy all the given constraints.

We produce 3D visualizations in Figure 10.3 (along with a video demo in `https://shorturl.at/fmtyS`) to show the comparison of the RetMol-optimized inhibitors that bind to the SARS-CoV-2 main protease with the original and GA-optimized inhibitors. We can see that 1) there are more polar contacts (shown as red dotted lines around the molecule) in the RetMol-optimized compound, and 2) the binding mode of the GA-optimized molecule is much more similar to the original compound than the RetMol-optimized binding

Table 10.4 : **Left**: Comparing different training schemes in the unconditional generation setting. **Right**: Generation performance with different retrieval database constructions based on the experiment in Section 10.3.2.

Training objective	**Validity**	**Novelty**	**Uniqueness**
RetMol (predict NN)	**0.902**	**0.998**	**0.922**
Conventional (recon. input)	0.834	0.998	0.665

Ret. database construction	**Success %**	**Novelty**	**Diversity**
GSK3β + JNK3 + QED + SA	**96.9**	**0.862**	**0.732**
GSK3β + JNK3	84.7	0.736	0.700
GSK3β or JNK3	44.1	0.571	0.708

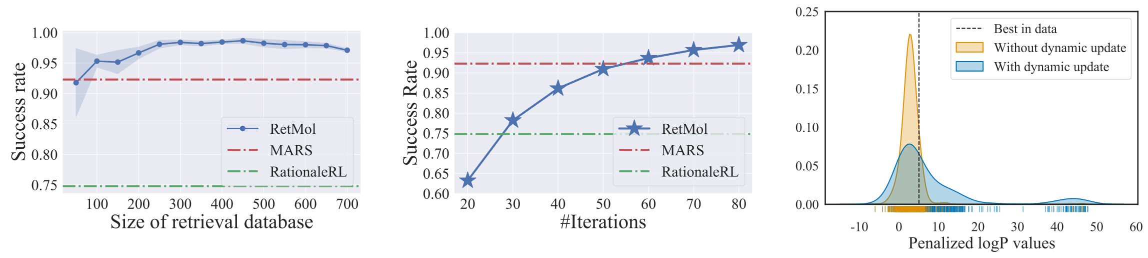


Figure 10.4 : Generation performance with varying retrieval database size (**left**), varying number of iterations (**middle**), and with or without dynamically updating the retrieval database (**right**). The left two plots are based on the experiment in Section 10.3.2 while the right plot is based on the penalized logP experiment in Section 10.3.1.

mode, implying RetMol optimizes the compound beyond local edits to the scaffold. These qualitative and quantitative results together demonstrate that RetMol has the potential to effectively optimize inhibitors in a real-world scenario. Besides, we provide extensive 2D graph visualizations of optimized molecules in Tables D.10 and D.11 in the Appendix. Finally, we also perform another set of experiments on Antibacterial drug design for the MurD protein [288] and observe similar results; see Appendix D.3.5 for more details.

10.3.5 Analyses

We analyze how the design choices in our framework impact generation performance. We summarize the main results and defer the detailed settings and additional results to Appendix D.2 and D.3. Unless otherwise stated, we perform all analyses on the experiment setting in Section 10.3.2.

Training objectives. We evaluate how the proposed self-training objective in Sec. 10.2.2 affects the quality of generated molecules in the unconditional generation setting. Table 10.4 (**left**) shows that, training with our proposed nearest neighbor objective (i.e., predict NN) indeed achieves better generation performance than the conventional input reconstruction objective (i.e., recon. input).

Types of retrieval database. We evaluate how the retrieval database construction impacts controllable generation performance, by comparing four different constructions: with molecules that satisfy all four constraints, i.e., GSK3β + JNK3 + QED + SA; or satisfy only GSK3β + JNK3; or satisfy only GSK3β or JNK3 but not both. Table 10.4 (**right**) shows that a retrieval database that better satisfies the design criteria and better aligns with the controllable generation task generally leads to better performance. Nevertheless, we note that RetMol performs reasonably well even with exemplar molecules that only partially satisfy the properties (e.g., GSK3β + JNK3), achieving comparable performance to RationaleRL [137].

Size of retrieval database. Figure 10.4 (**left**) shows a larger retrieval database generally improves all metrics and reduces the variance. It is particularly interesting that our framework achieves strong performance with a small retrieval database, which already outperforms the best baseline on the success rate metric with only a 100-molecule retrieval database.

Number of optimization iterations. Figure 10.4 (**middle**) shows that RetMol outperforms the best existing methods with a small number of iterations (outperforms RationaleRL at 30 iterations and MARS at 60 iterations); its performance continues to improve with increasing iterations.

Dynamic retrieval database update. Figure 10.4 (**right**) shows that, for the penalized logP optimization experiment (Section 10.3.1), with dynamical update our framework generates more molecules with property values beyond the best in the data compared to the case without dynamic update. This comparison shows that dynamic update is crucial to generalizing beyond the retrieval database.

10.4 Related Work

RetMol is most related to controllable molecule generation methods, which we have briefly reviewed and compared with in Section 4.1. Another line of work in this direction leverages constraints based on *explicit*, pre-defined molecule scaffold structures to guide the generative process [220, 177]. Our approach is fundamentally different in that the retrieved exemplar molecules *implicitly* guides the generative process through the information fusion module.

RetMol is also inspired by a recent line of work that integrates a retrieval module in various NLP and vision tasks, such as language modeling [27, 198, 377], code generation [108], question answering [102, 403], and image generation [326, 36, 23, 45]. Among them, the retrieval mechanism is mainly used for improving the generation quality either with a smaller model [27, 23] or with very few data points [36, 45]. None of them has explicitly explored the *controllable* generation with retrieval. The retrieval module also appears in bio-related applications such as multiple sequence alignment (MSA), which can be seen as a way to search and retrieve relevant protein sequences, and has been an essential building block in MSA transformer [272] and AlphaFold [146]. However, the MSA methods focus on the pairwise interactions among a set of evolutionarily related (protein) sequences while RetMol considers the cross attention between the input and a set of retrieved examples.

There also exist priors work that study retrieval-based approaches for controllable text generation [155, 380]. These methods require task-specific training/fine-tuning of the generative model for each controllable generation task whereas our framework can be applied to many different tasks without it. While we focus on molecules, our framework is general and has the potential to achieve controllable generation for multiple other modalities beyond molecules.

10.5 Discussions

Applications that involve molecule generation such as drug discovery are high-stake in nature. These applications are highly regulated to prevent potential misuse [116]. RetMol as a technology to improve controllable molecule generation has the potential to be subjected to malicious use. For example, one could change the retrieval database and the design criteria into harmful ones, such as increased drug toxicity. However, we note that RetMol is a computational tool useful for *in silico* experiments. As a result, although RetMol can suggest new molecules according to arbitrary design criteria, the properties of the generated molecules are estimations of the real chemical and biological properties and need to be further validated in lab experiments. Thus, while RetMol's real-world impact is limited to *in silico* experiments, it is also prevented from directly generating real drugs that can be readily used. In addition, controllable molecule generation is an active area of research; we hope that our work contribute to this ongoing line of research and make ML methods safe and reliable for molecule generation applications in the real world.

10.6 Conclusions

We proposed RetMol, a new retrieval-based framework for controllable molecule generation. By incorporating a retrieval module with a pre-trained generative model, RetMol leverages exemplar molecules retrieved from a task-specific retrieval database to steer the generative model towards generating new molecules with the desired design criteria. RetMol is versatile, requires no task-specific fine-tuning and is agnostic to the generative models (see Appendix D.3.7). We demonstrated the effectiveness of RetMol on a variety of benchmark tasks and a real-world inhibitor design task for the SARS-CoV-2 virus, achieving state-of-the-art performances in each case comparing to existing methods. Since RetMol still requires exemplar molecules that at least partially satisfy the design criteria, it becomes challenging when those molecules are unavailable at all. A valuable future work is to improve the retrieval mechanisms such that even weaker molecules, i.e., those that do not satisfy but are close to satisfying the design criteria, can be leveraged to guide the generation process.

Chapter 11

Summary and Future Work

This thesis has developed machine-learning methods for personalized human learning at scale, addressing the need for customized learning content and learning analytics as well as for doing so in a manner that learners and instructors can trust. Under learning content personalization, I have proposed methods and obtained results for 1) generation, specifically for math word problems; 2) evaluation, specifically for factual quiz question generation from textbooks with today's large-scale language models; 3) representation, specifically for scientific and mathematical formulae; and 4) analyses, specifically for quiz and diagnostic questions. Under learning analytics personalization, I have proposed methods to track and analyze a learner's unique learning journey both scalably and flexibly. Under trustworthy machine learning, I have proposed methods to both understand and improve existing, popular algorithms underlying many innovative methods for personalized learning in the literature. The work presented in this thesis lays a solid foundation for further developing methodologies for high-quality, accessible, and trustworthy content customization and learning analytics in personalized learning.

The recent advances in AI and ML have made it clear that the future of education will never be the same. For example, the emergence of powerful generative models, such as chatGPT, will likely be able to solve students' homework questions and write essays for them, potentially rendering certain pedagogical practices and assessments widely use today obsolete. A promising avenue for future research is to identify the emerging challenges and opportunities in human learning, develop new AI systems to empower learners to flourish in new learning environments, and analyze these systems to ensure safe and interpretable interaction with learners. At the same time, by tackling problems in human learning, this research will also advance the state-of-the-art in fundamental AI and ML methodologies. Below, I outline a few opportunities for future research.

Complex, multi-modal, heterogeneous learning content customization and evaluation. Powerful generative models have the potential to dramatically change how learning content is being created, consumed, and shared. Realizing this goal requires innovation capable of organizing and producing learning content that humans can trust as much as human expert-produced content. My future research will vastly expand on my prior work and develop new methodologies for generating and curating a wide range of diverse and heterogeneous learning content that learners and instructors can confidently and safely use. One promising direction is **multi-modal educational content generation.** Today's diverse learning content involves modalities beyond text, including images, tables, and videos that provide rich additional contextual information. Incorporating these modalities into generative models will enable generating more varied questions with richer contexts for more types of learning content. For example, my lab will investigate visual question generation (VQG) to create practice questions from images and videos that describe scientific processes and illustrates relationships among multiple concepts. My lab will also develop new methods to adaptively generate distractors for multiple-choice questions (MCQs) tailored to each learner's knowledge level. This will effectively assess and help resolve each learner's unique misconceptions. Another promising direction is **automated quality assurance for educational content generation.** Generated content may contain various sources of errors, such as factual errors that can be misleading and render the content inappropriate to use in real-world learning scenarios. My lab will develop new methods to mitigate and eliminate factual errors in the generated content. These methods will also incorporate the course syllabus and other learning materials such that the generated content does not contain information beyond the current learning progress.

Closing the gap between generated content and learning outcomes. Generated learning content, even if it is high-quality in its own right, is meaningless if it fails to benefit learners and improve learning in the wild. It is thus critical to evaluate the educational utility of the generated content, which will help us understand humans' reception of the generated content and aid the design of more educationally valuable generation methods that incorporate human preferences. One promising direction is **new quality metrics learned from human preferences**. Evaluating generative models has remained an active area of research, which becomes more challenging in human learning applications because the metrics are unclear. Unlike generative models for applications such as translation, where established metrics exist, no metrics evaluate whether the generated content is high-quality or useful for human learning. My lab will develop novel algorithms to learn metrics that assess the quality and usefulness of the generated content directly from human preferences. These human-centric metrics provide a unified measure to identify high and low-quality generated content and to quantitatively evaluate the different generative models' performance. Another promising direction is **designing learning intervention with generated content**. My lab will investigate the use of the generated content to intervene in the learning progress effectively and to improve learning outcomes. The first step is to study the generated content's impact on learning in various carefully designed, real-world learning environments and compare it to traditional, expert-designed learning content. In doing so, My lab strives to bridge the gap in content generation, learner understanding, and trustworthy machine intelligence toward more effective human learning in the wild.

Bibliography

[1] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[2] D. A. Abduljabbar and N. Omar, "Exam questions classification based on Bloom's taxonomy cognitive level using classifiers combination," *J. Theor. Applied Inf. Technol.*, vol. 78, no. 3, p. 447, 2015.

[3] T. A. Ackerman, "Using multidimensional item response theory to understand what items and tests are measuring," *Applied Measurement in Education*, vol. 7, no. 4, pp. 255–278, 1994.

[4] O. O. Adesope *et al.*, "Rethinking the use of tests: A meta-analysis of practice testing," *Review of Educational Research*, vol. 87, no. 3, pp. 659–701, 2017.

[5] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, "code2vec: Learning distributed representations of code," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, pp. 1–29, 2019.

[6] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, "MathQA: Towards interpretable math word problem solving with operation-based formalisms," in *Proc. NAACL*, Jun. 2019, pp. 2357–2367.

[7] S. An, J. Kim, M. Kim, and J. Park, "No task left behind: Multi-task learning of knowledge tracing and option tracing for better student assessment," *Proc. AAAI*, vol. 36, no. 4, pp. 4424–4431, Jun. 2022. [Online]. Available: https://doi.org/10.1609/aaai.v36i4.20364

[8] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier, "Cognitive tutors: Lessons learned," *Journal of the Learning Sciences*, vol. 4, no. 2, pp. 167–207, 1995.

[9] J. R. Anderson and R. Jeffries, "Novice lisp errors: Undetected losses of information from working memory," *Human–Computer Interact.*, vol. 1, no. 2, pp. 107–131, 1985.

[10] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 48, Jun. 2016, pp. 1120–1128.

[11] S. H. Bach, B. He, A. Ratner, and C. Ré, "Learning the structure of generative models without labeled data," in *Proc. Int. Conf. Mach. Learn.*, 2017, p. 273–282.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[13] D. Bajusz, A. Rácz, and K. Héberger, "Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations?" *Journal of Cheminformatics*, vol. 7, no. 1, May 2015. [Online]. Available: https://doi.org/10.1186/s13321-015-0069-3

[14] E. L. Baker, "Developing comprehensive assessments of higher order thinking," *Assessing Higher Order Thinking in Math.*, vol. 7, p. 20, 1990.

[15] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *Journal of Educational Data Mining*, vol. 1, no. 1, pp. 3–17, 2009.

[16] R. Balestriero and R. G. Baraniuk, "From Hard to Soft: Understanding Deep Network Nonlinearities via Vector Quantization and Statistical Inference," *ArXiv e-prints*, vol. 1810.09274, Oct 2018.

[17] R. Balestriero and R. G. Baraniuk, "A spline theory of deep networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 80, Jul 2018, pp. 374–383.

[18] W. Bao, "Covid-19 and online teaching in higher education: A case study of peking university," *Human Behavior and Emerging Technologies*, vol. 2, no. 2, pp. 113–115, 2020.

[19] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv:1308.3432*, 2013.

[20] G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan, and A. L. Hopkins, "Quantifying the chemical beauty of drugs," *Nature Chemistry*, vol. 4, no. 2, pp. 90–98, Jan. 2012. [Online]. Available: https://doi.org/10.1038/nchem.1243

[21] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[22] J. Blanchette, "Questions in the online learning environment," *Journal of distance education*, vol. 16, no. 2, pp. 37–57, 2001.

[23] A. Blattmann, R. Rombach, K. Oktay, and B. Ommer, "Retrieval-augmented diffusion models," *arXiv preprint arXiv:2204.11824*, 2022.

[24] B. S. Bloom, M. D. Engelhart, E. Furst, W. H. Hill, and D. R. Krathwohl, "Handbook i: cognitive domain," *New York: David McKay*, 1956.

[25] B. Bloom, M. Engelhart, E. Furst, W. Hill, , and D. Krathwohl, *Taxonomy of educational objectives: the classification of educational goals. Handbook 1: cognitive domain*. New York: David McKay, 1956.

[26] P. Blumberg, "Maximizing learning through course alignment and experience with different types of knowledge," *Innovative Higher Edu.*, vol. 34, no. 2, pp. 93–103, 2009.

[27] S. Borgeaud *et al.*, "Improving language models by retrieving from trillions of tokens," *arXiv e-prints*, p. arXiv:2112.04426, Dec. 2021.

[28] J. S. Brown and R. R. Burton, "Diagnostic models for procedural bugs in basic mathematical skills," *Cogn. sci.*, vol. 2, no. 2, pp. 155–192, 1978.

[29] N. Brown, M. Fiscato, M. H. Segler, and A. C. Vaucher, "GuacaMol: Benchmarking models for de novo molecular design," *Journal of Chemical Information and Modeling*, vol. 59, no. 3, pp. 1096–1108, Mar. 2019. [Online]. Available: https://doi.org/10.1021/acs.jcim.8b00839

[30] T. Brown *et al.*, "Language models are few-shot learners," in *Adv. Neural Inf. Process. Syst.*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, 2020, pp. 1877–1901.

[31] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[32] C. Buckley, "trec_eval," https://github.com/usnistgov/trec_eval, 2008.

[33] C. Buckley and E. M. Voorhees, "Retrieval evaluation with incomplete information," in *Prof. Int. ACM SIGIR Conf. Res. Develop. Info. Retrieval*, 2004, p. 25–32.

[34] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," in *Eur. Signal Process. Conf. (EUSIPCO)*, Aug 2017, pp. 1744–1748.

[35] A. Callahan, J. A. Fries, C. Ré, J. I. Huddleston, N. J. Giori, S. Delp, and N. H. Shah, "Medical device surveillance with electronic health records," *npj Digit. Med.*, vol. 2, no. 1, p. 94, Sep 2019.

[36] A. Casanova, M. Careil, J. Verbeek, M. Drozdzal, and A. Romero Soriano, "Instance-conditioned gan," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 27 517–27 529. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/e7ac288b0f2d41445904d071ba37aaff-Paper.pdf

[37] H. Cen, K. Koedinger, and B. Junker, "Learning factors analysis–A general method for cognitive model evaluation and improvement," in *Proc. Int. Conf. Intell. Tutoring Syst.*, 2006, pp. 164–175.

[38] R. P. Chalmers *et al.*, "mirt: A multidimensional item response theory package for the r environment," *Journal of Statistical Software*, vol. 48, no. 6, pp. 1–29, 2012.

[39] H.-H. Chang, Z. Ying *et al.*, "Nonlinear sequential designs for logistic item response theory models with applications to computerized adaptive tests," *The Annals of Statistics*, vol. 37, no. 3, pp. 1466–1488, 2009.

[40] B. Chen*, T. Wang*, C. Li, H. Dai, and L. Song, "Molecule optimization by explainable evolution," in *International Conference on Learning Representations*, 2021.

[41] H. Chen, "Can generative-model-based drug design become a new normal in drug discovery?" *Journal of Medicinal Chemistry*, vol. 65, no. 1, pp. 100–102, Dec. 2021. [Online]. Available: https://doi.org/10.1021/acs.jmedchem.1c02042

[42] L. Chen, S. Dai, C. Tao, D. Shen, Z. Gan, H. Zhang, Y. Zhang, R. Zhang, G. Wang, and L. Carin, "Adversarial text generation via feature-mover's distance," in *Proc. NeurIPS*, 2018, p. 4671–4682.

[43] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[44] T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Proc. Advances in Neural Information Processing Systems*, 2018, pp. 2610–2620.

[45] W. Chen, H. Hu, C. Saharia, and W. W. Cohen, "Re-imagen: Retrieval-augmented text-to-image generator," *arXiv preprint arXiv:2209.14491*, 2022.

[46] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick, "Microsoft COCO Captions: Data Collection and Evaluation Server," *arXiv:1504.00325*, Apr. 2015.

[47] X. Chen, C. Liang, A. W. Yu, D. Zhou, D. Song, and Q. V. Le, "Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension," in *Proc. ICLR*, 2020.

[48] X. Chen, C. Liu, and D. Song, "Tree-to-tree neural networks for program translation," in *Proc. Int. Conf. Neural Info. Process. Syst.*, 2018, p. 2552–2562.

[49] Y. Chen and M. de Rijke, "A collective variational autoencoder for top-n recommendation with side information," in *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*, Oct. 2018, pp. 3–9.

[50] V. Chenthamarakshan, P. Das, S. C. Hoffman, H. Strobelt, I. Padhi, K. W. Lim, B. Hoover, M. Manica, J. Born, T. Laino, and A. Mojsilovic, "Cogmol: Target-specific and selective drug design for covid-19 using deep generative models," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[51] M. Chi, K. R. Koedinger, G. J. Gordon, P. Jordon, and K. VanLahn, "Instructional factors analysis: A cognitive model for multiple instructional interventions," 2011.

[52] T.-R. Chiang and Y.-N. Chen, "Semantically-aligned equation generation for solving and reasoning math word problems," in *Proc. NAACL*, Jun. 2019, pp. 2656–2668.

[53] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Oct. 2014, pp. 1724–1734.

[54] B. Choffin, F. Popineau, Y. Bourda, and J.-J. Vie, "DAS3H: Modeling student learning and forgetting for pptimally scheduling distributed practice of skills," in *Proc. Int. Conf. Educ. Data Mining*, 2019, pp. 29–38.

[55] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim, and J. Heo, "Ednet: A large-scale hierarchical dataset in education," in *Int. Conf. Artif. Intell. Educ.* Springer, 2020, pp. 69–73.

[56] K. Chrysafiadi and M. Virvou, "Student modeling approaches: A literature review for the last decade," *Expert Systems with Applications*, vol. 40, no. 11, pp. 4715–4729, 2013.

[57] K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[58] P. A. Connor-Greene, "Assessing and promoting student learning: Blurring the line between teaching and testing," *Teaching Psychol.*, vol. 27, no. 2, pp. 84–88, 2000.

[59] T. Cooijmans, N. Ballas, C. Laurent, C. Gulcehre, and A. C. Courville, "Recurrent batch normalization," in *Proc. Int. Conf. Learn. Representations (ICLR)*, Apr. 2017.

[60] A. Corbett and J. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Model. User-adapted Interact.*, vol. 4, no. 4, pp. 253–278, Dec. 1994.

[61] E. Creager, D. Madras, J.-H. Jacobsen, M. Weis, K. Swersky, T. Pitassi, and R. Zemel, "Flexibly fair representation learning by disentanglement," in *Proc. Int. Conf. Mach. Learn.*, ser. Proc. Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 1436–1445.

[62] A. Crowe, C. Dirks, and M. P. Wenderoth, "Biology in Bloom: implementing bloom's taxonomy to enhance student learning in biology," *CBE—Life Sciences Edu.*, vol. 7, no. 4, pp. 368–381, 2008.

[63] A. Dalke, J. Hert, and C. Kramer, "mmpdb: An open-source matched molecular pair platform for large multiproperty data sets," *Journal of Chemical Information and Modeling*, vol. 58, no. 5, pp. 902–910, May 2018. [Online]. Available: https://doi.org/10.1021/acs.jcim.8b00173

[64] P. Das, T. Sercu, K. Wadhawan, I. Padhi, S. Gehrmann, F. Cipcigan, V. Chenthamarakshan, H. Strobelt, C. dos Santos, P.-Y. Chen, Y. Y. Yang, J. P. K. Tan, J. Hedrick, J. Crain, and A. Mojsilovic, "Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations," *Nature Biomedical Engineering*, vol. 5, no. 6, pp. 613–623, Mar. 2021. [Online]. Available: https://doi.org/10.1038/s41551-021-00689-x

[65] D. Datta, M. Phillips, J. Chiu, G. S. Watson, J. P. Bywater, L. Barnes, and D. Brown, "Improving classification through Weak supervision in context-specific conversational agent development for teacher education," *arXiv e-prints*, Oct. 2020.

[66] K. Davila and R. Zanibbi, "Layout and semantics: Combining representations for mathematical formula search," in *Prof. Int. ACM SIGIR Conf. Res. Develop. Info. Retrieval*, 2017, p. 1165–1168.

[67] N. De Freitas, P. Højen-Sørensen, M. I. Jordan, and S. Russell, "Variational mcmc," in *Proc. conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 120–127.

[68] P. Deane and K. Sheehan, "Automatic item generation via frame semantics: Natural language generation of math word problems." ERIC, Tech. Rep., 2003.

[69] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, "Image-to-markup generation with coarse-to-fine attention," in *Proc. Int. Conf. Mach. Learn.*, 2017, p. 980–989.

[70] A. Desai, S. Gulwani, V. Hingorani, N. Jain, A. Karkare, M. Marron, and S. Roy, "Program synthesis using natural language," in *Proc. 38th Int. Conf. Softw. Eng.*, 2016, pp. 345–356.

[71] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[72] S. Diab and B. Sartawi, "Classification of questions and learning outcome statements (los) into Blooms taxonomy (bt) by similarity measurements towards extracting of learning outcome from learning material," *arXiv preprint*, 2017.

[73] A. Dieng, R. Ranganath, J. Altosaar, and D. Blei, "Noisin: Unbiased regularization for recurrent neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 80, Jul. 2018, pp. 1252–1261.

[74] M. P. Driscoll, *Psychology of learning for instruction.* Allyn & Bacon, 1994.

[75] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," in *Proc. ACL*, Jul. 2017, pp. 1342–1352.

[76] N. Duan, D. Tang, P. Chen, and M. Zhou, "Question generation for question answering," in *Proc. Conference on EMNLP*, Sep. 2017, pp. 866–874.

[77] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, pp. 179–211, 1990.

[78] P. Ertl and A. Schuffenhauer, "Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions," *Journal of Cheminformatics*, vol. 1, no. 1, Jun. 2009. [Online]. Available: https://doi.org/10.1186/1758-2946-1-8

[79] M. Q. Feldman, J. Y. Cho, M. Ong, S. Gulwani, Z. Popović, and E. Andersen, "Automatic diagnosis of students' misconceptions in K-8 mathematics," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2018, pp. 1–12.

[80] J. Feng, B. Zhang, Y. Li, and Q. Xu, "Bayesian diagnosis tracing: Application of procedural misconceptions in knowledge tracing," in *Proc. Int. Conf. Artif. Intell. Educ.* Springer, 2019, pp. 84–88.

[81] J.-P. Fox, *Bayesian item response modeling: Theory and applications.* Springer Science & Business Media, 2010.

[82] J.-P. Fox and C. A. Glas, "Bayesian estimation of a multilevel irt model using gibbs sampling," *Psychometrika*, vol. 66, no. 2, pp. 271–288, 2001.

[83] M. Freitag and Y. Al-Onaizan, "Beam search strategies for neural machine translation," in *Proc. First Workshop Neural Mach. Transl.*, Aug. 2017, pp. 56–60.

[84] J. A. Fries, E. Steinberg, S. Khattar, S. L. Fleming, J. Posada, A. Callahan, and N. H. Shah, "Trove: Ontology-driven weak supervision for medical entity classification," *arXiv e-prints*, Aug. 2020.

[85] J. A. Fries, P. Varma, V. S. Chen, K. Xiao, H. Tejeda, P. Saha, J. Dunnmon, H. Chubb, S. Maskatia, M. Fiterau, S. Delp, E. Ashley, C. Ré, and J. R. Priest, "Weakly supervised classification of aortic valve malformations using unlabeled cardiac MRI sequences," *Nature Commun.*, vol. 10, no. 1, p. 3111, Jul 2019.

[86] L. Gao, Z. Jiang, Y. Yin, K. Yuan, Z. Yan, and Z. Tang, "Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval: can mathematical formulae be embedded like a natural language?" *arXiv e-prints*, Jul. 2017.

[87] W. Gao, R. Mercado, and C. W. Coley, “Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design,” in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=FRxhHdnxt1

[88] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte, M. Davies, N. Dedman, A. Karlsson, M. P. Magariños, J. P. Overington, G. Papadatos, I. Smit, and A. R. Leach, “The ChEMBL database in 2017,” *Nucleic Acids Research*, vol. 45, no. D1, pp. D945–D954, Nov. 2016. [Online]. Available: https://doi.org/10.1093/nar/gkw1074

[89] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[90] T. George and S. Merugu, “A scalable collaborative filtering framework based on co-clustering,” in *Proc. IEEE International Conference on Data Mining*. IEEE Computer Society, 2005, p. 625–628.

[91] Z. Ghahramani and M. J. Beal, “Variational inference for bayesian mixtures of factor analysers,” in *Adv. Neural Inf. Process. Syst.*, 2000, pp. 449–455.

[92] A. Ghosh, N. Heffernan, and A. S. Lan, “Context-aware attentive knowledge tracing,” in *Proc. ACM SIGKDD*, 2020, pp. 2330–2339.

[93] A. Ghosh, J. Raspat, and A. Lan, “Option tracing: Beyond correctness analysis in knowledge tracing,” in *Int. Conf. Artif. Intell. Educ.* Springer, 2021, pp. 137–149.

[94] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC, 1995.

[95] X. Glorot, B. A., and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proc. Int. Conf. Artificial Intell. and Statist. (AISTATS)*, vol. 15, Apr. 2011, pp. 315–323.

[96] Y. Goldberg and G. Hirst, *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[97] W. Gong, S. Tschiatschek, R. Turner, S. Nowozin, and J. M. Hernández-Lobato, “Icebreaker: Element-wise active information acquisition with bayesian deep latent gaussian model,” in *Proc. Advances Neural Information Processing Systems*, 2019.

[98] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[99] S. Graf and Kinshuk, *Personalized Learning Systems*. Boston, MA: Springer US, 2012, pp. 2594–2596. [Online]. Available: https://doi.org/10.1007/978-1-4419-1428-6_152

[100] A. Graves, A. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process. (ICASSP)*, May 2013, pp. 6645–6649.

[101] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar, “Accelerating large-scale inference with anisotropic vector quantization,” in *International Conference on Machine Learning*, 2020. [Online]. Available: https://arxiv.org/abs/1908.10396

[102] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, “Retrieval augmented language model pre-training,” in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 13–18 Jul 2020, pp. 3929–3938.

[103] R. Habib and D. Barber, “Auxiliary variational mcmc,” *Proc. Int. Conf. Learn Representations*, 2019.

[104] B. Hammer, “On the approximation capability of recurrent neural networks,” *Neurocomputing*, vol. 31, no. 1, pp. 107–123, Mar. 2000.

[105] L. A. Hannah and D. B. Dunson, “Multivariate convex regression with adaptive partitioning,” *J. Mach. Learn. Res.*, vol. 14, pp. 3261–3294, 2013.

[106] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. New York, NY, USA: Springer New York Inc., 2001.

[107] ——, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[108] S. A. Hayati, R. Olivier, P. Avvaru, P. Yin, A. Tomasic, and G. Neubig, "Retrieval-based neural code generation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 925–930. [Online]. Available: https://aclanthology.org/D18-1111

[109] N. T. Heffernan and C. Heffernan, "The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching," *Int. J. Artif. Intell. Educ.*, vol. 24, pp. 470–497, 2014.

[110] K. Helfrich, D. Willmott, and Q. Ye, "Orthogonal recurrent neural networks with scaled Cayley transform," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 80, Jul. 2018, pp. 1969–1978.

[111] M. Henaff, A. Szlam, and Y. LeCun, "Recurrent orthogonal networks and long-memory tasks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 48, Jun. 2016, pp. 2034–2042.

[112] D. Hendrycks, S. Basart, S. Kadavath, M. Mazeika, A. Arora, E. Guo, C. Burns, S. Puranik, H. He, D. Song *et al.*, "Measuring coding challenge competence with apps," in *Thirty-fifth Conf. Neural Inf. Process. Syst. Datasets and Benchmarks Track (Round 2)*, 2021.

[113] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the math dataset," in *Proc. NeurIPS*, 2021.

[114] J. Herrington and R. Oliver, "Using situated learning and multimedia to investigate higher-order thinking," *J. Interactive Learn. Res.*, vol. 10, no. 1, pp. 3–24, 1999.

[115] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *Proc. International Conference on Learning Representations*, 2017.

[116] R. G. Hill and D. Richards, *Drug discovery and development: technology in transition*, 2022.

[117] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[118] S. C. Hoffman, V. Chenthamarakshan, K. Wadhawan, P.-Y. Chen, and P. Das, "Optimizing molecules using efficient queries from property evaluations," *Nature Machine Intelligence*, vol. 4, no. 1, pp. 21–31, Dec. 2021. [Online]. Available: https://doi.org/10.1038/s42256-021-00422-y

[119] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, "Learning to solve arithmetic word problems with verb categorization," in *Proc. EMNLP*, Oct. 2014, pp. 523–533.

[120] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proc. ICML*, 2017, p. 1587–1596.

[121] D. Huang, J. Liu, C.-Y. Lin, and J. Yin, "Neural math word problem solver with reinforcement learning," in *Proc. ACL*, Aug. 2018, pp. 213–223.

[122] D. Huang, S. Shi, C.-Y. Lin, J. Yin, and W.-Y. Ma, "How well do computers solve math word problems? large-scale dataset construction and evaluation," in *Proc. ACL*, Aug. 2016, pp. 887–896.

[123] Y.-T. Huang, M. C. Chen, and Y. S. Sun, "Bringing personalized learning into computer-aided question generation," 2018.

[124] T. Huynh, H. Wang, and B. Luan, "*In Silico* exploration of the molecular mechanism of clinically oriented drugs for possibly inhibiting SARS-CoV-2's main protease," *The Journal of Physical Chemistry Letters*, vol. 11, no. 11, pp. 4413–4420, May 2020. [Online]. Available: https://doi.org/10.1021/acs.jpclett.0c00994

[125] S. L. Hyland and G. Rätsch, "Learning unitary operators with help from u (n)." in *Proc. AAAI conf. Artificial Intell.*, Feb. 2017, pp. 2050–2058.

[126] J. J. Irwin and B. K. Shoichet, "ZINC - a free database of commercially available compounds for virtual screening," *Journal of Chemical Information and Modeling*, vol. 45, no. 1, pp. 177–182, Dec. 2004. [Online]. Available: https://doi.org/10.1021/ci049714+

[127] R. Irwin, S. Dimitriadis, J. He, and E. J. Bjerrum, "Chemformer: a pre-trained transformer for computational chemistry," *Machine Learning: Science and Technology*, vol. 3, no. 1, p. 015022, jan 2022. [Online]. Available: https://doi.org/10.1088/2632-2153/ac3ffb

[128] H. Ishwaran, J. S. Rao *et al.*, "Spike and slab variable selection: frequentist and bayesian strategies," *The Annals of Statistics*, vol. 33, no. 2, pp. 730–773, 2005.

[129] S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, "Mapping language to code in programmatic context," *arXiv preprint arXiv:1808.09588*, 2018.

[130] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. ICLR*, 2017.

[131] K. Jayakodi, M. Bandara, and I. Perera, "An automatic classifier for exam questions in engineering: A process for Bloom's taxonomy," in *IEEE Int. Conf. Teaching Assessment Learn. Eng.*, 2015, pp. 195–202.

[132] J. H. Jensen, "A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space," *Chemical Science*, vol. 10, no. 12, pp. 3567–3572, 2019. [Online]. Available: https://doi.org/10.1039/c8sc05372c

[133] J. L. W. V. Jensen *et al.*, "Sur les fonctions convexes et les inégalités entre les valeurs moyennes," *Acta Mathematica*, vol. 30, pp. 175–193, 1906.

[134] A. H. Jha, S. Anand, M. Singh, and V. Veeravasarapu, "Disentangling factors of variation with cycle-consistent variational auto-encoders," in *Proc. ECCV*, September 2018.

[135] Z. Jiao and F. Ren, "WRGAN: Improvement of RelGAN with wasserstein loss for text generation," *Electronics*, vol. 10, no. 3, p. 275, Jan. 2021.

[136] J. Jiménez-Luna, F. Grisoni, N. Weskamp, and G. Schneider, "Artificial intelligence in drug discovery: recent advances and future perspectives," *Expert Opinion on Drug Discovery*, vol. 16, no. 9, pp. 949–959, Apr. 2021. [Online]. Available: https://doi.org/10.1080/17460441.2021.1909567

[137] W. Jin, D. Barzilay, and T. Jaakkola, "Multi-objective molecule generation using interpretable substructures," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4849–4859.

[138] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 2323–2332.

[139] W. Jin, R. Barzilay, and T. Jaakkola, "Hierarchical Graph-to-Graph Translation for Molecules," *arXiv e-prints*, p. arXiv:1907.11223, Jun. 2019.

[140] W. Jin, K. Yang, R. Barzilay, and T. Jaakkola, “Learning multimodal graph-to-graph translation for molecule optimization,” in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1xJAsA5F7

[141] Z. Jin *et al.*, “Structure of mpro from SARS-CoV-2 and discovery of its inhibitors,” *Nature*, vol. 582, no. 7811, pp. 289–293, Apr. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2223-y

[142] L. Jing, Y. Shen, T. Dubcek, J. Peurifoy, S. Skirlo, Y. LeCun, M. Tegmark, and M. Soljačić, “Tunable efficient unitary neural networks (EUNN) and their application to RNNs,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 70, Aug. 2017, pp. 1733–1741.

[143] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[144] K. S. Jones, “A statistical interpretation of term specificity and its application in retrieval,” *J. Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[145] C. Jose, M. Cisse, and F. Fleuret, “Kronecker recurrent units,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, Apr. 2018.

[146] J. Jumper *et al.*, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. 7873, pp. 583–589, Jul. 2021.

[147] G. Karamanolakis, K. R. Cherian, A. R. Narayan, J. Yuan, D. Tang, and T. Jebara, “Item recommendation with variational autoencoders and heterogeneous priors,” in *Workshop on Deep Learning for Recommender Systems*, Oct. 2018, pp. 10–14.

[148] S. Karamustafaoğlu, S. Sevim, O. Karamustafaoğlu, and S. Cepni, “Analysis of turkish high-school chemistry-examination questions according to bloom’s taxonomy,” *Chemistry Edu. Res. and Pract.*, vol. 4, no. 1, pp. 25–30, 2003.

[149] J. D. Karpicke, “Retrieval-based learning: Active retrieval promotes meaningful learning,” *Current Directions Psychol. Sci.*, vol. 21, no. 3, pp. 157–163, May 2012.

[150] J. D. Karpicke and H. L. Roediger, “The critical importance of retrieval for learning,” *Science*, vol. 319, no. 5865, pp. 966–968, 2008.

[151] J. D. Karpicke and J. R. Blunt, “Retrieval practice produces more learning than elaborative studying with concept mapping,” *Science*, vol. 331, no. 6018, pp. 772–775, 2011.

[152] N. S. Keskar, B. McCann, L. Varshney, C. Xiong, and R. Socher, “CTRL - A Conditional Transformer Language Model for Controllable Generation,” *arXiv:1909.05858*, 2019.

[153] M. Khajah, Y. Huang, J. González-Brenes, M. Mozer, and P. Brusilovsky, “Integrating knowledge tracing and item response theory: A tale of two frameworks,” in *Proc. Int. Workshop Personalization Approaches Learn. Environ.*, vol. 1181, 2014, pp. 7–15.

[154] H. Kim and A. Mnih, “Disentangling by factorising,” in *Proc. International Conference on Machine Learning*, vol. 80, 2018, pp. 2649–2658.

[155] J. Kim, S. Choi, R. K. Amplayo, and S.-w. Hwang, “Retrieval-augmented controllable review generation,” in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020.

[156] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Representations*, 2015.

[157] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *Proc. Int. Conf. Learn. Representations*, 2013.

[158] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[159] K. R. Koedinger, J. Kim, J. Z. Jia, E. A. McLaughlin, and N. L. Bier, "Learning is not a spectator sport: Doing is better than watching for learning from a mooc," in *Proc. Conf. Learn. Scale*, 2015, pp. 111–120.

[160] P. Koehn, "Pharaoh: A beam search decoder for phrase-based statistical machine translation models," in *Mach. Transl.: Real Users Res.*, R. E. Frederking and K. B. Taylor, Eds., 2004, pp. 115–124.

[161] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi, "MAWPS: A math word problem repository," in *Proc. NAACL*, Jun. 2016, pp. 1152–1157.

[162] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. SIGKDD Conference on Knowledge Discovery and Data Mining*, 2008, p. 426–434.

[163] G. Kovacs, "Effects of in-video quizzes on mooc lecture viewing," in *Proc. Conf. Learn. Scale*, 2016, pp. 31–40.

[164] D. R. Krathwohl, "A Revision of Bloom's Taxonomy : An Overview," *Theory into Pract.*, vol. 41, no. 4, pp. 212–218, 2002.

[165] D. Krueger, T. Maharaj, J. Kramar, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, H. Larochelle, A. C. Courville, and C. Pal, "Zoneout: Regularizing rnns by randomly preserving hidden activations," in *Proc. Int. Conf. Learn. Representations (ICLR)*, Apr. 2017.

[166] O. Kuchaiev and B. Ginsburg, "Training Deep AutoEncoders for Collaborative Filtering," *arXiv preprint arXiv:1708.01715*, Aug. 2017.

[167] V. Kuleshov, J. Ding, C. Vo, B. Hancock, A. Ratner, Y. Li, C. Ré, S. Batzoglou, and M. Snyder, "A machine-compiled database of genome-wide association studies," *Nature Commun.*, vol. 10, no. 1, p. 3341, Jul 2019.

[168] K. Kunen, *Set theory - an introduction to independence proofs*. North-Holland, 1983, vol. 102.

[169] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay, "Learning to automatically solve algebra word problems," in *Proc. ACL*, Jun. 2014, pp. 271–281.

[170] M. J. Kusner and J. M. Hernández-Lobato, "Gans for sequences of discrete elements with the gumbel-softmax distribution," *arXiv:1611.04051*, 2016.

[171] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, "Grammar variational autoencoder," in *Proc. Int. Conf. Mach. Learn.*, vol. 70, 06–11 Aug 2017, pp. 1945–1954.

[172] G. Lample and F. Charton, "Deep learning for symbolic mathematics," in *Proc. Int. Conf. Learn. Representations*, 2020.

[173] A. S. Lan, C. Studer, and R. G. Baraniuk, "Time-varying learning and content analytics via sparse factor analysis," in *Proc. SIGKDD Conference on Knowledge Discovery and Data Mining*, Aug. 2014, pp. 452–461.

[174] A. S. Lan, D. Vats, A. E. Waters, and R. G. Baraniuk, "Mathematical language processing: Automatic grading and feedback for open response mathematical questions," in *Proc. ACM Conf. Learn. Scale*, 2015, pp. 167–176.

[175] A. S. Lan, A. E. Waters, C. Studer, and R. G. Baraniuk, "Sparse factor analysis for learning and content analytics," *Journal of Machine Learning Research*, vol. 15, pp. 1959–2008, 2014.

[176] T. Lancaster and C. Cotarlan, "Contract cheating by stem students through a file sharing website: a covid-19 pandemic perspective," *Int. J. Educ. Integrity*, vol. 17, no. 1, p. 3, Feb 2021.

[177] M. Langevin, H. Minoux, M. Levesque, and M. Bianciotto, "Scaffold-constrained molecular generation," *Journal of Chemical Information and Modeling*, vol. 60, no. 12, pp. 5637–5646, 2020, pMID: 33301333. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c01015

[178] A. Lavie and A. Agarwal, "Meteor: An automatic metric for MT evaluation with high levels of correlation with human judgments," in *Proc. Workshop Statistical Mach. Transl.*, Jun. 2007, pp. 228–231.

[179] Q. V. Le, N. Jaitly, and G. E. Hinton, "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units," *ArXiv e-prints*, vol. 1504.00941, Apr. 2015.

[180] A. LeClair and C. McMillan, "Recommendations for datasets for source code summarization," *arXiv preprint arXiv:1904.02660*, 2019.

[181] S. D. Levitt and M.-J. Lin, "Catching cheating students," National Bureau of Economic Research, Tech. Rep., 2015.

[182] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

[183] J. Li, X. Chen, E. Hovy, and D. Jurafsky, "Visualizing and understanding neural models in NLP," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Language Technol. (NAACL HLT)*, Jun. 2016, pp. 681–691.

[184] J. Li, L. Wang, J. Zhang, Y. Wang, B. T. Dai, and D. Zhang, "Modeling intra-relation in math word problems with different functional multi-head attentions," in *Proc. ACL*, Jul. 2019, pp. 6162–6167.

[185] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Human Lang. Technol.*, Jun. 2016, pp. 110–119.

[186] S. Li, L. Wu, S. Feng, F. Xu, F. Xu, and S. Zhong, "Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem," in *Proc. EMNLP*, Nov. 2020, pp. 2841–2852.

[187] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. ACL*, Aug. 2021, pp. 4582–4597.

[188] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in *Proc. SIGKDD Conference on Knowledge Discovery and Data Mining*, Aug. 2017, pp. 305–314.

[189] Y. Li, N. Duan, B. Zhou, X. Chu, W. Ouyang, and X. Wang, "Visual Question Generation as Dual Task of Visual Question Answering," *ArXiv e-prints*, Sep. 2017.

[190] Y. Li, L. Zhang, and Z. Liu, "Multi-objective de novo drug design with conditional graph generative model," *Journal of Cheminformatics*, vol. 10, no. 1, Jul. 2018. [Online]. Available: https://doi.org/10.1186/s13321-018-0287-6

[191] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago *et al.*, "Competition-level code generation with alphacode," *arXiv preprint arXiv:2203.07814*, 2022.

[192] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. Int. Conf. World Wide Web*, 2018, pp. 689–698.

[193] J. Lim, S. Ryu, J. W. Kim, and W. Y. Kim, "Molecular generative model based on conditional variational autoencoder for de novo molecular design," *Journal of Cheminformatics*, vol. 10, no. 1, Jul. 2018. [Online]. Available: https://doi.org/10.1186/s13321-018-0286-7

[194] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Workshop Text Summarization Branches Out*, Jul. 2004, pp. 74–81.

[195] R. V. Lindsey, J. D. Shroyer, H. Pashler, and M. C. Mozer, "Improving students' long-term knowledge retention through personalized review," *Psychological Science*, vol. 25, no. 3, pp. 639–647, 2014.

[196] P. Liu *et al.*, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," 2021.

[197] Q. Liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu, "Ekt: Exercise-aware knowledge tracing for student performance prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 1, pp. 100–115, 2019.

[198] Q. Liu, D. Yogatama, and P. Blunsom, "Relational Memory Augmented Language Models," *arXiv e-prints*, p. arXiv:2201.09680, Jan. 2022.

[199] T. Liu, Y. Lin, X. Wen, R. N. Jorissen, and M. K. Gilson, "BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities," *Nucleic Acids Research*, vol. 35, no. Database, pp. D198–D201, Jan. 2007. [Online]. Available: https://doi.org/10.1093/nar/gkl999

[200] T. Liu, Q. Fang, W. Ding, and Z. Liu, "Mathematical word problem generation from commonsense knowledge graph and equations," *arXiv:2010.06196*, 2020.

[201] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," 2021.

[202] O. H. Lu, A. Y. Huang, D. C. Tsai, and S. J. Yang, "Expert-authored and machine-generated short-answer questions for assessing students learning performance," *Educational Technology & Society*, vol. 24, no. 3, pp. 159–173, 2021.

[203] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang *et al.*, "Codexglue: A machine learning benchmark dataset for code understanding and generation," *arXiv preprint arXiv:2102.04664*, 2021.

[204] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.

[205] C. Ma, S. Tschiatschek, K. Palla, J. M. Hernandez-Lobato, S. Nowozin, and C. Zhang, "EDDI: Efficient dynamic discovery of high-value information with partial VAE," in *Proc. Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 4234–4243.

[206] C. Ma, W. Gong, J. M. Hernández-Lobato, N. Koenigstein, S. Nowozin, and C. Zhang, "Partial vae for hybrid recommender system," in *Proc. Conf. Neural Inf. Process. Syst. workshop Bayesian Deep Learn.*, Dec. 2018.

[207] Y.-A. Ma, T. Chen, and E. Fox, "A complete recipe for stochastic gradient mcmc," in *Adv. Neural Inf. Process. Syst.*, 2015, pp. 2917–2925.

[208] D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[209] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proc. ICLR*, 2017.

[210] A. Magnani and S. P. Boyd, "Convex piecewise-linear fitting," *Optimization Eng.*, vol. 10, no. 1, pp. 1–17, Mar. 2009.

[211] B. Mansouri, S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, and R. Zanibbi, "Tangent-cft: An embedding model for mathematical formulas," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Info. Retrieval*, 2019, p. 11–18.

[212] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," in *Proc. Int. Conf. Learn. Representations (ICLR)*, May 2015.

[213] Y. Mao, Y. Shi, S. Marwan, T. W. Price, T. Barnes, and M. Chi, "Knowing" when" and" where": Temporal-astnn for student learning progression in novice programming tasks." *Int. Educ. Data Mining Soc.*, 2021.

[214] P. Maragakis, H. Nisonoff, B. Cole, and D. E. Shaw, "A deep-learning view of chemical space designed to facilitate drug discovery," *Journal of Chemical Information and Modeling*, vol. 60, no. 10, pp. 4487–4496, Jul. 2020. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c00321

[215] C. H. Martin and M. W. Mahoney, "Implicit Self-Regularization in Deep Neural Networks: Evidence from Random Matrix Theory and Implications for Learning," *ArXiv e-prints*, vol. 1810.01075, Oct 2018.

[216] L. Martin, C. Mills, S. K. D'Mello, and E. F. Risko, "Re-watching lectures as a study strategy and its effect on mind wandering," *Experimental Psychology*, 2018.

[217] P. Mateusz and A. Nikolaus, "Efficient computation of the tree edit distance," *ACM Trans. Database Syst.*, vol. 40, no. 1, Mar. 2015.

[218] E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh, "Disentangling disentanglement in variational autoencoders," in *Proc. International Conference on Machine Learning*, Jun 2019, pp. 4402–4412.

[219] P.-A. Mattei and J. Frellsen, "missiwae: Deep generative modelling and imputation of incomplete data," *arXiv preprint arXiv:1812.02633*, 2018.

[220] K. Maziarz, H. R. Jackson-Flux, P. Cameron, F. Sirockin, N. Schneider, N. Stiefl, M. Segler, and M. Brockschmidt, "Learning to extend molecular scaffolds with structural motifs," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=ZTsoE8G3GG

[221] D. McCabe, K. Butterfield, and L. Treviño, *Cheating in college: Why students do it and what educators can do about it*. The Johns Hopkins University Press, 2012.

[222] N. A. Meanwell, "Improving drug candidates by design: A focus on physicochemical properties as a means of improving compound disposition and safety," *Chemical Research in Toxicology*, vol. 24, no. 9, pp. 1420–1456, Jul. 2011. [Online]. Available: https://doi.org/10.1021/tx200211v

[223] A. Merceron and K. Yacef, "Educational data mining: a case study." in *Proc. Artificial Intelligence in Education*, 2005, pp. 467–474.

[224] E. C. Merkle, "A comparison of imputation methods for bayesian factor analysis models," *Journal of Educational and Behavioral Statistics*, vol. 36, no. 2, pp. 257–276, 2011.

[225] Z. Mhammedi, A. Hellicar, A. Rahman, and J. Bailey, "Efficient orthogonal parametrisation of recurrent neural networks using householder reflections," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 70, Aug. 2017, pp. 2401–2409.

[226] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NeurIPS*, 2013, pp. 3111–3119.

[227] S. Minn, Y. Yu, M. C. Desmarais, F. Zhu, and J.-J. Vie, "Deep knowledge tracing and dynamic student classification for knowledge tracing," in *Proc. IEEE International Conference on Data Mining*. IEEE, 2018, pp. 1182–1187.

[228] R. J. Mislevy, "Bayes modal estimation in item response models," *Psychometrika*, vol. 51, no. 2, pp. 177–195, 1986.

[229] O. J. Mohamed, N. A. Zakar, and B. Alshaikhdeeb, "A combination method of syntactic and semantic approaches for classifying examination questions into bloom's taxonomy cognitive," *J. Eng. Sci. Technol.*, vol. 14, no. 2, pp. 935–950, 2019.

[230] M. Mohammed and N. Omar, "Question classification based on Bloom's taxonomy using enhanced tf-idf," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, pp. 1679–1685, 2018.

[231] J. Morris, "Python language tool," 2021. [Online]. Available: https://github.com/jxmorris12/language_tool_python

[232] M. C. Mozer and R. V. Lindsey, "Predicting and improving memory retention: Psychological theory matters in the big data era," in *Big Data in Cognitive Science*. Psychology Press, 2016, pp. 43–73.

[233] D. Muller, F. K. Ringer, and B. Simon, Eds., *The rise of the modern educational system*. Cambridge, England: Cambridge University Press, Nov. 1989.

[234] J. Naito, Y. Baba, H. Kashima, T. Takaki, and T. Funo, "Predictive modeling of learning continuation in preschool education using temporal patterns of development tests," in *Proc. AAAI*, Feb. 2018.

[235] K. Nandhini and S. R. Balasundaram, "Math word question generation for training the students with learning difficulties," in *Proc. Int. Conf. Workshop Emerg. Trends Technol.*, 2011.

[236] A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera, "Handling Incomplete Heterogeneous Data using VAEs," *arXiv preprint arXiv:1807.03653*, Jul. 2018.

[237] X.-P. Nguyen, S. Joty, S. Hoi, and R. Socher, "Tree-structured attention with hierarchical accumulation," in *Proc. Int. Conf. Learn. Representations*, 2020.

[238] W. Nie, N. Narodytska, and A. Patel, "RelGAN: Relational generative adversarial networks for text generation," in *Proc. ICLR*, 2019.

[239] A. Nigam, P. Friederich, M. Krenn, and A. Aspuru-Guzik, "Augmenting Genetic Algorithms with Deep Neural Networks for Exploring the Chemical Space," *arXiv e-prints*, p. arXiv:1909.11655, Sep. 2019.

[240] B. D. Nye, A. C. Graesser, and X. Hu, "Autotutor and family: A review of 17 years of natural language tutoring," *International Journal of Artificial Intelligence in Education*, vol. 24, no. 4, pp. 427–469, 2014.

[241] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *Journal of Cheminformatics*, vol. 9, no. 1, Sep. 2017. [Online]. Available: https://doi.org/10.1186/s13321-017-0235-x

[242] K. Osadi, M. Fernando, W. Welgama *et al.*, "Ensemble classifier based approach for classification of examination questions into Bloom's taxonomy cognitive levels," *Int. J. Computer Appl.*, vol. 162, no. 4, pp. 1–6, 2017.

[243] A. Osman and A. A. Yahya, "Classifications of exam questions using natural language syntatic features: a case study based on Bloom's taxonomy," in *Proc. Int. Arab Conf. Quality Assurance Higher Edu.*, 2016.

[244] S. Pandey and J. Srivastava, "Rkt: Relation-aware self-attention for knowledge tracing," *arXiv preprint arXiv:2008.12736*, 2020.

[245] J. Pane, E. Steiner, M. Baird, and L. Hamilton, *Continued Progress: Promising Evidence on Personalized Learning*. RAND Corporation, 2015. [Online]. Available: https://doi.org/10.7249/rr1365

[246] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. ACL*, Jul. 2002, pp. 311–318.

[247] Z. A. Pardos and N. T. Heffernan, "Modeling individualization in a Bayesian networks implementation of knowledge tracing," in *Proc. Int. Conf. User Model. Adaptation Personalization*, 2010, pp. 255–266.

[248] Z. A. Pardos, N. T. Heffernan, B. Anderson, C. L. Heffernan, and W. P. Schools, "Using fine-grained skill models to fit student performance with bayesian networks," *Handbook of Dducational Data Mining*, vol. 417, 2010.

[249] M. D. Parenti and G. Rastelli, "Advances and applications of binding affinity prediction methods in drug discovery," *Biotechnology Advances*, vol. 30, no. 1, pp. 244–250, Jan. 2012. [Online]. Available: https://doi.org/10.1016/j.biotechadv.2011.08.003

[250] P. Parhi, A. Pal, and M. Aggarwal, "A survey of methods of collaborative filtering techniques," in *Proc. International Conference on Inventive Systems and Control*, Jan 2017, pp. 1–7.

[251] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2013, pp. 1310–1318.

[252] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[253] P. Pavlik Jr, H. Cen, and K. Koedinger, "Performance factors analysis–A new alternative to knowledge tracing," in *Proc. Int. Conf. Artif. Intell. Educ.*, 2009.

[254] M. Pawlik and N. Augsten, "Tree edit distance: Robust and memory-efficient," *Info. Syst.*, vol. 56, pp. 157 – 173, 2016.

[255] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. EMNLP*, Oct. 2014, pp. 1532–1543.

[256] Perspective, "Using machine learning to reduce toxicity online," 2021. [Online]. Available: https://www.perspectiveapi.com/

[257] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *Proc. Int. Conf. Frontiers Handwriting Recognition (ICFHR)*, Sept. 2014, pp. 285–290.

[258] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Adv. Neural Inf. Process. Syst.*, 2015, pp. 505–513.

[259] C. Piech, M. Sahami, J. Huang, and L. Guibas, "Autonomously generating hints by inferring problem solving policies," in *Proc. ACM conf. learn. Scale*, 2015, pp. 195–204.

[260] J. F. Pimentel, "Python apted algorithm for the tree edit distance," https://github.com/JoaoFelipe/apted, 2017.

[261] P. G. Polishchuk, T. I. Madzhidov, and A. Varnek, "Estimation of the size of drug-like chemical space based on gdb-17 data," *Journal of computer-aided molecular design*, vol. 27, no. 8, pp. 675–679, 2013.

[262] O. Polozov, E. O'Rourke, A. M. Smith, L. Zettlemoyer, S. Gulwani, and Z. Popovic, "Personalized mathematical word problem generation," in *Proc. AAAI*, 2015, p. 381–388.

[263] M. Pota, M. Esposito, and G. De Pietro, “A forward-selection algorithm for SVM-based question classification in cognitive systems,” in *Intell. Interactive Multimedia Syst. Services*, G. D. Pietro, L. Gallo, R. J. Howlett, and L. C. Jain, Eds., 2016, pp. 587–598.

[264] S. Prabhumoye, A. W. Black, and R. Salakhutdinov, “Exploring controllable text generation techniques,” in *Proc. ACL*, Dec. 2020, pp. 1–14.

[265] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proc. Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[266] J. Qin, L. Lin, X. Liang, R. Zhang, and L. Lin, “Semantically-aligned universal tree-structured solver for math word problems,” in *Proc. EMNLP*, Nov. 2020, pp. 3780–3789.

[267] Y. Qiu, Y. Wang, X. Jin, and K. Zhang, “Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision,” in *Proc. Int. Conf. Web Search Data Mining*, 2020, pp. 474–482.

[268] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[269] A. N. Rafferty, M. M. LaMar, and T. L. Griffiths, “Inferring learners’ knowledge from their actions,” *Cognitive Science*, vol. 39, no. 3, pp. 584–618, 2015.

[270] P. Rajpurkar *et al.*, “SQuAD: 100,000+ questions for machine comprehension of text,” in *Proc. Conference on EMNLP*, Nov. 2016, pp. 2383–2392.

[271] A. Ramesh, S. H. Kumar, J. Foulds, and L. Getoor, “Weakly supervised models of aspect-sentiment for online course discussion forums,” in *Proc. Annu. Meeting Assoc. Comput. Linguistics Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 74–83.

[272] R. M. Rao, J. Liu, R. Verkuil, J. Meier, J. Canny, P. Abbeel, T. Sercu, and A. Rives, “Msa transformer,” in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 18–24 Jul 2021, pp. 8844–8856.

[273] G. Rasch, *Studies in mathematical psychology: I. Probabilistic models for some intelligence and attainment tests.*, ser. Studies in mathematical psychology: I. Probabilistic models for some intelligence and attainment tests. Oxford, England: Nielsen & Lydiche, 1960.

[274] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, “Snorkel: rapid training data creation with weak supervision,” *Proc. VLDB Endow.*, vol. 11, no. 3, p. 269–282, Nov. 2017.

[275] S. Ren, D. Guo, S. Lu, L. Zhou, S. Liu, D. Tang, N. Sundaresan, M. Zhou, A. Blanco, and S. Ma, “CodeBLEU: a Method for Automatic Evaluation of Code Synthesis,” *arXiv preprint arXiv:2009.10297*, Sep. 2020.

[276] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 32, no. 2, Jun. 2014, pp. 1278–1286.

[277] ——, “Stochastic backpropagation and approximate inference in deep generative models,” in *Proc. Int. Conf. Mach. Learn.*, 2014, p. II–1278–II–1286.

[278] S. Ritter, J. R. Anderson, K. R. Koedinger, and A. Corbett, “Cognitive tutor: Applied research in mathematics education,” *Psychonomic Bulletin & Review*, vol. 14, no. 2, pp. 249–255, 2007.

[279] K. Rivers and K. R. Koedinger, “Data-driven hint generation in vast solution spaces: a self-improving python programming tutor,” *Int. J. Artif. Intell. Educ.*, vol. 27, no. 1, pp. 37–64, 2017.

[280] D. Rohrer and H. Pashler, “Recent research on human learning challenges conventional instructional strategies,” *Educational Researcher*, vol. 39, no. 5, pp. 406–412, Oct. 2010.

[281] C. Romero and S. Ventura, "Educational data mining: a review of the state of the art," *IEEE Trans. Syst. Man Cybernetics*, vol. 40, no. 6, pp. 601–618, 2010.

[282] S. Roy and D. Roth, "Solving general arithmetic word problems," in *Proc. EMNLP*, Sep. 2015, pp. 1743–1752.

[283] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Conference on Neural Information Processing Systems*, Dec. 2007, pp. 1257–1264.

[284] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 791–798.

[285] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. USA: McGraw-Hill, Inc., 1986.

[286] J. Sandars, R. Correia, M. Dankbaar, P. de Jong, P. S. Goh, I. Hege, K. Masters, S.-Y. Oh, R. Patel, K. Premkumar *et al.*, "Twelve tips for rapidly migrating to online learning during the covid-19 pandemic," *MedEdPublish*, vol. 9, 2020.

[287] A. Sangodiah, R. Ahmad, and W. F. W. Ahmad, "A review in feature extraction approach in question classification using support vector machine," in *IEEE Int. Conf. Control Syst. Comput. Eng.*, 2014, pp. 536–541.

[288] J. N. Sangshetti, S. S. Joshi, R. H. Patil, M. G. Moloney, and D. B. Shinde, "Mur ligase inhibitors as anti-bacterials: A comprehensive review," *Current Pharmaceutical Design*, vol. 23, no. 21, Aug. 2017. [Online]. Available: https://doi.org/10.2174/1381612823666170214115048

[289] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing mathematical reasoning abilities of neural models," *arXiv preprint arXiv:1904.01557*, 2019.

[290] A. M. Schäfer and H. G. Zimmermann, "Recurrent neural networks are universal approximators," in *Proc. Int. Conf. Artificial Neural Netw. (ICANN)*, Sept. 2006, pp. 632–640.

[291] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proc. Int. Conf. World Wide Web*, May 2015, pp. 111–112.

[292] M. H. S. Segler, T. Kogej, C. Tyrchan, and M. P. Waller, "Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks," *arXiv e-prints*, p. arXiv:1701.01329, Jan. 2017.

[293] I. V. Serban *et al.*, "Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus," in *Proc. ACL*, Aug. 2016, pp. 588–598.

[294] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," in *Proc. NeurIPS*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, 2017.

[295] X. Shen, J. Suzuki, K. Inui, H. Su, D. Klakow, and S. Sekine, "Select and attend: Towards controllable content selection in text generation," in *Proc. EMNLP-IJCNLP*, Nov. 2019, pp. 579–590.

[296] Y. Shen, S. Tan, A. Sordoni, and A. Courville, "Ordered neurons: Integrating tree structures into recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2019.

[297] S. Shi, Y. Wang, C.-Y. Lin, X. Liu, and Y. Rui, "Automatically solving number word problems by semantic parsing and reasoning," in *Proc. EMNLP*, Sep. 2015, pp. 1132–1142.

[298] T. Shidi, C. Ruiqi, L. Mengru, L. Qingde, Z. Yanxiang, D. Ji, W. Jiansheng, H. Haifeng, and L. Ming, "Accelerating AutoDock VINA with GPUs," Mar. 2022. [Online]. Available: https://doi.org/10.26434/chemrxiv-2021-3qvn2-v4

[299] B. Shin, S. Park, J. Bak, and J. C. Ho, “Controlled molecule generator for optimizing multiple chemical properties,” in *Proceedings of the Conference on Health, Inference, and Learning*. ACM, Apr. 2021. [Online]. Available: https://doi.org/10.1145/3450439.3451879

[300] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi, “Saint+: Integrating temporal features for ednet correctness prediction,” in *11th Int. Learn. Analytics Knowl. Conf.*, 2021, pp. 490–496.

[301] V. Shiv and C. Quirk, “Novel positional encodings to enable tree-based transformers,” in *Proc. Int. Conf. Neural Info. Process. Syst.*, 2019, pp. 12 081–12 091.

[302] V. J. Shute and J. Psotka, “Intelligent tutoring systems: Past, present, and future,” Tech. Rep., 1994.

[303] N. Siddharth, B. Paige, J.-W. Van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr, “Learning disentangled representations with semi-supervised deep generative models,” in *Proc. Advances in Neural Information Processing Systems*, Dec. 2017, pp. 5925–5935.

[304] H. T. Siegelmann and E. D. Sontag, “On the computational power of neural nets,” *J. Comput. Syst. Sci.*, vol. 50, no. 1, pp. 132–150, Feb. 1995.

[305] A. Singla and N. Theodoropoulos, “From {Solution Synthesis} to {Student Attempt Synthesis} for block-based visual programming tasks,” *arXiv preprint arXiv:2205.01265*, 2022.

[306] G. Sliwoski, S. Kothiwale, J. Meiler, and E. W. Lowe, “Computational methods in drug discovery,” *Pharmacological Reviews*, vol. 66, no. 1, pp. 334–395, Dec. 2013. [Online]. Available: https://doi.org/10.1124/pr.112.007336

[307] J. P. Smith III, A. A. DiSessa, and J. Roschelle, “Misconceptions reconceived: A constructivist analysis of knowledge in transition,” *j. learn. sci.*, vol. 3, no. 2, pp. 115–163, 1994.

[308] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, Oct. 2013, pp. 1631–1642.

[309] J. Song, S. Zhao, and S. Ermon, “A-nice-mc: Adversarial training for mcmc,” in *Adv. Neural Inf. Process. Syst.*, 2017, pp. 5140–5150.

[310] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger, “Algebra dataset,” *Challenge data set from KDD Cup 2010 Educational Data Mining Challenge*, 2010.

[311] J. C. Stamper and Z. A. Pardos, “The 2010 kdd cup competition dataset: Engaging the machine learning community in predictive learning analytics,” 2016.

[312] D. H. Stern, R. Herbrich, and T. Graepel, “Matchbox: large scale online bayesian recommendations,” in *Proc. International World Wide Web Conference*. ACM, 2009, pp. 111–120.

[313] D. Stowell and M. D. Plumbley, “An open dataset for research on audio field recording archives: Freefield1010,” in *Proc. Audio Eng. Soc. 53rd Conf. Semantic Audio (AES53)*, 2014, pp. 1–7.

[314] J. Sun, P. Han, Z. Cheng, E. Wu, and W. Wang, “Transformer based multi-grained attention network for aspect-based sentiment analysis,” *IEEE Access*, vol. 8, pp. 211 152–211 163, 2020.

[315] C. Supriyanto, N. Yusof, B. Nurhadiono *et al.*, “Two-level feature selection for naive bayes with kernel density estimation in question classification based on Bloom’s cognitive levels,” in *Int. Conf. Inf. Technol. Elect. Eng.*, 2013, pp. 237–241.

[316] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. Int. Conf. Neural Info. Process. Syst.*, 2014, p. 3104–3112.

[317] H. Swaminathan and J. A. Gifford, “Bayesian estimation in the rasch model,” *Journal of Educational Statistics*, vol. 7, no. 3, pp. 175–191, 1982.

[318] A. J. Swart, "Evaluation of final examination papers in engineering: a case study using Bloom's taxonomy," *IEEE Trans. Educ.*, vol. 53, no. 2, pp. 257–264, 2010.

[319] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Jul. 2015, pp. 1556–1566.

[320] S. S. Talathi and A. Vartak, "Improving performance of recurrent network network with relu nonlinearity," in *Proc. Int. Conf. Learn. Representations (ICLR)*, Apr. 2016.

[321] B. Tang, J. Ewalt, and H.-L. Ng, "Generative AI models for drug discovery," in *Biophysical and Computational Tools in Drug Discovery*. Springer International Publishing, 2021, pp. 221–243. [Online]. Available: https://doi.org/10.1007/7355_2021_124

[322] K. K. Tatsuoka, "Rule space: An approach for dealing with misconceptions based on item response theory," *Journal of Educational Measurement*, vol. 20, no. 4, pp. 345–354, 1983. [Online]. Available: http://www.jstor.org/stable/1434951

[323] Z. Teng, D. T. Vo, and Y. Zhang, "Context-sensitive lexicon features for neural sentiment analysis," in *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, Nov. 2016, pp. 1629–1638.

[324] M. K. Titsias and M. Lázaro-Gredilla, "Spike and slab variational inference for multi-task and multiple kernel learning," in *Adv. Neural Inf. Process. Syst.*, 2011, pp. 2339–2347.

[325] F. Tonolini, B. S. Jensen, and R. Murray-Smith, "Variational sparse coding," July 2019. [Online]. Available: http://eprints.gla.ac.uk/191553/

[326] H.-Y. Tseng, H.-Y. Lee, L. Jiang, M.-H. Yang, and W. Yang, "RetrieveGAN: Image synthesis via differentiable patch retrieval," in *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 242–257. [Online]. Available: https://doi.org/10.1007/978-3-030-58598-3_15

[327] C. A. Twigg, "Improving learning and reducing costs: New models for online learning," *EDUCAUSE Review*, vol. 38, no. 5, pp. 28–38, 2003.

[328] S. Upadhyay and M.-W. Chang, "Draw: A challenging and diverse algebra word problem set," Microsoft, Tech. Rep., 2015.

[329] J. Vamathevan, D. Clark, P. Czodrowski, I. Dunham, E. Ferran, G. Lee, B. Li, A. Madabhushi, P. Shah, M. Spitzer, and S. Zhao, "Applications of machine learning in drug discovery and development," *Nature Reviews Drug Discovery*, vol. 18, no. 6, pp. 463–477, Apr. 2019. [Online]. Available: https://doi.org/10.1038/s41573-019-0024-5

[330] W. J. van der Linden and R. K. Hambleton, *Handbook of modern item response theory*. Springer Science & Business Media, 2013.

[331] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[332] H. L. van Trees, *Detection, Estimation, and Modulation Theory: Radar-Sonar Signal Processing and Gaussian Signals in Noise*. Krieger Publishing Co., Inc., 1992.

[333] ——, *Detection, estimation, and modulation theory, part I*. John Wiley & Sons, Inc., 2013.

[334] K. VanLehn, "Bugs are not enough: Empirical studies of bugs, impasses and repairs in procedural skills." *J. Math. Behavior*, 1982.

[335] ——, "Student modeling," *Foundations of Intelligent Tutoring Systems*, vol. 55, p. 78, 1988.

[336] P. Varma, B. D. He, P. Bajaj, N. Khandwala, I. Banerjee, D. Rubin, and C. Ré, "Inferring generative model structure with static analysis," in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 240–250.

[337] P. Varma and C. Ré, "Snuba: automating weak supervision to label training data," *Proc. VLDB Endow.*, vol. 12, no. 3, p. 223–236, Nov. 2018.

[338] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, 2017.

[339] U. Verawardina, L. Asnur, A. L. Lubis, Y. Hendriyani, D. Ramadhani, I. P. Dewi, R. Darni, T. J. Betri, W. Susanti, and T. Sriwahyuni, "Reviewing online learning facing the covid-19 outbreak." *Talent Development & Excellence*, vol. 12, 2020.

[340] L. Verschaffel, S. Schukajlow, J. Star, and W. V. Dooren, "Word problems in mathematics education: a survey," *ZDM*, vol. 52, no. 1, pp. 1–16, Jan. 2020.

[341] J.-J. Vie and H. Kashima, "Knowledge tracing machines: Factorization machines for knowledge tracing," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 750–757.

[342] E. M. Voorhees, D. K. Harman *et al.*, *TREC: Experiment and evaluation in information retrieval.* MIT press Cambridge, 2005, vol. 63.

[343] S. Wager, S. Wang, and P. Liang, "Dropout training as adaptive regularization," in *Proc. Advances Neural Inform. Process. Syst. (NIPS)*, vol. 1, Dec. 2013, pp. 351–359.

[344] C. A. Walkington, "Using adaptive learning technologies to personalize instruction to student interests: The impact of relevant contexts on performance and learning outcomes." *J. Educ. Psychol.*, vol. 105, no. 4, pp. 932–945, Nov. 2013.

[345] W. P. Walters and M. Murcko, "Assessing the impact of generative AI on medicinal chemistry," *Nature Biotechnology*, vol. 38, no. 2, pp. 143–145, Jan. 2020. [Online]. Available: https://doi.org/10.1038/s41587-020-0418-2

[346] W. Walters, M. T. Stahl, and M. A. Murcko, "Virtual screening—an overview," *Drug Discovery Today*, vol. 3, no. 4, pp. 160–178, Apr. 1998. [Online]. Available: https://doi.org/10.1016/s1359-6446(97)01163-x

[347] H. Wang, X. Shi, and D.-Y. Yeung, "Relational stacked denoising autoencoder for tag recommendation," in *Proc. AAAI Confefence on Artificial Intelligence*, Jan. 2015, pp. 3052–3058.

[348] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, "Translating a math word problem to a expression tree," in *Proc. EMNLP*, Oct.-Nov. 2018, pp. 1064–1069.

[349] L. Wang, D. Zhang, L. Gao, J. Song, L. Guo, and H. T. Shen, "Mathdqn: Solving arithmetic word problems via deep reinforcement learning," in *Proc. AAAI*, S. A. McIlraith and K. Q. Weinberger, Eds., 2018, pp. 5545–5552.

[350] L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B. T. Dai, and H. T. Shen, "Template-based math word problem solvers with recursive neural networks," in *Proc. AAAI*, vol. 33, Jul. 2019, pp. 7144–7151.

[351] L. Wang, A. Sy, L. Liu, and C. Piech, "Learning to represent student knowledge on programming exercises using deep learning." *Int. Educ. Data Mining Soc.*, 2017.

[352] T. Wang, F. Ma, Y. Wang, T. Tang, L. Zhang, and J. Gao, "Towards learning outcome prediction via modeling question explanations and student responses," in *Proc. SIAM Int. Conf. Data Mining*. SIAM, 2021, pp. 693–701.

[353] T. Wang, X. Yuan, and A. Trischler, "A joint model for question answering and question generation," *arXiv:1706.01450*, 2017.

[354] W. Wang, Z. Gan, H. Xu, R. Zhang, G. Wang, D. Shen, C. Chen, and L. Carin, "Topic-guided variational auto-encoder for text generation," in *Proc. NAACL*, Jun. 2019, pp. 166–177.

[355] Y. Wang, X. Liu, and S. Shi, “Deep neural solver for math word problems,” in *Proc. EMNLP*, Sep. 2017, pp. 845–854.

[356] Y. Wang, H.-Y. Lee, and Y.-N. Chen, “Tree transformer: Integrating tree structures into self-attention,” in *Proc. Conf. Empirical Methods Natural Lang. Process. and Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 1061–1070.

[357] Y. Wang and N. Heffernan, “Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes,” in *Int. conf. artif. intell. educ.* Springer, 2013, pp. 181–188.

[358] Y. Wang, J. Zheng, Q. Liu, Z. Zhao, J. Xiao, and Y. Zhuang, “Weak supervision enhanced generative network for question generation,” *arXiv preprint*, 2019.

[359] Z. Wang, A. Lamb, E. Saveliev, P. Cameron, Y. Zaykov, J. M. Hernández-Lobato, R. E. Turner, R. G. Baraniuk, C. Barton, S. P. Jones *et al.*, “Diagnostic questions: The neurips 2020 education challenge,” *arXiv preprint arXiv:2007.12061*, 2020.

[360] Z. Wang, A. S. Lan, and R. G. Baraniuk, “Mathematical formula representation via tree embeddings,” Dec. 2021.

[361] Z. Wang, A. S. Lan, W. Nie, A. E. Waters, P. J. Grimaldi, and R. G. Baraniuk, “Qg-net: A data-driven question generation model for educational content,” in *Proc. Fifth Annu. ACM Conf. Learn. at Scale*, 2018.

[362] A. E. Waters, C. Studer, and R. G. Baraniuk, “Collaboration-type identification in educational datasets,” *Journal of Educatinal Data Mining*, vol. 6, no. 1, pp. 28–52, 2014.

[363] A. E. Waters, D. Tinapple, and R. G. Baraniuk, “Bayesrank: A bayesian approach to ranked peer grading,” in *Proc. ACM Conference on Learning at Scale*, Mar. 2015, pp. 177–183.

[364] D. Weininger, “SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules,” *Journal of Chemical Information and Modeling*, vol. 28, no. 1, pp. 31–36, Feb. 1988. [Online]. Available: https://doi.org/10.1021/ci00057a005

[365] C. Wiklund-Hörnqvist, B. Jonsson, and L. Nyberg, “Strengthening concept learning by repeated testing,” *Scandinavian journal of psychology*, vol. 55, no. 1, pp. 10–16, 2014.

[366] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Mach. Learn.*, vol. 8, no. 3–4, p. 229–256, May 1992.

[367] S. Williams, “Generating mathematical word problems,” in *Proc. AAAI*, vol. FS-11-04, 2011.

[368] A. Willis *et al.*, “Key phrase extraction for generating educational question-answer pairs,” in *Proc. Conference on Learning at Scale*, 2019.

[369] K. H. Wilson, Y. Karklin, B. Han, and C. Ekanadham, “Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation,” in *Proc. Educational Data Mining*, 2016.

[370] R. Winter, F. Montanari, F. Noé, and D.-A. Clevert, “Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations,” *Chemical Science*, vol. 10, no. 6, pp. 1692–1701, 2019. [Online]. Available: https://doi.org/10.1039/c8sc04175j

[371] R. Winter, F. Montanari, A. Steffen, H. Briem, F. Noé, and D.-A. Clevert, “Efficient multi-objective molecular optimization in a continuous latent space,” *Chemical Science*, vol. 10, no. 34, pp. 8016–8024, 2019. [Online]. Available: https://doi.org/10.1039/c9sc01928f

[372] S. Wisdom, T. Powers, J. R. Hershey, J. Le Roux, and L. Atlas, “Full-capacity unitary recurrent neural networks,” in *Proc. Advances Neural Inform. Process. Syst. (NIPS)*, Dec. 2016.

[373] M. Wu, R. L. Davis, B. W. Domingue, C. Piech, and N. Goodman, “Variational item response theory: Fast, accurate, and expressive,” *arXiv preprint arXiv:2002.00276*, 2020.

[374] M. Wu, M. Mosse, N. Goodman, and C. Piech, "Zero shot learning for code education: Rubric sampling with deep learning inference," in *Proc. AAAI Confefence on Artificial Intelligence*, vol. 33, Jul. 2019, pp. 782–790.

[375] Q. Wu, Q. Zhang, J. Fu, and X. Huang, "A knowledge-aware sequence-to-tree network for math word problem solving," in *Proc. EMNLP*, Nov. 2020, pp. 7137–7146.

[376] Y. Wu, P. Zhou, A. G. Wilson, E. Xing, and Z. Hu, "Improving gan training with probability ratio clipping and sample reweighting," in *Proc. NeurIPS*, vol. 33, 2020, pp. 5729–5740.

[377] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy, "Memorizing transformers," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=TrjbxzRcnf-

[378] Y. Xie, C. Shi, H. Zhou, Y. Yang, W. Zhang, Y. Yu, and L. Li, "{MARS}: Markov molecular sampling for multi-objective drug discovery," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=kHSu4ebxFXY

[379] Z. Xie and S. Sun, "A goal-driven tree-structured neural model for math word problems," in *Proc. IJCAI*, Aug. 2019.

[380] P. Xu, M. Patwary, M. Shoeybi, R. Puri, P. Fung, A. Anandkumar, and B. Catanzaro, "MEGATRON-CNTRL: Controllable story generation with external knowledge using large-scale language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020.

[381] Y. Xu, K. Lin, S. Wang, L. Wang, C. Cai, C. Song, L. Lai, and J. Pei, "Deep learning for molecular generation," *Future Medicinal Chemistry*, vol. 11, no. 6, pp. 567–597, Mar. 2019. [Online]. Available: https://doi.org/10.4155/fmc-2018-0358

[382] D. Xue, Y. Gong, Z. Yang, G. Chuai, S. Qu, A. Shen, J. Yu, and Q. Liu, "Advances and challenges in deep generative models for de novo molecule generation," *WIREs Computational Molecular Science*, vol. 9, no. 3, Oct. 2018. [Online]. Available: https://doi.org/10.1002/wcms.1395

[383] A. A. Yahya and A. Osman, "Automatic classification of questions into Bloom's cognitive levels using support vector machines," in *Int. Arab Conf. Inform. Techno.*, 2011.

[384] D. Yan, A. A. Von Davier, and C. Lewis, *Computerized multistage testing: Theory and applications*. CRC Press, 2016.

[385] Y. Yang, J. Shen, Y. Qu, Y. Liu, K. Wang, Y. Zhu, W. Zhang, and Y. Yu, "Gikt: A graph-based interaction model for knowledge tracing," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2020.

[386] Z. Yang, J. Hu, R. Salakhutdinov, and W. Cohen, "Semi-supervised QA with generative domain-adaptive nets," in *Proc. ACL*, Jul. 2017, pp. 1040–1050.

[387] M. Yasunaga and J. Lafferty, "TopicEq: A Joint Topic and Mathematical Equation Model for Scientific Texts," in *Proc. AAAI conf. Artificial Intell.*, 2019.

[388] N. Yoshikawa, K. Terayama, M. Sumita, T. Homma, K. Oono, and K. Tsuda, "Population-based de novo molecule generation, using grammatical evolution," *Chemistry Letters*, vol. 47, no. 11, pp. 1431–1434, Nov. 2018. [Online]. Available: https://doi.org/10.1246/cl.180665

[389] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 6412–6422.

[390] K. Yuan, D. He, Z. Jiang, L. Gao, Z. Tang, and C. L. Giles, "Automatic generation of headlines for online math questions," in *Proc. AAAI conf. Artificial Intell.*, 2020, pp. 9490–9497.

[391] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon, "Individualized bayesian knowledge tracing models," in *Int. Conf. artif. intell. educ.* Springer, 2013, pp. 171–180.

[392] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. Conference on Neural Information Processing Systems*, 2017, pp. 3391–3401.

[393] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, and K. Davila, "Ntcir-12 mathir task overview." in *Proc. NTCIR Conf. Eval. Info. Access*, 2016.

[394] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *Int. J. Document Anal. Recognit.*, vol. 15, no. 4, pp. 331–357, Dec 2012.

[395] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," *ArXiv e-prints*, vol. 1409.2329, Sep. 2014.

[396] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 2008–2026, Aug 2019.

[397] C.-H. Zhang, E. A. Stone, M. Deshmukh, J. A. Ippolito, M. M. Ghahremanpour, J. Tirado-Rives, K. A. Spasov, S. Zhang, Y. Takeo, S. N. Kudalkar, Z. Liang, F. Isaacs, B. Lindenbach, S. J. Miller, K. S. Anderson, and W. L. Jorgensen, "Potent noncovalent inhibitors of the main protease of SARS-CoV-2 from molecular sculpting of the drug perampanel guided by free energy perturbation calculations," *ACS Central Science*, vol. 7, no. 3, pp. 467–475, Feb. 2021. [Online]. Available: https://doi.org/10.1021/acscentsci.1c00039

[398] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, "A novel neural source code representation based on abstract syntax tree," in *IEEE/ACM 41st Int. Conf. Software Eng.* IEEE, 2019, pp. 783–794.

[399] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," in *Proc. Int. Conf. World Wide Web*, Apr. 2017, pp. 765–774.

[400] M. Zhang, Z. Wang, R. Baraniuk, and A. Lan, "Math operation embeddings for open-ended solution analysis and feedback," *Proc. Int. Conf. Educ. Data Min.*, pp. 216–227, June 2021.

[401] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, "D-vae: A variational autoencoder for directed acyclic graphs," in *Proc. Int. Conf. Neural Info. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, 2019.

[402] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson, "Cyclical stochastic gradient mcmc for bayesian deep learning," *Proc. Int. Conf. Learn Representations*, 2020.

[403] X. Zhang, A. Bosselut, M. Yasunaga, H. Ren, P. Liang, C. D. Manning, and J. Leskovec, "GreaseLM: Graph REASoning enhanced language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=41e9o6cQPj

[404] J.-h. Zhao and L. Philip, "A note on variational bayesian factor analysis," *Neural Networks*, vol. 22, no. 7, pp. 988–997, 2009.

[405] S. Zhao, J. Song, and S. Ermon, "Infovae: Balancing learning and inference in variational autoencoders," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 33, Feb. 2019, pp. 5885–5892.

[406] W. Zhong, S. Rohatgi, J. Wu, C. Giles, and R. Zanibbi, "Accelerating substructure similarity search for formula retrieval," in *Proc. European Conf. Info. Retrieval*, 2020, pp. 714–727.

[407] W. Zhong and R. Zanibbi, "Structural similarity search for formulas using leaf-root paths in operator subtrees," in *Proc. Int. Conf. Neural Info. Process. Syst.*, L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, Eds., 2019, pp. 116–129.

[408] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, “A C-LSTM Neural Network for Text Classification,” *ArXiv e-prints*, vol. 1511.08630, Nov. 2015.

[409] Q. Zhou and D. Huang, “Towards generating math word problems from equations and topics,” in *Proc. Int. Conf. Natural Lang. Gener.*, Oct.–Nov. 2019, pp. 494–503.

[410] Z. Zhou, S. Kearnes, L. Li, R. N. Zare, and P. Riley, “Optimization of molecules via deep reinforcement learning,” *Scientific Reports*, vol. 9, no. 1, Jul. 2019. [Online]. Available: https://doi.org/10.1038/s41598-019-47148-x

[411] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proc. ICCV*, 2017, pp. 2242–2251.

[412] R. Zhu, D. Zhang, C. Han, M. Gao, X. Lu, W. Qian, and A. Zhou, “Programming knowledge tracing: A comprehensive dataset and a new model,” *arXiv preprint arXiv:2112.08273*, 2021.

[413] J. Zhuo, Z. Xu, W. Dai, H. Zhu, H. Li, J. Xu, and K. Gai, “Learning optimal tree models under beam search,” in *Proc. Int. Conf. Mach. Learn.*, vol. 119, 13–18 Jul 2020, pp. 11 650–11 659.

[414] H. Zimmermann, C. Tietz, and R. Grothmann, “Forecasting with recurrent neural networks: 12 tricks,” *Neural Netw.: Tricks of the Trade*, pp. 687–707, 2012.

[415] U. Zoller, “Are lecture and learning compatible? maybe for locs: Unlikely for hocs,” *J. Chemical Edu.*, vol. 70, no. 3, p. 195, 1993.

[416] Y. Zou and W. Lu, “Text2Math: End-to-end parsing text into math expressions,” in *Proc. EMNLP-IJCNLP*, Nov. 2019, pp. 5327–5337.

Appendix A

A.1 Gumbel-Softmax in Section 2.2.1

We describe in detail the procedure to approximate sampling M' from p_Φ, i.e., sampling discrete tokens $m'_t \sim p_\Phi(m_t|E, \mathbf{c}, m_1, \ldots, m_{t-1})$, using the Gumbel-softmax relaxation. In the first step, we reparametrize sampling from a categorical distribution p_Θ using the Gumbel-max trick [209] as follows:

$$\begin{aligned} u^{(i)} &\sim \mathrm{uniform}(0,1)\,, \\ g_t^{(i)} &= -\log(-\log(u^{(i)}))\,, \\ m_t &= \mathrm{one\_hot}\Big(\operatorname*{argmax}_{i\in|V|}\big(f_{\Theta,t}^{(i)} + g_t^{(i)}\big)\Big)\,, \end{aligned}$$

where $|V|$ is the size of the vocabulary, $f_{\Theta,t}^{(i)}$ is the pre-softmax activation of p_Θ at the t-th generation step for the i-th word, and $g_t^{(i)}$ are i.i.d. samples from the standard Gumbel distribution.

In the second step, we approximate the discrete argmax operator with the continuous, differentiable softmax operator, which enables us to obtain the final approximation

$$m'_t = \mathrm{softmax}((f_{\Theta,t} + g_t)/\tau)\,,$$

where τ is a temperature hyperparameter, resulting in the Gumbel-softmax distribution. When τ approaches 0, this approximation approaches the categorical distribution

Table A.1 : Model configurations.

architecture	#layers	input size	layer size	#params	optimizer	learning rate	batch size	training epochs/steps
seq2seq-rnn	2	300	512	11M	adagrad	0.15	64	{5000, 15000*}
seq2seq-attn-rnn	1	300	512	11M	adagrad	0.15	64	{5000, 15000*}
seq2seq-transformer	6	512	512	52M	Adam	2	4096	{5000, 15000*}
GPT	36	1280	1280	774M	Adam	5e-5	8	20
ours	36	1280	1280	774M	Adam	5e-5	{8*, 16}	{5, 15†}

Table A.2 : Additional examples of MWPs generated by our approach.

Equation	`x = num1 / num2`
Context	`Sue bag bags cookies fill mother`
Gen. MWP	Sue ' s mom baked num1 cookies. If she wants to distribute the cookies among num2 children, how many cookies will each child get?
Equation	`x = num1 + num2 + num3`
Context	`Charlie Cortland bag bags fruit pick picked visit`
Gen. MWP	Cortland picked num1 pears, and Mike picked num2 pears, and Alyssa picked num3 pears from the pear tree. How many pears were picked in total?
Equation	`x = num1 - num2`
Context	`cousin game playing points scored video`
Gen. MWP	Zach scored num1 points in the football game. Ben scored num2 points. How many more points did Zach score than Ben?
Equation	`x = num1 + num2`
Context	`teacher worksheet`
Gen. MWP	The secretary prints a copy of the worksheet, num1 copies in the morning, and num2 copies in the afternoon. How many copies were printed throughout the day?
Equation	`x = ( num1 + num2 )`
Context	`mike baseball football marble total toy`
Gen. MWP	The total cost of a toy factory to produce a football is num1 yuan, which is num2 yuan less than the total cost. How much is the total cost?
Equation	`x = ( num1 * num2 )`
Context	`anne hour mile`
Gen. MWP	It takes num2 hours for a car to travel num1 kilometers per hour from A to B. How many kilometers are the distance between A and B?

parametrized by $\mathrm{one\_hot}\big(\mathrm{argmax}_{i\in|V|}(f_{\Theta,t})\big)$.

A.2 Quality of the Math23K Dataset

Some reviewers brought up a concern on the quality of the Math23K dataset because it is originally in Chinese and we use the English-translated version (via Google Translate

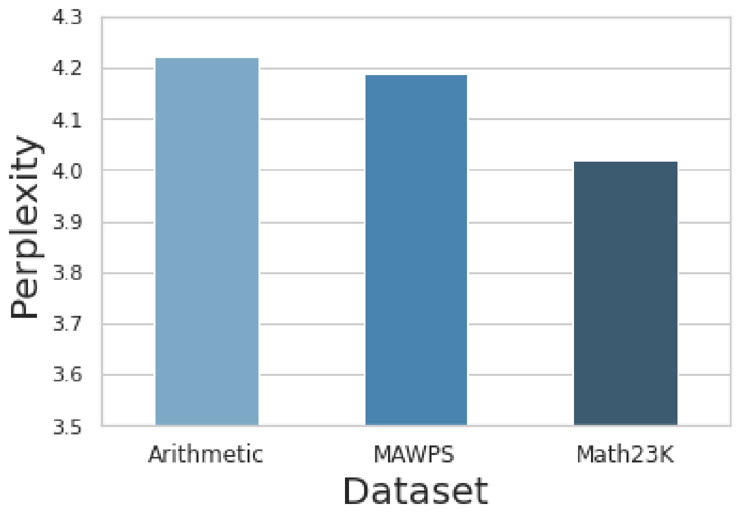


Figure A.1 : Averaged perplexity of each dataset under a small GPT-2. The translated Math23K dataset has similar perplexity compared to the other two datasets, suggesting similar language quality of the three datasets.

API) of this dataset. Despite using an automated translation service, we find that most data points in the translated Math23K dataset is good enough to use for training and evaluation. Figure A.1 reports the perplexity score under a small GPT-2 model for each dataset, averaged over all data points. We see that the translated Math23K dataset has comparable perplexity compared to that of the other two datasets. This observation suggests that the translated Math23K dataset has similar language quality compared to the other two datasets that are originally in English.

A.3 Experiment Details

A.3.1 Training Details

We train the three components in our method jointly. Specifically, we first jointly train the context keyword selection model and the MWP generator. After that, we freeze the context selection model and continue to jointly train the MWP generator and the mwp2eq model.

Notably, the input token embeddings to the context selector are the same as the pre-trained GPT-2 token embeddings. These embeddings are kept fixed throughout training for training stability.

Table A.1 provides the configurations for all models under consideration. The numbers marked with an asteroid (*) are the setting for the Math23K dataset. The number marked with a dagger (†) is the configuration for our approach. For all baselines, we use the noun+pronoun words (see Section A.3.2) extracted from the MWPs as the input context. Each model are trained on a single NVIDIA RTX 8000 GPU. For the GPT-based models, including our approach, the training time ranging from 1.3 minute per epoch (for the Arithmetic and MAWPS datasets) to 20 min per epoch (for the Math23K dataset). The much slower runtime for the Math23K dataset is due to both its smaller batch size in order to fit into memory and its large size, which is almost 20 times as big as the Arithmetic dataset. For the seq2seq baselines, the training speed is about 7 seconds per 10 steps (equivalent of 640 data points). The seq2seq baselines are implemented using the OpenNMT framework.[1] The pre-trained GPT-2 model is loaded from the HuggingFace repository.[2]

When training all models, we also randomly drop words in the context with a probability of 0.3 for each word and permute the order of these words. We do so to improve the models' robustness with respect to the number and the order of the keywords in the context. In addition, for the seq2seq baselines, we additionally lowercase all string to reduce the total number of vocabulary.

[1]`https://opennmt.net/`

[2]`https://huggingface.co/`

Table A.3 : Examples of failed cases.

Equation	`x = ( num1 / num2)`
Context	`Tom cars dollars money week weekend`
Gen. MWP	On Sunday, num1 yuan can buy num2 cars. So how much money is needed to buy a car?
Error	Incomplete information: implicitly assumes each car costs the same. Can be remedied by adding "on average".)
Equation	`x = ( num1 * num2 )`
Context	`Tom box boxes brother candy chocolate pieces`
Gen. MWP	There are num2 boxes of chocolates in the candy store, and the price is num1 yuan per piece. How much does it cost to buy a piece of chocolate?
Error	Wrong question asked. Can be remedied by changing the "a piece of chocolate" to "those chocolates".
Equation	`x = ( num1 * num2 )`
Context	`David box dog dogs dollars toy`
Gen. MWP	A toy dog is num1 yuan, and the price of a puppy is num2 times that of a puppy. How much is a puppy?
Error	Incoherent question statement. Can be remedied by changing the second "puppy" to "dog".

A.3.2 Baselines for the Ablation Study

Here, we explain the different method to select keywords. We first tokenize[3] the MWPs. For the noun+pronoun method, we extract words that have "noun" or "pronoun" as their part-of-speech tags. Stopwords and punctuation are not included. For the TF-IDF method, we compute the TF-IDF weights for all tokens, again excluding stopwords and punctuation, and then choose 5 words with the highest weights for each MWP as its context.

A.3.3 Mathematical Consistency Metric

In principle, A more accurate mwp2eq model leads to more accurate mathematical consistency evaluation and many other state-of-the-art mwp2eq methods, including those targeting automatic MWP answering that we reviewed in Section 2.4, can be employed. We have observed that fine-tuning pre-trained GPT-2 achieves competitive performance comparing to

[3]`https://spacy.io/`

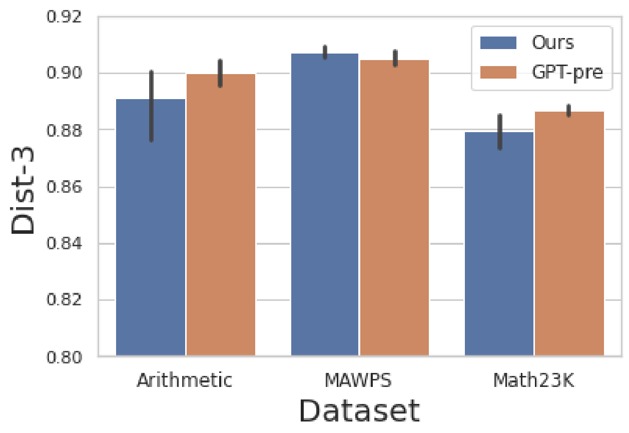


Figure A.2 : Diversity of generation comparing our approach with a fine-tuned pre-trained GPT-2. Our approach achieves similar generation diversity according to the Dist-3 metric.

a number of existing approaches and thus use it for this present work. Using more advanced methods to improve the mathematical consistency evaluation is left for future work.

A.4 Additional Results

A.4.1 Generation Diversity

Per the reviewers request, we compare the generation diversity using the Dist-3 metric [185] in Figure A.2, where higher numbers indicating more diversity. We can observe that our approach achieves similar generation diversity across all datasets compared to GPT-2, with differences smaller than 0.1, suggesting our regularizations do not compromise the generation diversity.

A.4.2 Additional Qualitative Examples

Table A.2 presents additional examples of MWPs generated by our approach. The contexts and equations in the first three rows and the last three rows are taken from the Arithmetic and the Math23K datasets, respectively. These examples are consistent with the qualitative results in Section 2.3.2.

A.5 Limitations

Despite promising results, our approach can still generates problematic MWPs. Because some of our baselines simply copy a sample from the training data as the “generated” sample during evaluation, which would make a unfair comparison, here we instead conduct a small case study for our approach on the most challenging generation scenario where we randomly sample 25 contexts and combine it with each of the four equations that involve two variables.[4] This procedure produces 100 generated examples. We then qualitatively evaluate their generation quality. In total, we find that 17 out of the 100 generated samples are completely satisfactory and another 17 can become satisfactory with minor changes. Some common errors in our generated samples include: 1) incomplete information; 2) wrong question asked; and 3) incoherent question statement. We illustrate these types of errors in Table A.3. The errors suggest that, for better generation quality, we should further improve the model’s understanding of the semantics of the math operations and the relationship between various mathematical entities in the equation and the important words in the MWPs.

[4]In general, evaluating generated MWPs is a challenging task and we defer the investigation of human evaluation criteria and more comprehensive human evaluations to a future work.

Appendix B

B.1 Dataset Statistics and Preprocessing Steps

Statistic	**Raw**	**Processsed**
#codes	46825	39796
#avg. lines of code per submission	17.52	17.64
#avg. submissions per student	190.34	161.77
#avg. submissions per problem	936.5	795.9

Table B.4 : Dataset statistics comparing the raw and our processed dataset, the latter of which is used throughout our experiments.

Since we choose to use the AST representation for code, we perform a preprocessing step to remove student solutions that cannot be converted to AST format. Overall, about 85% of all student solutions are AST-convertible, which means that this preprocessing step does not result in significant data loss. Table B.4 describes the summary statistics of the original dataset and the resulting preprocessed dataset that we use for all our experiments. For KT, we follow standard procedure in the literature [258, 92, 399] by setting the maximum solution sequence for any student to 200. For students with more than 200 solutions, we split their solutions into separate sequences of length 200.

B.2 Experimental Setup Details

In both settings, we split all students in the dataset into disjoint (train, validation, test) sets with a $80\% - 10\% - 10\%$ ratio and report all metrics on the test set. In the default

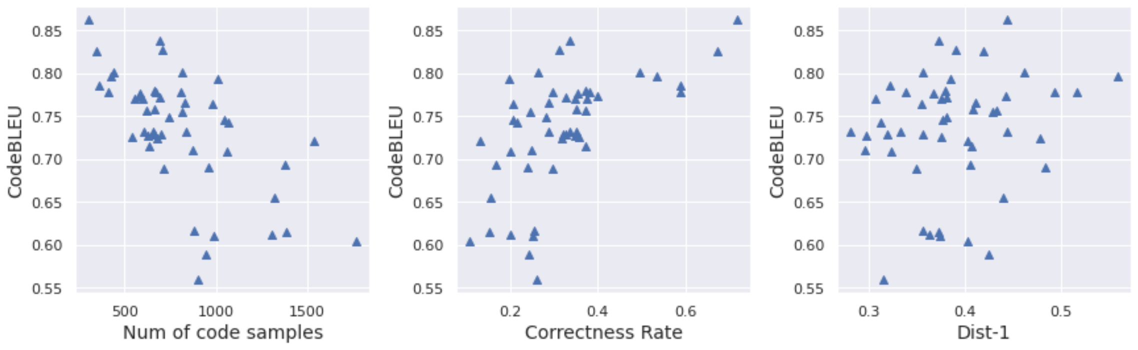
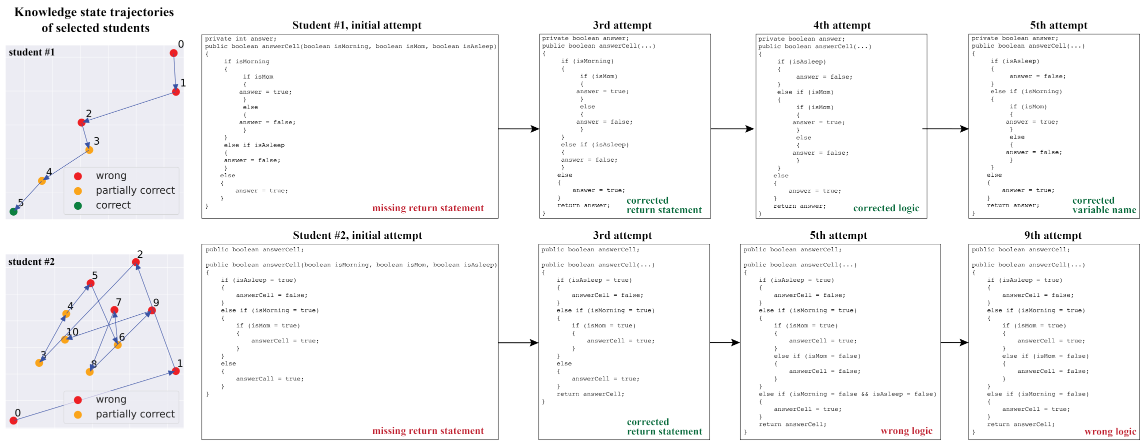


Figure B.3 : Visualization of CodeBLEU metric versus number of student responses (left), rate of correct submissions (middle) and Dist-1 metric (right) in each question. Each point represents one question.

setting, we add knowledge states into token embeddings for the question prompt as input to the RG component using a linear projection layer as detailed above. We pre-train both the KE and RG components of OKT on the training data with a correctness prediction objective (i.e., pre-train an existing KT method) and a prompt-to-code supervised generation objective, respectively. For the KE component, we follow the original DKT, DKVMN, and AKT methods with 768 hidden units in their models. When combining DKVMN or AKT with the answer generator, we use the context reader output from DKVMN and the hidden state from AKT, respectively, as input to the answer generator at each time step. We refer readers to [399, 92] for more details. For the RG component, we use a small GPT-2 with 12 transformer decoder layers [268]. We use the RMSProp optimizer for the knowledge update component and the Adam optimizer for the RG component, both with a default learning rate of 0.00001. Also, we freeze the parameters of the question and code representation models and only train the KT model and the answer prediction model, Although the former two components can also be optimized.

We run all experiments using a single NVIDIA Quadro RTX 8000 GPU. The KT model pre-training usually takes less than 5 minutes per epoch of wall clock time. The OKT



Student #1, initial attempt

```
private int answer;
public boolean answerCell(boolean isMorning, boolean isMom, boolean isAsleep)
{
     if isMorning
     {
          if isMom
          {
         answer = true;
          }
          else
          {
         answer = false;
          }
     }
     else if isAsleep
     {
     answer = false;
     }
    else
    {
        answer = true;
    }
}
```

3rd attempt

```
private boolean answer;
public boolean answerCell(...)
{
     if (isMorning)
     {
          if (isMom)
          {
         answer = true;
          }
          else
          {
         answer = false;
          }
     }
     else if (isAsleep)
     {
     answer = false;
     }
    else
    {
        answer = true;
    }
    return answer;
}
```

4th attempt

```
private boolean answer;
public boolean answerCell(...)
{
     if (isAsleep)
     {
          answer = false;
     }
     else if (isMom)
     {
          if (isMom)
          {
         answer = true;
          }
          else
          {
         answer = false;
          }
     }
    else
    {
        answer = true;
    }
    return answer;
}
```

5th attempt

```
private boolean answer;
public boolean answerCell(...)
{
     if (isAsleep)
     {
          answer = false;
     }
     else if (isMorning)
     {
          if (isMom)
          {
         answer = true;
          }
          else
          {
         answer = false;
          }
     }
    else
    {
        answer = true;
    }
    return answer;
}
```

Student #2, initial attempt

```
public boolean answerCell;

public boolean answerCell(boolean isMorning, boolean isMom, boolean isAsleep)
{
    if (isAsleep = true)
    {
        answerCell = false;
    }
    else if (isMorning = true)
    {
        if (isMom = true)
        {
            answerCell = true;
        }
    }
    else
    {
        answerCall = true;
    }
}
```

3rd attempt

```
public boolean answerCell;

public boolean answerCell(...)
{
    if (isAsleep = true)
    {
        answerCell = false;
    }
    else if (isMorning = true)
    {
        if (isMom = true)
        {
            answerCell = true;
        }
    }
    else
    {
        answerCell = true;
    }
    return answerCell;
}
```

5th attempt

```
public boolean answerCell;

public boolean answerCell(...)
{
    if (isAsleep = true)
    {
        answerCell = false;
    }
    else if (isMorning = true)
    {
        if (isMom = true)
        {
            answerCell = true;
        }
        else if (isMom = false)
        {
            answerCell = false;
        }
    }
    else if (isMorning = false && isAsleep = false)
    {
        answerCell = true;
    }
    return answerCell;
}
```

9th attempt

```
public boolean answerCell;

public boolean answerCell(...)
{
    if (isAsleep = true)
    {
        answerCell = false;
    }
    else if (isMorning = true)
    {
        if (isMom = true)
        {
            answerCell = true;
        }
        else if (isMom = false)
        {
            answerCell = false;
        }
    }
    else if (isMorning = false)
    {
        answerCell = true;
    }
    return answerCell;
}
```

Figure B.4 : Four sample submissions of two students corresponding to the top right figures, respectively. One student gradually proceeded to a correct code while the other got stuck.

training with DKT as the KT model takes about 10 minutes and 30 minutes per epoch of wall clock time for the the two scenarios, namely, using only students' first submitted code and all submitted code that can be converted to AST format, respectively. OKT training with AKT as the KT model takes about the same time as DKT as the KT model while with DKVMN, training is about 1.5 times slower due to the more expensive memory computation [399].

B.3 Visualizing Quantitative Results

Following the results in Section 7.3.1 and Table 7.1, we additionally examine the model performance across questions and measure the correlation between its CodeBLEU score and some features (i.e. difficulty level, response diversity). Figure B.3 shows that model performance has a positive correlation with student performance, i.e., the portion of correct responses, and a negative correlation with the number of student responses. In other words, an easy question with fewer submissions is more likely to achieve better prediction results. However, CodeBLEU is minimally correlated with the diversity in student responses. Also, the range of CodeBLEU performance across questions is relatively big, with the highest of 0.86 and lowest of 0.56.

B.4 Visualizing Code Revisions

We also show how the learnt knowledge state space can be useful for tracing and understanding students' consecutive submissions to the same question. On the right-hand side of Figure B.4, we show the knowledge state trajectories of two students responding to this question. The colors in these two figures represent knowledge states that correspond to wrong, partially correct and fully correct codes at different time steps. We see that both students start with a wrong solution. However, one student gradually proceeded to the correct solution after a few edits, whereas the other student got stuck after a few unsuccessful edits and eventually gave up on solving this prompt. The steady progress versus getting stuck is clearly visualized in these figures, where for the former student, the knowledge states gradually moves from the upper right corner in the knowledge state space to the lower left, whereas for the latter student, the knowledge states circle around and bounce back and forth in the space. We also show four selected submissions by each student during their response process, further illustrating how the first student made steady progress, i.e., adding the return statement (first two code submissions) and correcting logic (last two code submissions), and how the second student got stuck, i.e., making reasonable changes initially but then some repetitive edits.

B.5 Real-World Use Cases and Implications

One crucial highlight of our work is that, through OKT, we can predict students' responses to open-ended questions *before actually assigning them.* On the contrary, existing student and teacher support tools can only be applied *after* observing their responses. Therefore, OKT can be used in practice in many ways by anticipating student errors and struggles ahead of time. For example, for teacher support, we can use OKT to provide diagnosis information to teachers via a dashboard. For any open-ended question that the teacher wants to assign to their class, we can predict the responses that each student will write given their current knowledge states and show teachers clusters that represent typical errors. This way, the teacher can anticipate student performance, switch to an easier (or more challenging) question if necessary, and prepare feedback for individual students ahead of time. For student support, if a student struggles, we can use OKT to find other students stuck in a similar place but ultimately succeeded in answering the question and provide incremental hints or feedback on their errors. These advantages over traditional KT methods will potentially enable OKT to become the next-generation workhorse for large-scale, intelligent educational systems.

Appendix C

C.1 Notation

L, ℓ	Total number of layers: $L \geq 0$; Index of the layer in an RNN: $\ell \in \{0, \cdots, L\}$
T, t	Total number of time steps: $T \geq 0$; Index of time steps of an RNN: $t \in \{0, \cdots, T\}$
C, c	Total number of output classes: $C > 0$; Index of the output class: $c \in \{1, \cdots, C\}$
N, n	total number of examples: $N \geq 1$; Index of example in a dataset: $n \in \{1, \cdots, N\}$
$R^{(\ell)}$	Index of the partition region induced by the piecewise nonlinearity at layer l
$D^{(\ell)}$	Dimension of input to the RNN at layer ℓ
K, k	Total number of output dimensions of a MASO, $K \geq 0$; Index of MASO output dimension, $k \in \{1, \cdots, K\}$
Q	The partition region selection matrix
$\boldsymbol{x}^{(t)}$	t^{th} time step of a discrete time-serie, $\boldsymbol{x}^{(t)} \in \mathbb{R}^{D^{(0)}}$
$\boldsymbol{x}$	Concatenation of the whole length T time-serie: $\boldsymbol{x} = \left[\boldsymbol{x}^{(1)\top}, \cdots, \boldsymbol{x}^{(T)\top}\right]^\top$, $\boldsymbol{x} \in \mathbb{R}^{D^{(0)}T}$
$\boldsymbol{X}$	A dataset of N time-series: $\boldsymbol{X} = \{\boldsymbol{x}_n\}_1^N$
$\widehat{y}(\boldsymbol{x})$	Output/prediction associated with input $\boldsymbol{x}$
y_n	True label (target variable) associated with the nth time-serie example $\boldsymbol{x}_n$. For classification $y_n \in \{1, \ldots, C\},\ C > 1$; For regression $y_n \in \mathbb{R}^C, C \geq 1$
$\boldsymbol{h}^{(\ell,t)}$	Output of an RNN cell at layer ℓ and time step t; Alternatively, input to an RNN cell at layer $\ell + 1$ and time step $t - 1$
$\boldsymbol{h}^{(\ell)}$	Concatenation of hidden state $\boldsymbol{h}^{(\ell,t)}$ of all time steps at layer ℓ: $\boldsymbol{h}^{(\ell)} = \left[\boldsymbol{h}^{(\ell,1)\top}, \cdots, \boldsymbol{x}^{(\ell,T)\top}\right]^\top$, $\boldsymbol{h}^{(\ell)} \in \mathbb{R}^{D^{(\ell)}T}$
$\boldsymbol{z}^{(\ell,t)}$	Concatenated input to an RNN cell at layer ℓ and time step t: $\boldsymbol{z}^{(\ell,t)} = \left[\boldsymbol{h}^{(\ell-1,t)\top}, \boldsymbol{h}^{(\ell,t-1)\top}\right]^\top$, $\boldsymbol{z}^{(\ell,t)} \in \mathbb{R}^{2D^{(\ell)}}$
$\boldsymbol{W}_r^{(\ell)}$	ℓth layer RNN weight associated with the input $\boldsymbol{h}^{(\ell,t-1)}$ from the previous time step: $\boldsymbol{W}_r^{(\ell)} \in \mathbb{R}^{D^{(\ell)} \times D^{(\ell)}}$
$\boldsymbol{W}^{(\ell)}$	ℓth layer RNN weight associated with the input $\boldsymbol{h}^{(\ell-1,t)}$ from the previous layer: $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{D^{(\ell)} \times D^{(\ell-1)}}$
$\boldsymbol{W}$	Weight of the last fully connected layer: $\boldsymbol{W} \in \mathbb{R}^{C \times D^{(L)}}$
$\boldsymbol{b}^{(\ell)}$	ℓth layer RNN bias: $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{D^{(\ell)}}$
$\boldsymbol{b}$	Bias of the last fully connected layer: $\boldsymbol{b} \in \mathbb{R}^C$
$\sigma(\cdot)$	Pointwise nonlinearity in an RNN (assumed to be piecewise affine and convex in this work)
$\sigma_{\boldsymbol{\epsilon}}$	Standard deviation of noise injected into the initial hidden state $\boldsymbol{h}^{(\ell,0)}\ \forall \ell$
$A_\sigma^{(\ell,t)}$	MASO formula of the RNN activation $\sigma(\cdot)$ at layer ℓ and time step t: $A_\sigma \in \mathbb{R}^{D^{\ell} \times D^{(\ell)}}$
$A^{(\ell,t)}$, $B^{(\ell,t)}$	MASO parameters of an RNN at layer ℓ and time step t: $A^{(\ell,t)} \in \mathbb{R}^{D^{(\ell)} \times R^{(\ell)} \times D^{(\ell-1)}}$, $B^{(\ell,t)} \in \mathbb{R}^{D^{(\ell)} \times R^{(\ell)}}$

C.2 Datasets and Preprocessing Steps

Below we describe the datasets and explain the preprocessing steps for each dataset.

MNIST. The dataset[5] consists of 60k images in the training set and 10k images in the test set. We randomly select 10k images from the training set as validation set. We flatten each image to a 1-dimensional vector of size 784. Each image is also centered and normalized with mean of 0.1307 and standard deviation of 0.3081 (PyTorch default values).

permuted MNIST. We apply a fixed permutation to all images in the MNIST dataset to obtain the permuted MNIST dataset.

SST-2. The dataset[6] consists of 6920, 872, 1821 sentences in the training, validation and test set, respectively. Total number of vocabulary is 17539, and average sentence length is 19.67. Each sentence is minimally processed into sequences of words and use a fixed-dimensional and trainable vector to represent each word. We initialize these vectors either randomly or using GloVe [255]. Due to the small size of the dataset, the phrases in each sentence that have semantic labels are also used as part of the training set in addition to the whole sentence during training. Dropout of 0.5 is applied to all experiments. Phrases are not used during validation and testing, i.e., we always use entire sentences during validation and testing.

Bird Audio Dataset. The dataset[7] consists of $7,000$ field recording signals of 10 seconds sampled at 44 kHz from the Freesound [313] audio archive representing slightly less than 20 hours of audio signals. The audio waveforms are extracted from diverse scenes such as city, nature, train, voice, water, etc., some of which include bird sounds. The labels regarding the bird detection task can be found in the file `freefield1010`. Performance is

[5]`http://yann.lecun.com/exdb/mnist/`

[6]`https://nlp.stanford.edu/sentiment/index.html`

[7]`http://machine-listening.eecs.qmul.ac.uk/bird-audio-detection-challenge/`

measured by Area Under Curve (AUC) due to the unbalanced distribution of the classes. We preprocess every audio clip by first using short-time Fourier transform (STFT) with 40ms and 50% overlapping Hamming window to obtain audio spectrum and then by extracting 40 log mel-band energy features. After preprocessing, each input is of dimension $D = 96$ and $T = 999$.

C.3 Experimental Setup

Experiment setup for various datasets is summarized in Table C.5. Some of the experiments do not appear in the main text but in the appendix; we include setup for those experiments as well. A setting common to all experiments is that we use learning rate scheduler so that when validation loss plateaus for 5 consecutive epochs, we reduce the current learning rate by a factor of 0.7.

Setup of the experiments on influence of various standard deviations in random initial hidden state under different settings. We use σ_ϵ chosen in $\{0.001, 0.01, 0.1, 1, 5\}$ and learning rates in $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1.5 \times 10^{-4}, 2 \times 10^{-4}\}$ for RMSprop and $\{1 \times 10^{-7}, 1 \times 10^{-6}, 1.25 \times 10^{-6}, 1.5 \times 10^{-6}\}$ plain SGD.

Setup of input space partitioning experiments. For the results in the main text, we use t-SNE visualization [331] with 2 dimensions and the default settings from the python `sklearn` package. Visualization is performed on the whole 10k test set images. For finding the nearest neighbors of examples in the SST-2 dataset, since the examples are of varying lengths, we constrain the distance comparison to within +/-10 words of the target sentence. When the sentence lengths are not the same, we simply pad the shorter ones to the longest one, then process it with RNN and finally calculate the distance as the ℓ_2 distance of the partition codes (i.e., concatenation of all hidden states) that RNN computes. We justify the comparison between examples of different lengths using padding by noting that batching

Table C.5 : Various experiment setup. Curly brackets indicate that we attempted more than one value for this experiment. p-MNIST stands for permuted MNIST.

Settings	Dataset				
	Add task	MNIST	p-MNIST	SST-2	Bird detection
RNN type	ReLU RNN	ReLU RNN	ReLU RNN	ReLU RNN	GRU
#Layers	$\{1, 2\}$	$\{1, 2\}$	$\{1, 2\}$	$\{1, 2\}$	2
Input size	2	1	1	128	96
Hidden size	128	128	128	300	256
Output size	3	3	10	2	2
Initial Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4
Optimizer	RMSprop	{RMSprop, SGD}	RMSprop	Adam	Adam
Batch size	50	64	64	64	64
Epochs	100	200	200	100	50

examples and padding the examples to the longest example within a batch has been a common practice in modern natural language processing tasks.

Setup of exploratory experiments. We experimented with one-layer GRU with 128 hidden unites for MNIST and permuted MNIST datasets. We use RMSprop optimizer with an initial learning rate of 0.001. We experimented with various standard deviations in random initial hidden state including $\{0.01, 0.05, 0.1, 0.5, 1, 5\}$. The optimal standard deviations that produce the results in the main text are $\sigma_\epsilon = \{0.01, 0.05, 0.01\}$, for MNIST, permuted MNIST and bird detection datasets, respectively.

C.4 Additional Input Space Partition Visualizations

We provide ample additional visualizations to demonstrate the partition codes that an RNN computes on its input sequences. Here, the results are focused more on the properties of the final partition codes computed after the RNN processes the entire input sequence rather than

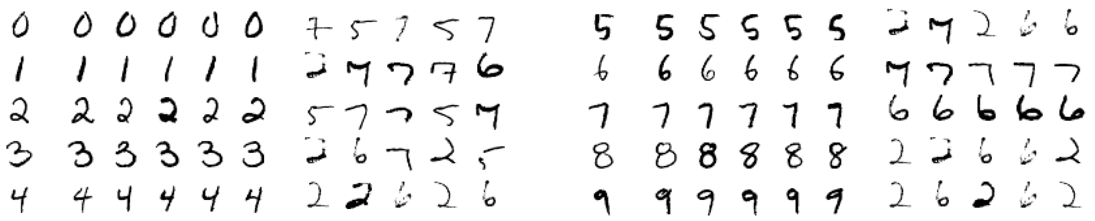


Figure C.5 : Visualization of partition codes for pixel-by-pixel (i.e., flattened to a 1-dimensional, length 784 vector) MNIST dataset using a trained ReLU RNN (one layer, 128-dimensional hidden state). Here, we visualize the nearest 5 and farthest 5 images of one selected image from each class. The distance is computed using the partition codes of the images. Leftmost column is the original image; the middle 5 images are the 5 nearest neighbors; the rightmost 5 images are the farthest neighbors.

part of the input sequence. Several additional sets of experimental results are shown; the first three on MNIST and the last one on SST-2.

First, we visualize the nearest and farthest neighbors of several MNIST digits in Figure C.5. Distance is calculated using the partition codes of the images. The left column is the original image; The next five columns are the five nearest neighbors to the original image; The last five columns are five farthest neighbors. This figure shows that partition codes the images are well clustered.

Second, we show the two dimensional projection using t-SNE of the raw pixel and VQ representations of each data points in the MNIST dataset and visualize them in Figure C.6. We clearly see a more distinct clustering using VQ representation of the data than using the raw pixel representation. This comparison demonstrate the ability of the RNN to extract useful information from the raw representation of the data in the form of VQ.

Third, we perform a KNN classification with $k \in \{1, 2, 5, 10\}$ using 1) the RNN computed partition codes of the inputs and 2) raw pixel data representation the MNIST test set to illustrate that the data reparametrized by the RNN has better clustering property than the original data representations. We use 80% of the test set to train the classifier and

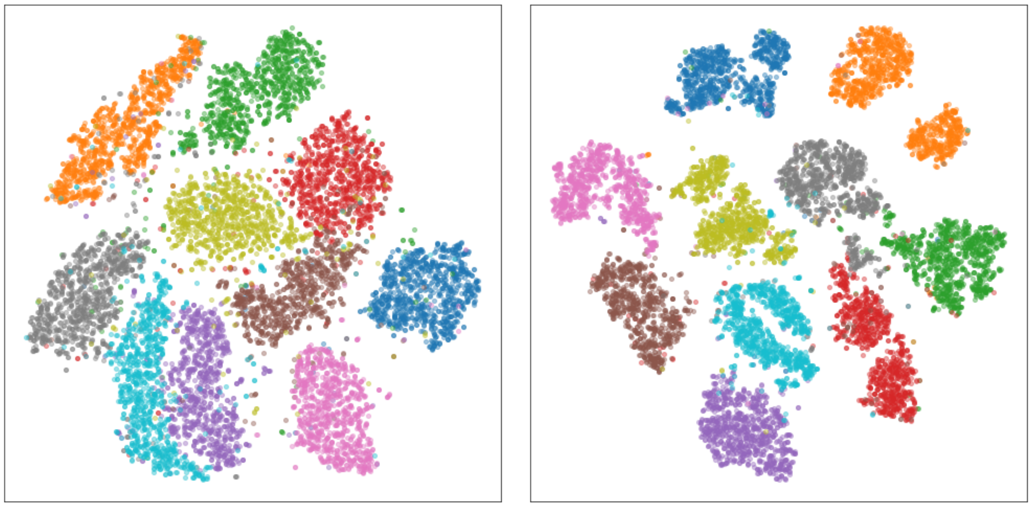


Figure C.6 : t-SNE visualization of MNIST test set images using raw pixel representation (**left**) and RNN VQ representation (**right**). We see more distinct clusters in the t-SNE plot using RNN VQ representation of images than the raw pixel representation, implying the useful information that RNN extracts in the form of VQ.

Table C.6 : K-nearest neighbor classification accuracies using data reparametrized by RNN compared to those using raw pixel data. We can see that classification accuracies using RNN reparametrized data are much higher than those using raw pixel data for all k's.

k	MNIST, raw pixels	MNIST, VQ
1	0.950	**0.977**
2	0.936	**0.974**
5	0.951	**0.977**
10	0.939	**0.975**

the rest for testing. The results are reported in Table C.6. We see that the classification accuracies when using RNN computed partition codes of the inputs are significantly higher than those when using raw pixel representations. This result again shows the superior quality of the input space partitioning that RNN produces, and may suggest a new way to improve classification accuracy by just using the reparametrized data with a KNN classifier.

Finally, we visualize the 5 nearest and 5 farthest neighbors of a selected sentence from the SST-2 dataset to demonstrate that the partitioning effect on dataset of another modality. Again, the distances are computed using the partition codes of the inputs. The results are shown in Figure C.7. We can see that all sentences that are nearest neighbors are of similar sentiment to the target sentence, whereas all sentences that are farthest neighbors are of the opposite sentiment.

C.5 Additional Template Visualizations

We provide here more templates on images and texts in Figures C.8 and C.9. Notice here that, although visually the templates may look similar or meaningless, they nevertheless have meaningful inner product with the input. The class index of the template that produces the largest inner product with the input is typically the correct class, as can be seen in the two figures.

C.6 Additional Experimental Results for Random Initial Hidden State

C.6.1 Regularization Effect for Regression Problem

We present the regularization effect on adding task formulated as a regression problem, following setup in [10]. Result is shown in Figure C.10. We see regularization effect similar to that presented in Figure 9.4, which demonstrates that the regularization effect does indeed happens for both classification and regression problems, as Thm. 3 suggests.

C.6.2 Choosing Standard Deviation in Random Initial Hidden State

Table C.7 shows the classification accuracies under various settings. The discussion of the results is in Section 9.6.

Original text	
It is a film that will have people walking out halfway through , will encourage others to stand up and applaud , and will , undoubtedly , leave both camps engaged in a ferocious debate for years to come . (+)	
Nearest 5 neighbors	**Farthest 5 neighbors**
Well-written , nicely acted and beautifully shot and scored , the film works on several levels , openly questioning social mores while ensnaring the audience with its emotional pull . (+, 22.00)	Marries the amateurishness of The Blair Witch Project with the illogic of Series 7 : The Contenders to create a completely crass and forgettable movie . (-, 37.60)
A stunning piece of visual poetry that will , hopefully , be remembered as one of the most important stories to be told in Australia 's film history . (+, 22.23)	K-19 may not hold a lot of water as a submarine epic , but it holds even less when it turns into an elegiacally soggy Saving Private Ryanovich . (-, 37.42)
Cute , funny , heartwarming digitally animated feature film with plenty of slapstick humor for the kids , lots of in-jokes for the adults and heart enough for everyone . (+, 22.61)	This is a great subject for a movie , but Hollywood has squandered the opportunity , using it as a prop for warmed-over melodrama and the kind of choreographed mayhem that director John Woo has built his career on . (-, 37.24)
Though it is by no means his best work , Laissez-Passer is a distinguished and distinctive effort by a bona-fide master , a fascinating film replete with rewards to be had by all willing to make the effort to reap them . (+, 22.78)	Flotsam in the sea of moviemaking , not big enough for us to worry about it causing significant harm and not smelly enough to bother despising . (-, 37.15)
An absorbing trip into the minds and motivations of people under stress as well as a keen , unsentimental look at variations on the theme of motherhood . (+, 22.89)	If you 're not a prepubescent girl , you 'll be laughing at Britney Spears ' movie-starring debut whenever it does n't have you impatiently squinting at your watch . (-, 36.98)

Figure C.7 : Nearest and furthest neighbors of a postive movie review. The sentiment (+ or -) and the euclidean distance between the input and the neighbor vector quantizations are shown in parenthesis after each neighbor.

C.7 Proofs

C.7.1 Proof of Thm. 1

To simplify notation, similar to the main text, in the proof here we drop the affine parameters' dependencies on the input, but keep in mind the input-dependency of these parameters.

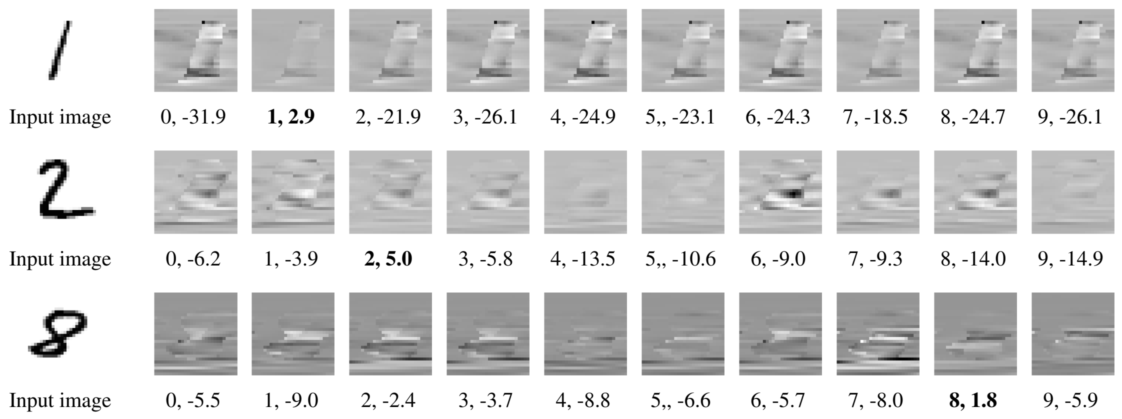


Figure C.8 : Templates of three selected MNIST images. The leftmost column is the original input image. The next ten images of each row are the ten templates of a particular input image corresponding to each class. For each template image, we show the class and the inner product of this template with the input. Text under the template of the true class of each input image is bolded.

We first derive the expression for a hidden state $\boldsymbol{h}^{(\ell,t)}$ at a given time step t and layer ℓ. Using Prop. 2, we start with unrolling the RNN cell of layer ℓ at time step t for two time steps to $t-2$ by recursively applying (9.1) as follows:

$$\begin{aligned}
\boldsymbol{h}^{(\ell,t)} &= \sigma\left(\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1,t)} + \boldsymbol{b}^{(\ell)} + \boldsymbol{W}_r^{(\ell)}\boldsymbol{h}^{(\ell,t-1)}\right) && (1)\\
&= A_\sigma^{(\ell,t)}\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1,t)} + A_\sigma^{(t)}\boldsymbol{b}^{(\ell)} + A_\sigma^{(t)}\boldsymbol{W}_r^{(\ell)}\boldsymbol{h}^{(\ell,t-1)} && (2)\\
&= A_\sigma^{(\ell,t)}\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1,t)} + A_\sigma^{(t)}\boldsymbol{b}^{(\ell)}\\
&\quad + A_\sigma^{(\ell,t)}\boldsymbol{W}_r^{(\ell)}\left(A_\sigma^{(\ell,t-1)}\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1,t-1)} + A_\sigma^{(\ell,t-1)}\boldsymbol{b}^{(\ell)} + A_\sigma^{(\ell,t-1)}\boldsymbol{W}_r^{(\ell)}\boldsymbol{h}^{(\ell,t-2)}\right) && (3)\\
&= \left(A_\sigma^{(\ell,t)}\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1,t)} + A_\sigma^{(\ell,t)}\boldsymbol{W}_r^{(\ell)}A_\sigma^{(\ell,t-1)}\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1,t-1)}\right)\\
&\quad + \left(A_\sigma^{(\ell,t)} + A_\sigma^{(\ell,t)}\boldsymbol{W}_r^{(\ell)}A_\sigma^{(\ell,t-1)}\right)\boldsymbol{b}^{(\ell)} + A_\sigma^{(\ell,t)}\boldsymbol{W}_r^{(\ell)}A_\sigma^{(\ell,t-1)}\boldsymbol{W}_r^{(\ell)}\boldsymbol{h}^{(\ell,t-2)}\,. && (4)
\end{aligned}$$

From (1) to (2), we use the result of Prop. 2. From (2) to (3), we expand the term that

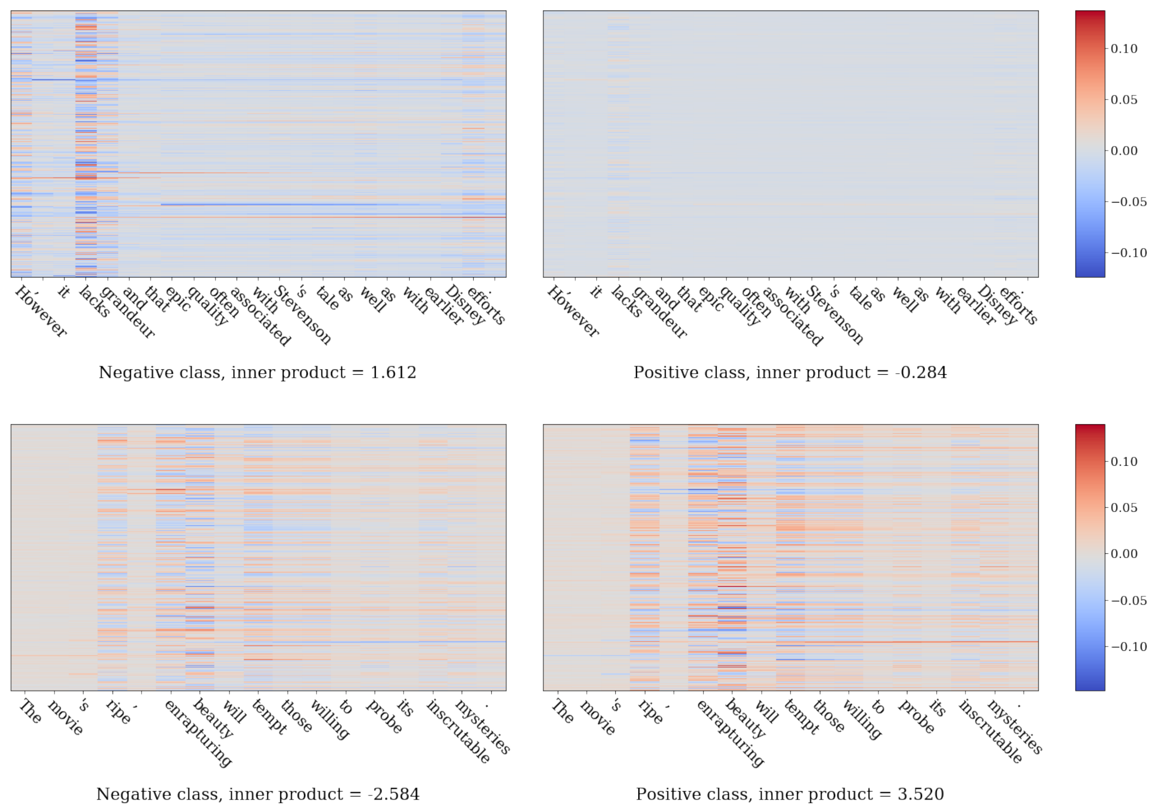


Figure C.9 : Additional template visualizations of an example from the SST-2 dataset. Each word in the sentence is marked as a tick label in the x axis. The values of inner products are marked below each template. The template that has the bigger inner product is the true class of the sentence. We see that the template corresponding to the correct class produces a significantly bigger inner product with the input than other templates.

involves the hidden state $\boldsymbol{h}^{(\ell,t-1)}$ at the previous time step recursively using (2). From (3) to (4), we group terms and write them compactly.

Now define $\mathcal{A}_{s:t}^{(\ell,s)} := \prod_{k=t}^{s+1} A_{\sigma}^{(\ell,k)} \boldsymbol{W}_r^{(\ell)}$ for $s < t$ and $\mathcal{A}_{t:t}^{(\ell,t)} := \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix. Using this definition, we rewrite (4) and proceed with the unrolling to the initial

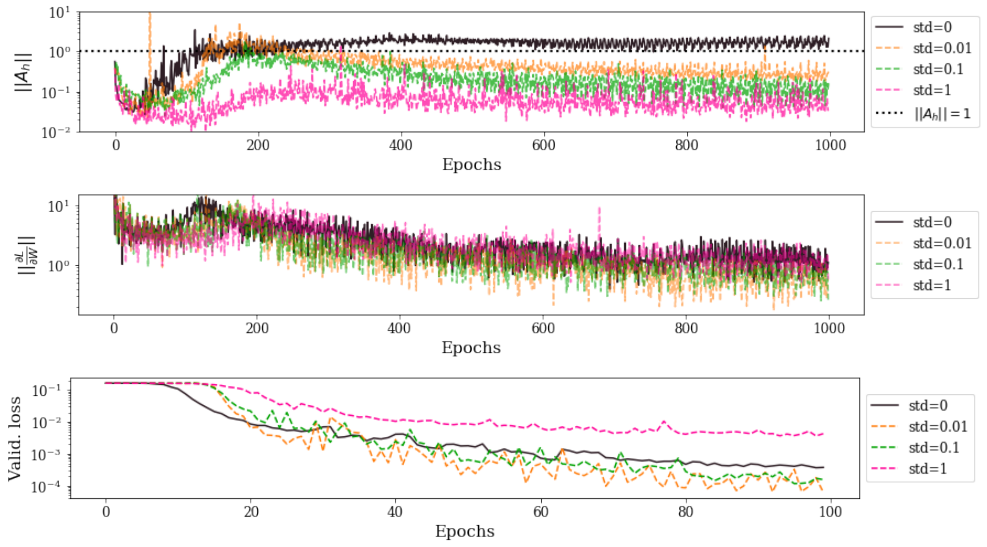


Figure C.10 : Various plots during training of add problem (T=100, regression). **Top**: norm of $\mathbf{A}_\mathrm{h}$ at every 100 iterations; **Middle**: norm of gradient of recurrent weight at every 100 iterations; **Bottom**: validation loss at every epoch. Each epoch contains 1000 iterations.

time step as follows:

$$\boldsymbol{h}^{(\ell,t)} = \begin{pmatrix} \mathcal{A}^{(\ell)}_{t:t} & \mathcal{A}^{(\ell)}_{t-1:t} \end{pmatrix} \begin{pmatrix} A^{(\ell,t)}_\sigma \boldsymbol{W}^{(\ell)} \\ A^{(\ell,t-1)}_\sigma \boldsymbol{W}^{(\ell)} \end{pmatrix} \begin{pmatrix} \boldsymbol{h}^{(\ell-1,t)} \\ \boldsymbol{h}^{(\ell-1,t-1)} \end{pmatrix} + \sum_{s=t}^{t-1} \mathcal{A}^{(\ell)}_{s:t} B^{(\ell,s)} \tag{5}$$

$$\cdots$$

$$= \begin{pmatrix} \mathcal{A}^{(\ell)}_{t:t} & \cdots & \mathcal{A}^{(\ell)}_{1:t} \end{pmatrix} \begin{pmatrix} A^{(\ell,t)}_\sigma \boldsymbol{W}^{(\ell)} \\ \vdots \\ A^{(\ell,1)}_\sigma \boldsymbol{W}^{(\ell)} \end{pmatrix} \begin{pmatrix} \boldsymbol{h}^{(\ell-1,t)} \\ \vdots \\ \boldsymbol{h}^{(\ell-1,1)} \end{pmatrix} + \sum_{s=t}^{t-1} \mathcal{A}^{(\ell)}_{s:t} B^{(\ell,s)} + \mathcal{A}^{(\ell)}_{0:t} \boldsymbol{h}^{(\ell,0)} , \tag{6}$$

where $B^{(\ell,s)} = A^{(\ell,s)}_\sigma \boldsymbol{b}^{(\ell)}$ as defined in Prop. 2.

Repeat the above derivation for $t \in \{1, \cdots, T\}$ and stack $\boldsymbol{h}^{(\ell,t)}$ in decreasing time steps

Table C.7 : Classification accuracy for MNIST dataset under 2 different optimizers, various learning rates and different standard deviation σ_ϵ in the random initial hidden state. Results suggest RMSprop tolerates various choices of σ_ϵ while SGD works for smaller σ_ϵ.

	RMSprop				**SGD**			
σ_ϵ	1e-5	1e-4	1.5e-4	2e-4	1e-7	1e-6	1.25e-6	1.5e-6
0	0.960	0.973	0.114	0.114	**0.837**	0.879	0.870	0.098
0.001	**0.963**	0.974	0.978	0.970	0.835	0.895	0.913	0.875
0.01	0.962	0.980	0.978	0.976	0.834	**0.898**	**0.922**	**0.918**
0.1	0.955	0.976	**0.981**	0.976	0.803	0.833	0.913	0.908
1	0.956	0.977	0.980	0.976	0.520	0.640	0.901	0.098
5	0.952	**0.981**	0.973	**0.981**	0.471	0.098	0.098	0.098

from top to bottom, we have:

$$\begin{pmatrix} \boldsymbol{h}^{(\ell,T)} \\ \vdots \\ \boldsymbol{h}^{(\ell,1)} \end{pmatrix} = \begin{pmatrix} \mathcal{A}^{(\ell)}_{T:T} & \dots & \mathcal{A}^{(\ell)}_{1:T} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathcal{A}^{(\ell)}_{1:1} \end{pmatrix} \begin{pmatrix} A^{(\ell,T)}_{\sigma} \boldsymbol{W}^{(\ell)} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & A^{(\ell,1)}_{\sigma} \boldsymbol{W}^{(\ell)} \end{pmatrix} \begin{pmatrix} \boldsymbol{h}^{(\ell-1,T)} \\ \vdots \\ \boldsymbol{h}^{(\ell-1,1)} \end{pmatrix}$$

$$+ \begin{pmatrix} \sum_{t=T}^{1} \mathcal{A}^{(\ell)}_{t:T} B^{(\ell,t)} + \mathcal{A}^{(\ell)}_{0:T} \boldsymbol{h}^{(\ell,0)} \\ \vdots \\ \mathcal{A}^{(\ell)}_{1:1} B^{(\ell,t)} + \mathcal{A}^{(\ell)}_{0:1} \boldsymbol{h}^{(\ell,0)} \end{pmatrix} = A^{(\ell)}_{\mathrm{RNN}} \boldsymbol{h}^{(\ell-1)} + B^{(\ell)}_{\mathrm{RNN}} \,, \quad (7)$$

where

$$A^{(\ell)}_{\mathrm{RNN}} = \begin{pmatrix} \mathcal{A}^{(\ell)}_{T:T} & \dots & \mathcal{A}^{(\ell)}_{1:T} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathcal{A}^{(\ell)}_{1:1} \end{pmatrix}, \; B^{(\ell)}_{\mathrm{RNN}} = \begin{pmatrix} \sum_{t=T}^{1} \mathcal{A}^{(\ell)}_{t:T} B^{(\ell,t)} + \mathcal{A}^{(\ell)}_{0:T} \boldsymbol{h}^{(\ell,0)} \\ \vdots \\ \mathcal{A}^{(\ell)}_{1:1} B^{(\ell,t)} + \mathcal{A}^{(\ell)}_{0:1} \boldsymbol{h}^{(\ell,0)} \end{pmatrix}, \; \boldsymbol{h}^{(\ell-1)} = \begin{pmatrix} \boldsymbol{h}^{(\ell-1,T)} \\ \vdots \\ \boldsymbol{h}^{(\ell-1,1)} \end{pmatrix},$$

which concludes the proof.

Thm. 2 follows from the recursive application of the above arguments for each layer $\ell \in \{1, \cdots, L\}$.

C.7.2 Proof of Thm. 3

We prove for the case of multi-class classification problem with softmax output. The proof for the case of regression problems easily follows.

Let $\boldsymbol{a}_i$ be the i^{th} row of the input-dependent affine parameter $\mathcal{A}_h$ where $\mathcal{A}_h := \mathcal{A}_{1:T} = \prod_{s=T}^{1} A_\sigma^{(\ell,s)} \boldsymbol{W}_r^{(\ell)}$ (recall Section 9.6), $\boldsymbol{x}_n = [\boldsymbol{x}_n^{(1)^\top}, \cdots, \boldsymbol{x}_n^{(T)^\top}]^\top$ be the concatenation of the n^{th} input sequence of length T and c be the index of the correct class. We assume the amplitude of random initial hidden state is small so that the input-dependent affine parameter $\mathcal{A}_h$, which also depends on $\boldsymbol{h}^{(0)}$, does not change when using random $\boldsymbol{h}^{(0)}$. Also, let $\boldsymbol{z}_n = f_{\text{RNN}}(\boldsymbol{x}_n, \boldsymbol{h}^{(0)})$ be the overall RNN computation that represents (9.9).

We first rewrite the cross entropy loss with random initial hidden state $\widetilde{\mathcal{L}}_{\text{CE}} = \mathcal{L}_{\text{CE}}\left(\text{softmax}\left(f_{\text{RNN}}\left(\boldsymbol{x}_n, \boldsymbol{h}^{(0)} + \boldsymbol{\epsilon}\right)\right)\right)$ as follows:

$$\begin{aligned}
\widetilde{\mathcal{L}}_{\text{CE}} &= \frac{1}{N}\sum_{n=1}^{N} -\log\left(\text{softmax}\left(f_{\text{RNN}}\left(\boldsymbol{x}_n^{(1:T)}, \boldsymbol{h}^{(0)} + \boldsymbol{\epsilon}\right)\right)\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} -\log\left(\frac{\exp(z_{nc} + \boldsymbol{a}_c\boldsymbol{\epsilon})}{\sum_{j=1}^{C}\exp(z_{nj} + \boldsymbol{a}_j\boldsymbol{\epsilon})}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left\{ -z_{nc} - \boldsymbol{a}_c\boldsymbol{\epsilon} + \log\left(\sum_{j=1}^{C}\exp(z_{nj} + \boldsymbol{a}_j\boldsymbol{\epsilon})\right)\right\}. \qquad (8)
\end{aligned}$$

Taking the expectation of the $\widetilde{\mathcal{L}}$ with respect to the distribution of the random Gaussian vector that the initial hidden state is set to, we have

$$\mathbb{E}[\widetilde{\mathcal{L}}_{\text{CE}}] = \mathcal{L}_{\text{CE}} + \mathcal{R}, \qquad (9)$$

where

$$\mathcal{R} = \frac{1}{N}\sum_{n=1}^{N}\left\{\mathbb{E}\left[\log\left(\sum_{j=1}^{C}\exp(z_{nj}+\boldsymbol{a}_j\boldsymbol{\epsilon})\right)\right] - \log\left(\sum_{j=1}^{C}\exp(z_{nj})\right)\right\}. \tag{10}$$

We note that similar forms of (10) have been previously derived by [343].

We now simplify (10) using second-order Taylor expansion on $\boldsymbol{h}^{(0)}$ of the summation inside the log function. Define function $u(\boldsymbol{x}_n^{(1:T)}, \boldsymbol{h}^{(0)}) := \log(\sum_j \exp(z_{nj})) = \log(\sum_j \exp(f(\boldsymbol{x}_n^{(1:T)}, \boldsymbol{h}^{(0)})))$. Then, we can approximate (10) as follows:

$$\begin{aligned}
\mathcal{R} &\approx \frac{1}{N}\sum_{n=1}^{N}\left\{\mathbb{E}\left[u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big) + \frac{\mathrm{d}u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)}{\mathrm{d}\boldsymbol{h}^{(0)}}\boldsymbol{\epsilon} + \frac{1}{2}\boldsymbol{\epsilon}^\top\frac{\mathrm{d}u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)^2}{\mathrm{d}^2\boldsymbol{h}^{(0)}}\boldsymbol{\epsilon}\right] - u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)\right\} \\
&= \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\mathbb{E}\left[\mathrm{Tr}\left(\boldsymbol{\epsilon}^\top\frac{\mathrm{d}u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)^2}{\mathrm{d}^2\boldsymbol{h}^{(0)}}\boldsymbol{\epsilon}\right)\right] \\
&= \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\,\mathrm{Tr}\left(\frac{\mathrm{d}u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)^2}{\mathrm{d}^2\boldsymbol{h}^{(0)}}\mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top\right]\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\frac{\sigma_{\boldsymbol{\epsilon}}^2}{2}\,\mathrm{Tr}\left(\frac{\mathrm{d}u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)^2}{\mathrm{d}^2\boldsymbol{h}^{(0)}}\right),
\end{aligned} \tag{11}$$

where $\frac{\mathrm{d}u(\boldsymbol{x}_n,\boldsymbol{h}^{(0)})^2}{\mathrm{d}^2\boldsymbol{h}^{(0)}}$ is the Hessian matrix:

$$\begin{aligned}
\left[\frac{\mathrm{d}u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)^2}{\mathrm{d}^2\boldsymbol{h}^{(0)}}\right]_{il} = \frac{\mathrm{d}u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)^2}{\mathrm{d}\boldsymbol{h}_i^{(0)}\mathrm{d}\boldsymbol{h}_l^{(0)}} &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{h}_l^{(0)}}\underbrace{\frac{\exp(z_{ni})}{\sum_{j=1}^{C}\exp(z_{nj})}}_{\widehat{y}_{ni}}\boldsymbol{a}_i \\
&= \frac{\mathrm{d}\widehat{y}_{ni}}{\mathrm{d}z_{nl}}\frac{\mathrm{d}z_{nl}}{\boldsymbol{h}_l^{(0)}}\boldsymbol{a}_i = \widehat{y}_{ni}(\mathbb{1}_{i=l}-\widehat{y}_{nl})\left\langle\boldsymbol{a}_i,\boldsymbol{a}_l\right\rangle.
\end{aligned}$$

Then, we can write the trace term in (11) as follows:

$$\mathrm{Tr}\left(\frac{\mathrm{d}u\big(\boldsymbol{x}_n,\boldsymbol{h}^{(0)}\big)^2}{\mathrm{d}^2\boldsymbol{h}^{(0)}}\right) = \left\|\mathrm{diag}\left(\left[\frac{\mathrm{d}y_{ni}}{\mathrm{d}z_{nl}}\right]_{i=l}\right)\mathcal{A}_h\right\|^2.$$

As a result, using the above approximations, we can rewrite the loss with random initial state in (8) as:

$$\widetilde{\mathcal{L}}_{\mathrm{CE}} = \mathcal{L}_{\mathrm{CE}} + \frac{\sigma_\epsilon^2}{2N}\sum_{n=1}^{N}\left\|\mathrm{diag}\left(\left[\frac{\mathrm{d}y_{ni}}{\mathrm{d}z_{nl}}\right]_{i=l}\right)\mathcal{A}_h\right\|^2. \tag{12}$$

We see that this regularizer term does not dependent on the correct class index c of each data points.

C.8 Prior Work on the Exploding Gradient in RNNs

The problem of exploding gradients has been widely studied from different perspectives. First approaches have attempted to directly control the amplitude of the gradient through gradient clipping [251]. A more model driven approach has leveraged the analytical formula of the gradient when using specific nonlinearities and topologies in order to develop parametrization of the recurrent weights. This has led to various unitary reparametrizations of the recurrent weight [10, 372, 110, 111, 142, 225, 125, 145]. A soft version of such parametrization lies in regularization of the DNNs. This includes dropout applied to either the output layer [257] or hidden state [395], noisin [73], zoneout [165] and recurrent batch normalization [59]. Lastly, identity initialization of ReLU RNNs has been studied in [179] and [320]. Our results complements prior works in that simply using random initial hidden state instead of zero initial hidden state and without changing the RNN structure also relieves the exploding gradient problem by regularization the potentially largest term in the recurrent weight gradient.

Appendix D

D.1 Framework details

More details of the information fusion module (Eq. (10.1)) Recall that Eq.(10.1) is

$$\boldsymbol{e} = f_{\mathrm{CA}}(\boldsymbol{e}_{\mathrm{in}}, \boldsymbol{E}_r; \theta) = \mathrm{Attn}(\mathrm{Query}(\boldsymbol{e}_{\mathrm{in}}), \mathrm{Key}(\boldsymbol{E}_r)) \cdot \mathrm{Value}(\boldsymbol{E}_r)$$

where f_{CA} represents the cross attention function with parameters θ, and $\boldsymbol{e}_{\mathrm{in}} \in \mathbb{R}^{L \times D}$ and $\boldsymbol{E}_r \in \mathbb{R}^{(\sum_{k=1}^{K} L_k) \times D}$ are the input embedding and retrieved exemplar embeddings, respectively.

Concretely, Query, Key, and Value are affine functions that do not change the shape of the the input. The function input and output dimensions are as follows:

$$\mathrm{Query} : \mathbb{R}^{L \times D} \to \mathbb{R}^{L \times D}, \tag{13}$$

$$\mathrm{Key} : \mathbb{R}^{(\sum_{k=1}^{K} L_k) \times D} \to \mathbb{R}^{(\sum_{k=1}^{K} L_k) \times D}, \tag{14}$$

$$\mathrm{Value} : \mathbb{R}^{(\sum_{k=1}^{K} L_k) \times D} \to \mathbb{R}^{(\sum_{k=1}^{K} L_k) \times D} \tag{15}$$

In particular, the Key and Value functions first apply to each of the k-th retrieved molecule embedding $\boldsymbol{e}_r^k \in \mathbb{R}^{L_k \times D}$ (which is a block of matrix in the retrieved molecule embedding $\boldsymbol{E}_r$) and then stack these K output matrices horizontally to obtain the output matrices of shape $(\sum_{k=1}^{K} L_k) \times D$.

The Attn function first computes the inner product between each slice of the query output matrix (of shape D) and the key output matrix (of shape $(\sum_{k=1}^{K} L_k) \times D$), followed by a softmax function which results in $(\sum_{k=1}^{K} L_k)$ un-normalized weights. These weights are applied to the value output matrix (of shape $(\sum_{k=1}^{K} L_k) \times D$) to obtain one slice in the output fused matrix $\boldsymbol{e}$. The above procedure is performed in parallel to obtain the full output fused matrix $\boldsymbol{e}$, such that it maintains the same dimensionality with the input embedding $\boldsymbol{e}_{\text{in}}$, i.e., $\boldsymbol{e} \in \mathbb{R}^{L \times D}$.

More details of the CE **loss** The cross entropy in Eq. (2) is the maximum likelihood objective commonly used in sequence modeling and sequence-to-sequence generation tasks such as machine translation. It takes two molecules as its inputs: one is the ground-truth molecule sequence, and the other one is the generated molecule sequence (i.e., the softmax output of decoder). Specifically, if we define $y := x_{\text{1NN}}$ and $\hat{y} := \mathrm{Dec}(f_{\text{CA}}(e_{\text{in}}, E_r); \theta)$, then the "CE" in Eq. (8.11) is given as follows:

$$\text{CE}(\hat{y}, y) = -\sum_{l=0}^{L-1} \sum_{v=0}^{V-1} y_{l,v} \log \hat{y}_{l,v}$$

where L is the molecule sequence length, V is the vocabulary size, $y_{l,v}$ is the ground-truth (one-hot) probability of vocabulary entry v on the l-th token, and $\hat{y}_{l,v}$ is the predicted probability (i.e., softmax output of decoder) of the vocabulary entry v on the l-th token.

More details of the RetMol training objective (Sec. 10.2.2) We first provide more detailed descriptions of our proposed objective: Given an input molecule, we use its nearest neighbor molecule from the retrieval database, based on the cosine similarity in the embedding space of the CDDD encoder, as the prediction target. The decoder takes in the fused embeddings from the information fusion module to generate new molecules. As shown in Eq. (2), we calculate the cross-entropy distance between the decoded output and the nearest neighbor molecule as the training objective.

The motivation is that: Other similar molecules (i.e., the remaining $K - 1$ nearest neighbors) from the retrieval database, through the fusion module, can provide good guidance for transforming the input to its nearest neighbor (e.g., how to perturb the molecule and how large the perturbation would be). Accordingly, the fusion module can be effectively updated through this auxiliary task.

On the contrary, if we use the conventional encoder-decoder training objective (i.e., reconstructing the input molecule), the method actually does not need anything from the retrieval database to do well in the input reconstruction (as the input molecule itself contains all the required information already). As a result, the information fusion module would not be effectively updated during training.

Molecule retriever Algorithm 2 describes the how the retriever retrieves exemplar molecules from a retrieval database given the design criteria.

Inference Algorithm 3 describes the inference procedure.

Parameters The total number of parameters in RetMol and in the base generative model [127] is 10471179 and 10010635, respectively. The additional 460544 parameters come exclusively from the information fusion model, which means that it adds only less than 5% parameter overhead and is very lightweight compared to the base generative model.

Algorithm 2: Exemplar molecule retriever

```
Require : Property predictors $a_\ell$ and desired property thresholds $\delta_\ell$ for
          $\ell \in [1, \dots, L]$ for property constraints; scoring function $s$ for properties to
          be optimized; retrieval database $\mathcal{X}_R$; number of retrieved exemplar
          molecules $K$
Input   : Input molecule $x_{\text{in}}$
Output  : Retrieved exemplar molecules $\mathcal{X}_r$
1  $\ell' = L$;
2  $\mathcal{X}' = \cap_{\ell=1}^{L} \{x \in \mathcal{X}_R \mid a_\ell(x) \geq \delta_\ell\}$;
3  while $|\mathcal{X}'| \leq K$ do
4      $L := L - 1$;
5      $\ell' := L$;
6      $\mathcal{X}' := \cap_{\ell=1}^{L} \{x \in \mathcal{X}_R \mid a_\ell(x) \geq \delta_\ell\}$;
7  $\mathcal{X}'' = \{x \in \mathcal{X}' \mid a_{\ell'}(x) \geq a_{\ell'}(x_{\text{in}})\}$;
8  if $|\mathcal{X}''| \geq K$ then
9      return $\mathcal{X}_r = \text{top}_K(\mathcal{X}'', s)$;
10 else
11     return $\mathcal{X}_r = \text{top}_K(\mathcal{X}', s)$;
```

D.2 Detailed experiment setup

D.2.1 RetMol training

We only train the information fusion model in our RetMol framework. The training dataset uses either ZINC250k [126] (for the experiments in Section 3.1 or CheMBL [88]. The choice of the training data is to align with the setup for the baseline methods in each experiment; The experiments in Sections 3.1 and 3.3 use the ZINC250k dataset while the experiments in Sections 3.2 and 3.4 use the CheMBL dataset. For the ZINC250k dataset, we follow the train/validation/test splits in [140] and train on the train split. For the ChemMBL dataset, we train on the entire dataset without splits. Training is distributed over four V100 NVIDIA GPUs, each with 16GB memory, with a batch size of 256 samples on each GPU, for 50k iterations. The total training time is approximately 2 hours for either the ZINC250k or the

Algorithm 3: Generation with adaptive input and retrieval database update

```
Require : Encoder Enc, decoder Dec, information fusion model $f_{\text{CA}}$, exemplar
          molecule retriever Ret, property predictors $a_\ell(x)$ and desired property
          thresholds $\delta_\ell$ for $\ell \in [1, \ldots, L]$ for property constraints, retrieval database
          $\mathcal{X}_R$, scoring function $s(x)$ for properties to be optimized
Input   : Input molecule $x_{\text{in}}$, number of retrieved exemplar molecules $K$, number
          of optimization iterations $T$, the number of molecules $M$ to sample at
          each iteration
Output  : Optimized molecule $x'$
1  for $t \in [1, \ldots, T]$ do
2      $\boldsymbol{e}_{\text{in}} = \texttt{Enc}(x)$;
3      $\mathcal{X}_r = \texttt{Ret}\big(\mathcal{X}_R, \{a_\ell, \delta_\ell\}_{\ell=1}^{L}, s, x_{\text{in}}, K\big)$;
4      $\boldsymbol{E}_r = \texttt{Enc}(\mathcal{X}_r)$;
5      $\boldsymbol{e} = f_{\text{CA}}(\boldsymbol{e}_{\text{in}}, \boldsymbol{E}_r; \theta)$;
6      $\mathcal{X}_{\text{gen}} = \emptyset$;
7      for $i = 1, \ldots, M$ do
8          $\epsilon \sim \mathcal{N}(0, \mathbf{1})$;
9          $\boldsymbol{e}_i = \boldsymbol{e} + \epsilon$;
10         $x' = \texttt{Dec}(\boldsymbol{e}_i)$;
11         $\mathcal{X}_{\text{gen}} := \mathcal{X}_{\text{gen}} \cup \{x'\}$;
12     $\mathcal{X}_{\text{gen}} := \texttt{Ret}\big(\mathcal{X}_{\text{gen}}, \{a_\ell, \delta_\ell\}_{\ell=1}^{L}, s, x_{\text{in}}, K\big)$;
13     $x_{\text{in}} := \text{top}_1(\mathcal{X}_{\text{gen}}, s)$;
14     $\mathcal{X}_R := \mathcal{X}_R \cup \mathcal{X}_{\text{gen}} \setminus x_{\text{in}}$
```

CheMBL dataset. Our training infrastructure largely follows the Megatron[8] version of the molecule generative model in [127], which uses DeepSpeed,[9] Apex,[10] and half precision for improved training efficiency. Note that, once RetMol is trained, we do not perform further task-specific fine-tuning.

[8]`https://github.com/NVIDIA/Megatron-LM`

[9]`https://github.com/microsoft/DeepSpeed`

[10]`https://github.com/NVIDIA/apex`

D.2.2 RetMol inference

We use greedy sampling in the decoder throughout all experiments in this work. When we need to sample more than one molecule from the decoder given the (fused) embedding, we first perturb the (fused) embedding with independent random isotropic Gaussian with standard deviation of 1 and then generate a sample from each perturbed (fused) embedding using the decoder. Inference uses a single V100 NVIDIA GPU with 16 GB memory.

Note that during inference, the multi-property design objective is provided with a general form $s(x) = \sum_{l=1}^{L} w_l a_l(x)$. In the experiments, we simply set all the weight coefficients w_l to 1, i.e., the aggregated design criterion is the sum of individual property constraints.

D.2.3 Baselines

We briefly overview the existing methods and baselines that we have used for all experiments. Some baselines are applicable for more than one experiment while some specialize in a certain experiment.

JT-VAE [138] The junction tree variational autoencoder (JT-VAE) reconstructs a molecule in its graph (2 dimensional) representation using its junction tree and molecular graph as inputs. JT-VAE learns a structured, fixed-dimensional latent space. During inference, controllable generation is achieved by first performing optimization on the latent space via property predictors trained on the latent space and then generating a molecule via the decoder in the VAE.

MMPA [63] The Matched Molecular Pair Analysis (MMPA) platform uses rules and heuristics, such as matched molecular pair, to perform various molecule operations such as search, transformations, and synthesizing.

GCPN [389] Graph Convolutional Policy Network (GCPN) trains a graph convolutional neural network to generate molecules in their 2D graph representations with policy gradient reinforcement learning. The reward function consists of task-specific property predictors and adversarial loss.

Vseq2seq [12] This is a basic sequence-to-sequence model borrowed from the machine translation literature that translates the input molecule to an output molecule with more desired properties.

VJTNN [140] VJTNN is a graph-based method for generating molecule graphs based on junction tree variational autoencoders [138]. The method formalizes the controllable molecule generation problem as a graph translation problem and is trained using an adversarial loss to match the generation and data distribution.

HierG2G [139] The Hierarchical Graph-to-Graph generation method takes into account the molecule's substructures, which are interleaved with the molecule's atoms for more structured generation.

AtomG2G [139] The Atom Graph-to-Graph generation method is similar to HierG2G but only takes into account of the molecule's atoms information without the structure and substructure information.

DESMILES [214] The DESMILES method aims to translate the molecule's fingerprint into its SMILES representation. Controllable generation is achieved by fine-tuning the model for task-specific properties and datasets.

MolDQN [410] The Molecule Deep Q-Learning Network formalizes molecule generation and optimization as a sequence of Markov decision processes, which the authors use deep Q-learning and randomized reward function to optimize.

GA [239] This method augments the classic genetic algorithm with a neural network which acts as a discriminator to improve the diversity of the generation during inference.

QMO [118] The Query-based Molecule Optimization framework is a latent-optimization-based controllable generation method operated on the SMILES representation of molecules. Instead of finding a latent code in the latent space via property predictor trained on the latent space, QMO uses property predictor on the molecule space and performs latent optimization via zeroth order gradient estimation. Although this latent-optimization-based method removes the need to train latent-space property predictors, we find that it is sometimes challenging to tune the hyper-parameters to adapt QMO for different controllable generation tasks. For example, in the SARS-CoV-2 main protease inhibitor design task, we could not succeed to generate an optimized molecule using QMO even with extensive hyper-parameter tuning.

GVAE-RL [137] This method is the graph-based grammar VAE [171] that learns to expand a molecule with a set of expansion rules, based on a variational autoencoder. GVAE-RL further fine-tunes GVAE with RL objective for controllable generation, using the property values as the reward.

REINVENT [241] REINVENT trains a recurrent neural network using RL techniques to generate new molecules.

RationaleRL [137] The RationalRL method first assembles a “rationale”, a molecule fragment composed from different molecules with the desired properties. Then it trains a decoder using the assembled collection of rationales as input by first randomly sampling from the decoder, scoring each sample with the property predictors, and use the positive and negative samles as training data to fine-tune the decoder.

MARS [378] The MARS method models the molecule generation process as a Markov chain, where the transitions are the “edits” on the current molecule graph parametrized by a graph neural network. The generation process proceeds by either accepting the newly edited molecule if it has more desired attributes than the previous one. Otherwise, the previous molecule is kept and the Markov chain continues.

Graph MCTS and Graph GA [132] Graph MCTS and GA methods traverse the molecule space in its graph representation using genetic algorithm and Monte Carlo Tree Search algorithms, respectively.

SMILES LSTM [292] This method is a decoder (an LSTM) only method which generates molecules using a seed input symbol. Controllable generation is achieved by fine-tuning it on a task-specific dataset with RL using an property scores as the reward function.

SMILES GA [388] This method, similar to GA, applies various rules to the SMILES representation of molecules from a starting population.

D.2.4 QED and penalized logP experiments

To ensure a fair comparison with existing methods, we rely on using the same total number of calls to the property predictors. For example, an optimization process for an input molecule that runs for T iterations with M calls to the property predictors at each iteration will invoke a total of $T \times M$ property predictor calls. We use the same or less number of property predictor calls for each molecule's optimization.

QED experiment setup. When running RetMol, for each input molecule, we set the maximum number of iterations to 1000 and sample 50 molecules at each iteration, resulting in a total of $1000 \times 50 = 50,000$ property predictor calls. This number matches that in QMO [118], where the maximum number of optimization iterations is 20 with 50 "restarts" and 50 samples at each iteration for gradient estimation. Effectively, this results in a total of $20 \times 50 \times 50 = 50,000$ calls to the scoring function, which is the same as in our setting. We evaluate performance by success rate. For each input molecule, if a molecule that satisfy the design criteria (QED is above 0.9 and similarity with the input molecule is above 0.4), then we count it as a success. Success rate for all 800 input molecules is thus

$$\text{success rate} = \frac{\#\text{successful input molecules}}{800}.$$

Penalized logP experiment setup. For each input molecule, we run the optimization for 80 iterations and sample 100 molecules at each iteration, which is exactly the same setting as in QMO [118]. If optimization fails, i.e., no new molecules are generated that have higher penalized logP value than the input, then we set the relative improvement to 0. We evaluate performance by difference in penalized logP between the optimized and the input molecules, averaged over all 800 input molecules:

$$\text{avg. improvement} = \frac{1}{800} \sum_{i=1}^{800} \left(a_{\text{plogP}}(x'^{(i)}) - a_{\text{plogP}}(x_{\text{in}}^{(i)}) \right),$$

where a_{plogP} is the penalized logP predictor, x' is the generated molecule and x_{in} is the input molecule.

D.2.5 GSK3β + JNK3 + QED + SA experiment setup

For a fair comparison, we follow the same setup as in [137, 138] and make two major changes to RetMol's generative process. First, rather than starting with a known molecule, we first draw a random latent embedding as the starting point of the molecule generation process. 2) In the iterative refinement process, we generate only one molecule and directly use it as the input molecule in the next iteration; that is, in this experiment, at each iteration, we do not use the property predictors to select the best generated molecules because there is only one generated per iteration. For each input molecule, we run the optimization for 80 steps and generate one molecule at each step following [138]. Doing so results in a total of $80 \times 1 \times 3,700 = 296,000$ generated molecules, which is an order of magnitude less than the $550 \times 5,000 = 2,750,000$ number of generated molecules required for MARS [378]. This number in MARS remains high even if we change the number of samples at each iteration from $5,000$ to $3,700$ to align with the number of molecules evaluated in RetMol and the other baselines. In Table D.8, we provide the complete comparison of the competitive methods regarding the number of generated samples, where we can see that our method is sample efficient (with the best performance shown in Table 10.1b).

Unlike the previous tasks, there is no specification as to which molecule to use as the input molecule to be optimized. To obtain the input molecules, we first randomly select 3700 molecules from the CheMBL dataset [88], retrieve exemplar molecule randomly from the entire CheMBL dataset [88], and greedily generate one molecule using the random input and random retrieved exemplar molecules. This one generated molecule is the input to RetMol. We choose 3700 molecules to be optimized because 3700 is the number of molecules

Table D.8 : Number of generated molecules (or number of calls to the property prediction) for the competitive methods in the task of optimizing four properties: QED, SA, and two binding affinities to GSK3β and JNK3 estimated by pre-trained models from [137].

Method	**Number of generated samples**
RationaleRL [137]	1,086,000
MARS [378]	2,750,000
MolEvol [40]	200,000
RetMol	296,000

evaluated in existing methods reported in [137]. Note that the inputs to RetMol differ from those in existing methods. However, we believe our choice of input does not put our method in an advantage over existing methods and may even be at an disadvantage compared to existing methods. For example, the inputs to RationaleRL [137] are “rationales”, i.e., molecule fragments that already satisfy either the GSK3β or the JNK3 property and that are pieced together. These rationales are usually very close to the desired molecules. In contrast, the inputs to RetMol are mostly molecules with very low (close to 0) GSK3β and JNK3 values, making the optimization challenging.

Below, we show how to compute the metrics when using RetMol, which largely follows the computation in [138, 137]. For success rate, we count number of input molecules that result in at least one successful generated molecule (satisfy the four property constraints simultaneously) from all 80 molecules generated for that input molecule. If there exist more than one successful molecule for a given input, we choose the one with the least Tanimoto distance [13] with the input for the evaluation in the remaining metrics. For Novelty, we compute the percentage of the selected successful molecules that have less than 0.4 Tanimoto distance with any molecules in the retrieval database, i.e., those molecules that already satisfy the property constraints. For diversity, we compute the pairwise Tanimoto distance between each pair of the selected successful molecules.

D.2.6 Guacamol benchmark experiment setup

There are no known input molecules to any tasks in the Guacamol benchmark. Therefore, we first randomly select a population of molecules from the CheMBL dataset, which is the standard dataset in the benchmark. In this work, we choose five molecules as the starting population, although a larger population size is likely to yield better results at the higher computational cost. We perform iterative refinement for each molecule in the input population for 1000 iterations. After the optimization ends, we collect all the generated molecules from all molecules in the population and select the best N molecules, based on which the benchmark score is computed for each benchmark task. In all the MPO tasks in the benchmark, $N = 100$. The properties to be optimized and the scoring function details for each benchmark task are available in [29] and at `https://github.com/BenevolentAI/guacamol`. The scoring functions and evaluation protocol are provided in the benchmark. The benchmark provide an API which takes a population of input molecules and scoring functions to generated the optimized molecules. We implement RetMol using their API, which ensures a fair comparison and evaluation using the same evaluation protocol defined in the Guacamol benchmark.

D.2.7 SARS-CoV-2 main protease inhibitor design experiment setup

For RetMol, for each of the eight input inhibitors to be optimized, we run the iterative refinement for 10 iterations with 100 samples per iteration. For the graph GA baseline, for each of the eight inhibitors to be optimized, we use the same inhibitor as the initial population for crossover and mutation. We set the population and offspring size to 100 and run it for 10 iterations.

We use binding affinity between the generated inhibitor and the protein target. We computationally approximate binding affinity using Autodock-GPU, [11] which significantly speeds up docking compared to its CPU variants. The input files contains the receptor (target protein) of interest, the generated or identified ligands (inhibitor molecules), and a bounding box that indicates the docking site. For different docking experiments, the receptor and bounding box need to be prepared differently. Once the receptor and bounding box are prepared for a particular docking experiment, they are kept fixed and used for docking all ligands throughout the experiment. For the SARS-CoV-2 main protease docking experiments, we use the protein with PDB ID 7L11. We choose the bounding box by first docking a the protein with a known inhibitor ligand, Compound 5 [397], and then extract the bounding box configurations from its best docking pose [87]. The receptor, ligand, and bounding box preprocessing steps use the Autodock software suite.[12]

In addition, as we mentioned in the chapter associated with this Appendix, we also tested on QMO [118] with the following configurations: we set optimization iterations to 100, number of restarts to 1, weights for the property (binding affinity) to be optimized in $\{0.005, 0.01, 0.1, 0.25\}$, base learning rate in $\{0.05, 0.1\}$, and the property constraints the same as in those in RetMol. In our experiments, these configurations did not succeed in generating an optimized molecule given the constraints. We follow the official QMO codebase in these experiments. [13]

D.2.8 Analyses experiments: Training objectives

The purpose of this experiment is two-fold: 1) We want to show if only updating the fusion module while fixing the weights of the base generative model during training will cause a degradation in the quality of generated molecules; and 2) we want to check if our proposed

[11] `https://github.com/ccsb-scripps/AutoDock-GPU`

[12] `https://autodock.scripps.edu/`

[13] `https://github.com/IBM/QMO`

nearest neighbor objective can achieve better generation quality than the conventional language modeling objective (i.e., predicting the next token of input).

To evaluate RetMol, the unconditional generation procedure is the same as that in training the information fusion module, i.e., the retrieval database is the training split of the ZINC250 dataset, and the retrieval criterion is molecule similarity. Besides, since we mainly focus on evaluating the quality of generated molecules, we thus use validity, novelty and uniqueness as our metrics, which are standard in literature for evaluating molecule generation models' performance. The results suggest that our training objective that only updates the fusion module does not sacrifice the unconditioned molecule generation performance and even results in slight improvement on the novelty metric.

We perform the evaluation on the test split of the ZINC250k dataset. For each molecule in the test set, we generate 10 molecules by first randomly perturb the input to the decoder, i.e., an encoded (or fused) embedding matrix, 10 independent times using a isotropic random Gaussian with standard deviation of 1. Then, for validity, we compute the percentage of the 10 generated molecules that are valid according to RDKit, [14] averaged over all test molecules. For novelty, we compute the percentage of the 10 generated molecules that are not in the training split of the ZINC250k dataset, averaged over all molecules in the test set. For uniqueness, we compute the percentage of the 10 generated molecules that are unique, averaged over all molecules in the test set.

D.2.9 Remarks on number of iterations

In most experiments, we keep the number of iterative refinements in RetMol the same as the baseline iterative methods. For example, in Section 10.3.1, we use 80 optimization iteration steps for RetMol, which is the same as QMO. In Section 10.3.2, we use 80 optimization

[14] `https://www.rdkit.org/`

iteration steps, which is the same as JT-VAE. Note that in this experiment, RationaleRL does not require such iterative refinement process but does require extensive task-specific training and fine-tuning. In Section 10.3.3, the benchmark results do not mention the number of iterations and therefore we simply set the max number of iterations to 1k for RetMol. In Section 10.3.4, we use 20 iterations for both RetMol and Graph GA.

Our results also suggest that one does not need as many iterations as the baselines to achieve strong results. For example, we showed in Figure 3 (middle) that for experiments in Section 10.3.2, RetMol achieves better results than baselines with only 30 iterations.

D.3 Additional experiment results and analyses

D.3.1 QED and logP experiment

To visualize the property value improvements, we plot in Figure D.11 (i) the QED of the optimized (generated) molecules and of the input molecules under similarity constraint $\delta = 0.4$ and (ii) the penalized logP improvement between the generated and the input molecules under similarity constraints $\delta = \{0.4, 0.6\}$.

We can see that, for the penalized logP experiment, for similarity constraint $\delta = 0.4$, there are quite a few molecules with a large penalized logP value improvements. These molecules are the reason that the variance for RetMol in Table 1b is large. But even if we remove those molecules with extreme values, i.e., penalized logP bigger than 20, we still obtain an average improvement of 8.17 ± 4.12, which is still better than the best existing method. Furthermore, we also compute the mode of the penalized logP values for our method (i.e., the x-axis value of the largest peak in Figure D.11, Right), and we get 3.804 for $\sigma = 0.6$ and 6.389 for $\sigma = 0.4$. We can see our mode of the penalized logP values is still better than the mean in most of the baselines in Table 10.1b. For QMO, their mode looks

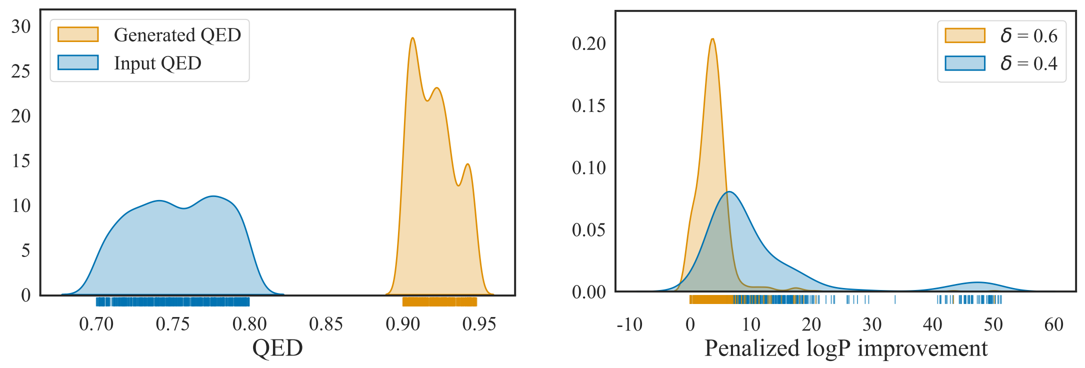


Figure D.11 : **Left**: distribution of QED values of the original and the optimized (generated) molecules under similarity constraint $\delta = 0.4$. **Right**: distribution of penalized logP improvement comparing similarity constraints $\delta = \{0.4, 0.6\}$.

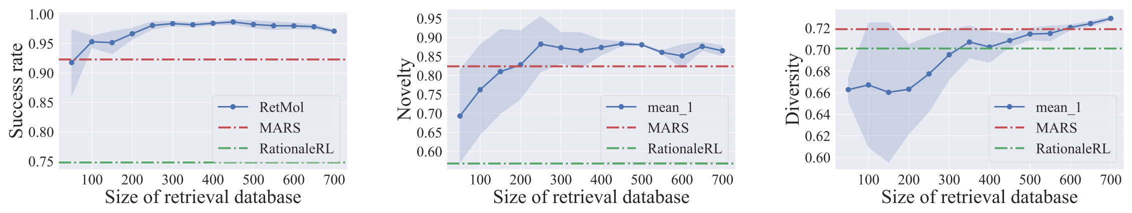


Figure D.12 : Generation performance with varying retrieval database size on the experiment in Section 3.2. Our framework achieves strong performance with as few as 100 molecules in the retrieval database and performance generally improves with increasing retrieval database size on all metrics.

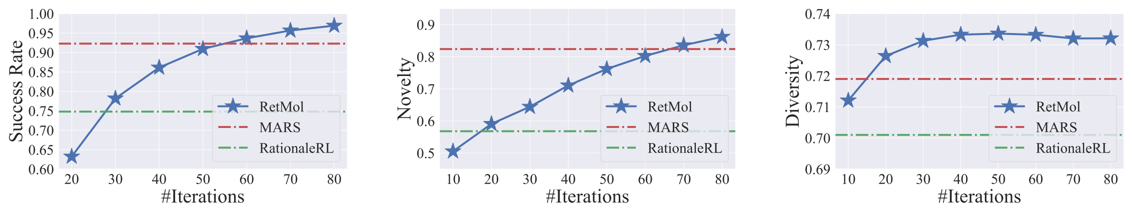


Figure D.13 : Generation performance with varying number of optimization iterations on the experiments in Section 3.2. Dashed lines are the success rates and novelty scores of the two best baselines. We observe that all the metrics improve as we increase the number of iterations.

very similar to ours from Figure 8 in their paper [118].

D.3.2 GSK3β and JNK3 experiment

In Figure D.14, we show 54 randomly chosen molecules generated by RetMol along with their highest similarity with the molecules in the retrieval database. These molecules demonstrate the diversity and novelty of the molecules that RetMol is capable of generating.

D.3.3 Guacamol experiment

We show in Table D.9 the detailed benchmark, SA, and QED results for all the MPO tasks in the Guacamol benchmark.

We additionally apply the functional group filters [15] to the optimized molecules of each method and compare the average number of molecules that pass the filters. Figure D.15 visualizes the results, which align with those in Sec. 10.3.3: RetMol strikes the best balance between optimizing benchmark score and maintaining a good functional group relevance. For example, Graph GA achieves the best benchmark performance but fails the filters more often than some of the methods under comparison. In contrast, RetMol achieves the second best benchmark performance and the highest number of passed molecules. Please see Table D.9 for the number of optimized molecules that pass the filters for each MPO task and for each method.

D.3.4 SARS-CoV-2 main protease inhibitor design experiment

Tables D.10 and D.11 visualizes the original and the RetMol optimized inhibitors along with the similarity map, Tanimoto distance, QED, SA, and docking (unit in `kcal/mol`) scores for both similarity constraint $\delta = \{0.6, 0.4\}$. We highlight the properties that the initial

[15] `https://github.com/PatWalters/rd_filters`

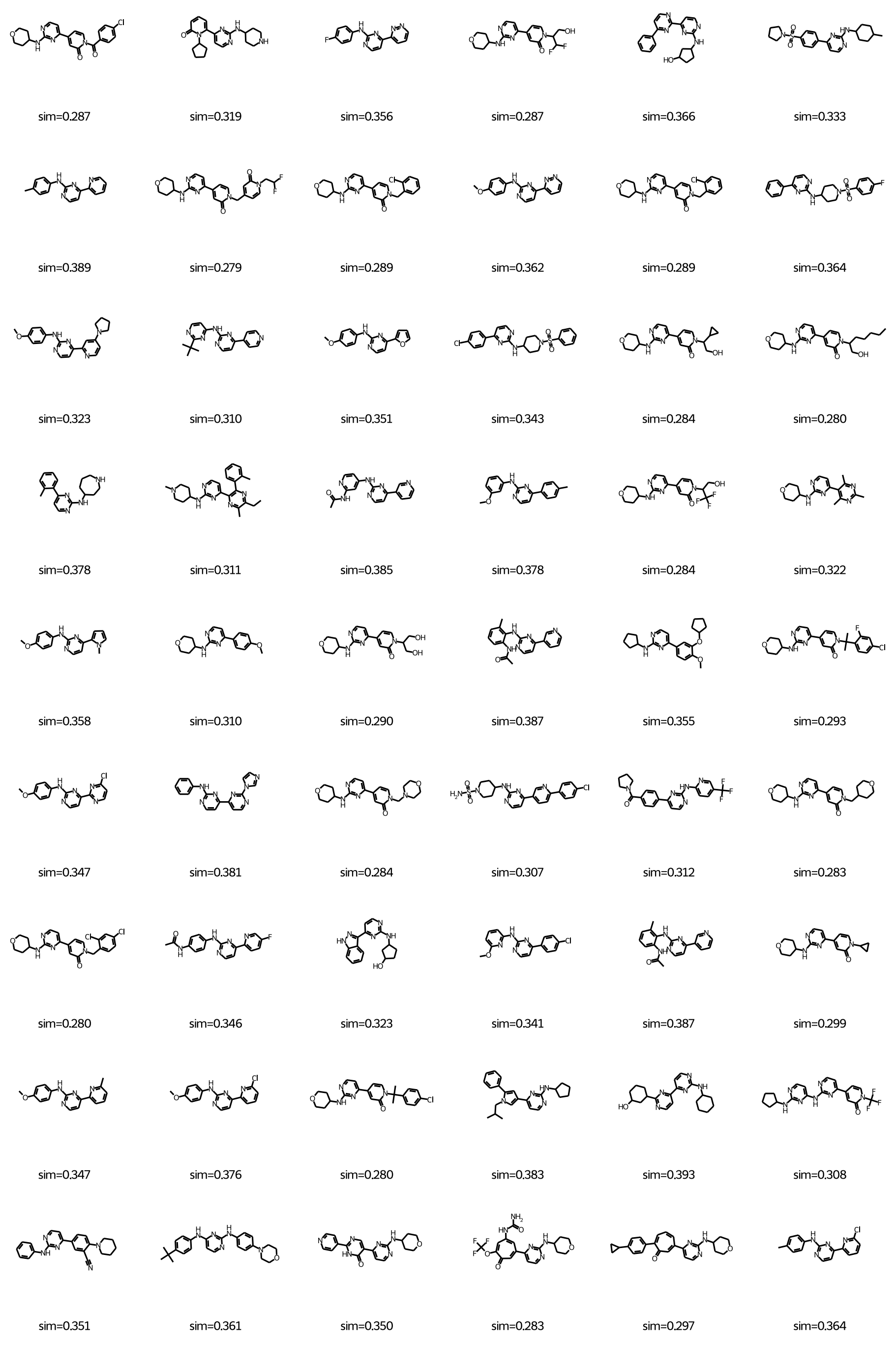


Figure D.14 : Visualizations of randomly chosen molecules generated by RetMol for the GSK3β + JNK3 + QED + SA experiment. Below each generated molecule, we show its highest similarity between each molecule in the retrieval database.

Table D.9 : Detailed results from the Guacamol MPO results. The tables from the top to the bottom are the benchmark results, averaged SA values, averaged QED values, and averaged numbers of generated molecules that pass the functional group filters, respectively. SA and QED values and the number of filter-passing molecules are averaged over all the 100 molecules evaluated in each MPO task. Bold and underline represent the best and the second best in each metric in each benchmark task, respectively.

Benchmarks	**SMILES GA**	**Graph MCTS**	**Graph GA**	**SMILES LSTM**	**Best of Dataset**	**Ours**
Osimertinib MPO	0.886	0.784	**0.953**	0.907	0.839	0.915
Fexofenadine MPO	0.931	0.695	**0.998**	0.959	0.817	0.969
Ranolazine MPO	0.881	0.616	0.920	0.855	0.792	**0.931**
Perindopril MPO	0.661	0.385	0.792	**0.808**	0.575	0.765
Amlodipine MPO	0.722	0.533	**0.894**	**0.894**	0.696	0.879
Sitagliptin MPO	0.689	0.458	**0.891**	0.545	0.509	0.735
Zaleplon MPO	0.413	0.488	**0.754**	0.669	0.547	0.713

Benchmarks - SA	**SMILES GA**	**Graph MCTS**	**Graph GA**	**SMILES LSTM**	**Best of Dataset**	**Ours**
Osimertinib MPO	6.386	3.901	3.357	2.923	**2.705**	3.061
Fexofenadine MPO	3.590	4.671	3.897	3.171	**3.097**	3.546
Ranolazine MPO	6.071	4.110	4.111	**2.900**	3.226	3.456
Perindopril MPO	5.343	**3.365**	4.286	4.017	3.645	4.276
Amlodipine MPO	4.717	3.529	3.575	3.329	**3.163**	3.431
Sitagliptin MPO	6.743	5.183	6.804	**2.794**	2.886	3.715
Zaleplon MPO	3.244	3.216	2.899	2.387	**2.294**	2.622

Benchmarks - QED	**SMILES GA**	**Graph MCTS**	**Graph GA**	**SMILES LSTM**	**Best of Dataset**	**Ours**
Osimertinib MPO	0.256	0.443	0.197	0.240	**0.478**	0.264
Fexofenadine MPO	0.207	0.495	0.309	0.335	**0.382**	0.301
Ranolazine MPO	0.096	**0.305**	0.095	0.113	0.129	0.112
Perindopril MPO	0.481	0.477	0.365	0.465	0.421	**0.546**
Amlodipine MPO	0.146	**0.582**	0.351	0.386	0.472	0.365
Sitagliptin MPO	0.254	0.453	0.086	0.700	**0.735**	0.701
Zaleplon MPO	0.206	0.679	0.562	**0.730**	0.712	0.753

Benchmarks - filters	**SMILES GA**	**Graph MCTS**	**Graph GA**	**SMILES LSTM**	**Best of Dataset**	**Ours**
Osimertinib MPO	0	**23**	0	0	19	0
Fexofenadine MPO	58	22	**73**	68	47	43
Ranolazine MPO	0	0	0	0	0	0
Perindopril MPO	12	29	42	60	40	**90**
Amlodipine MPO	4	31	56	**78**	49	58
Sitagliptin MPO	0	6	0	69	76	**82**
Zaleplon MPO	0	51	37	97	84	**98**

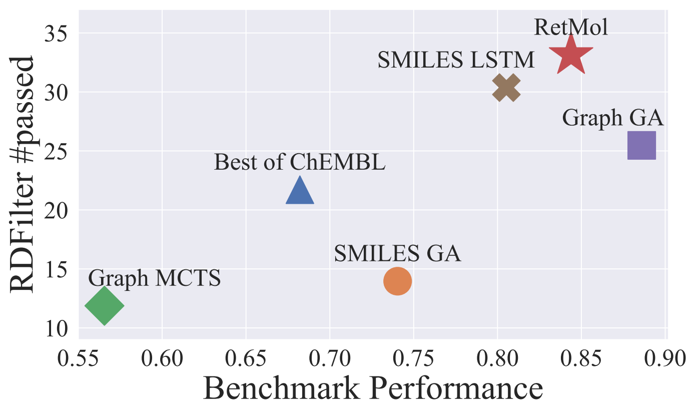


Figure D.15 : Average number of optimized molecules that pass the functional group filters against Guacamol benchmark scores. The results align with those in Sec. 10.3.3: RetMol strikes the best balance between optimizing benchmark score and maintaining a good functional group relevance.

inhibitor do not satisfy in red and the same properties in the optimized inhibitor in green. We can see that RetMol not only optimizes the docking score but also successfully improves the QED and SA scores for some inhibitors.

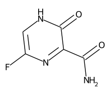
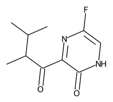
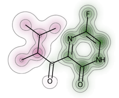
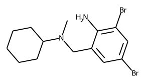
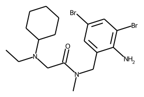
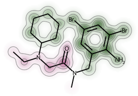
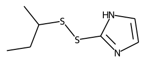
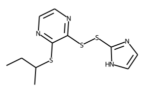
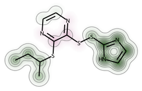
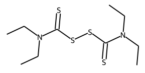
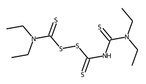
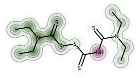
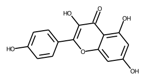
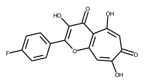
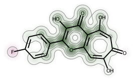
D.3.5 Antibacterial drug design for the MurD protein

In addition to the experiments above and in Chapter 10, here we demonstrate another real-world use case of RetMol for Antibacterial drug design. We choose the MurD protein (PDB ID: 3UAG) as the target. This is a promising target for antibiotic design because it is necessary for the development of the cell wall, which is essential to the bacterial survival. Inhibiting this target thus has the potential to destroy the bacterial without harming humans because the cell wall and thus the target protein is absent in animals [288]. [16]

The design criteria in this controllable generation experiment is similar to those in the SARS-CoV-2 main protease inhibitor design experiment. In addition to improving the

[16]Also see: `https://github.com/opensourceantibiotics/murligase/wiki/Overview`

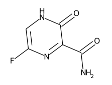
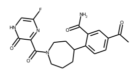
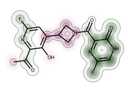
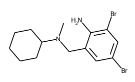
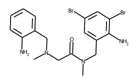
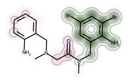
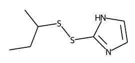
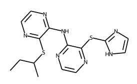
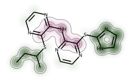
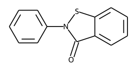
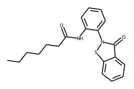
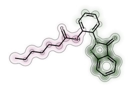
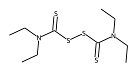
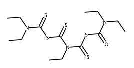
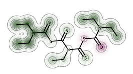
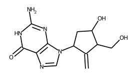
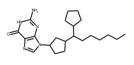
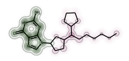
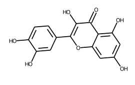
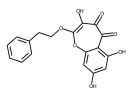
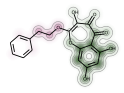
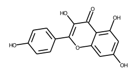
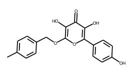
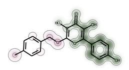
Table D.10 : Visualizations of the original and optimized inhibitors from RetMol for the SARS-CoV-2 main protease. Similarity constraint here is $\delta = 0.6$.

Inhibitor name	**Original**	**Properties (original)**	**Optimized**	**Properties (optimized)**	**Similarity map**	**Similarity**
Favipiravir		Docking -4.93 QED 0.55 SA 2.90		Docking -6.78 QED 0.77 SA 3.50		0.60
Bromhexine		Docking -9.64 QED 0.78 SA 2.94		Docking -11.48 QED 0.64 SA 2.57		0.60
PX-12		Docking -6.13 QED 0.74 SA 3.98		Docking -8.45 QED 0.64 SA 3.80		0.65
Disulfiram		Docking -8.58 QED 0.57 SA 3.12		Docking -9.09 QED 0.60 SA 3.39		0.64
Kaempferol		Docking -8.45 QED 0.55 SA 2.37		Docking -8.54 QED 0.63 SA 2.47		0.62

binding affinity of selected weakly-binding molecules, we have several desired properties [17] including a small logP value (below 3), a large QED value (above 0.6), a small SA score (below 4), and a molecule weight between 250-350 Da. Since the task encourages diverse generations, we set a small similarity threshold to 0.2. We select 100 molecules from bindingDB [199] that has experimental binding values to MurD, and choose eight molecules with the lowest binding affinity as input to be optimized. For RetMol, we use all the molecules resulted from bindingDB as the retrieval database. The remaining experiment

[17] https://github.com/opensourceantibiotics/murligase/issues/69

Table D.11 : Visualizations of the original and optimized inhibitors from RetMol for the SARS-CoV-2 main protease. Similarity constraint here is $\delta = 0.4$.

Inhibitor name	Original	Properties (original)	Optimized	Properties (optimized)	Similarity map	Similarity
Favipiravir		Docking -4.93		Docking -8.7		0.41
		QED 0.55		QED 0.62		
		SA 2.90		SA 3.25		
Bromhexine		Docking -9.64		Docking -12.65		0.40
		QED 0.78		QED 0.63		
		SA 2.38		SA 2.48		
PX-12		Docking -6.13		Docking -10.90		0.50
		QED 0.74		QED 0.62		
		SA 3.98		SA 3.72		
Ebselen		Docking -7.31		Docking -10.82		0.40
		QED 0.63		QED 0.61		
		SA 2.05		SA 2.27		
Disulfiram		Docking -8.58		Docking -10.44		0.45
		QED 0.57		QED 0.61		
		SA 3.12		SA 3.63		
Entecavir		Docking -9.00		Docking -12.34		0.41
		QED 0.53		QED 0.64		
		SA 4.09		SA 3.76		
Quercetin		Docking -9.25		Docking -9.84		0.41
		QED 0.43		QED 0.62		
		SA 2.54		SA 2.62		
Kaempferol		Docking -8.45		Docking -10.35		0.41
		QED 0.55		QED 0.67		
		SA 2.37		SA 2.51		

procedures, including docking, follows from those in the SARS-CoV-2 main protease inhibitor design experiment. Table D.12 compares the generation performance of RetMol with Graph GA. We can see that RetMol optimizes the input molecules better than Graph

Table D.12 : Antibacterial drug design with the MurD target comparing RetMol with Graph GA. RetMol optimizes input molecules better (in terms of binding affinity, unit in`kcal/mol`) than Graph GA under various property constraints.

Input	**Input score**	**RetMol optimized**	**Graph GA [132] optimized**
Oc1c2SCCc2nn1-c1cccc(Cl)c1	-7.73	-11.78	-9.76
Oc1c2SCCc2nn1-c1ccc(cc1)C(F)(F)F	-7.31	-12.82	-11.09
Oc1c2SCCc2nn1-c1ccc(Cl)cc1	-7.53	-12.46	-9.31
CC1Cc2nn(c(O)c2S1)-c1ccc(Cl)cc1	-7.86	-13.50	-9.39
Oc1c2SCCc2nn1-c1cccc(c1)C(F)(F)F	-7.74	-14.72	-8.98
Oc1c2SCCc2nn1-c1ccccc1C(F)(F)F	-7.63	-13.62	-8.81
Oc1c2SCCc2nn1-c1ccccc1	-7.70	-14.00	-10.68
Oc1c2SCCc2nn1-c1ccc(F)cc1	-7.19	-13.05	-11.67
Average improvement	-	**5.66**	2.38

GA.

D.3.6 Analyses

With and without retrieval module. We test the base generative model in our framework on the experiment in Section 3.2 with the same setup as our full framework. The base generative model without the retrieval module is unable to generate any molecules that satisfy the given constraints demonstrating the importance of the retrieval module in achieving controllable generation.

Varying retrieval database size. Figure D.12 shows how RetMol performs with varying retrieval database size with the novelty (middle) and diversity (right) metrics in addition to the success rate metric (left). We can observe that, similar to success rate, RetMol can achieve reasonable novelty and diversity with a small retrieval database, albeit with larger variance. The performance continues to improve and the variance continues to decrease with a larger retrieval database. This experiment corroborate the analyses in Chapter 10 that RetMol is efficient in the sense that a small retrieval database can already provide a strong performance.

Varying refinement iterations. Figure D.13 shows how RetMol performs with respect to the novelty (middle) and diversity (right) metrics in addition to success rate (left). These results are similar to and corroborate those presented in Chapter 10: RetMol achieves strong results and beats the best existing method with as little as 10 iterations on the diversity metric and with as little as 20 iterations on the novelty metric. Performance also continues to improve with more iterations.

Comparing to parameter-free information fusion To show the effectiveness of our proposed information fusion module that involves additional trainable parameters from the cross entropy function, we introduce two parameter-free information fusion modules detailed below for comparison.

Both two methods need to calculate a property score for each of the retrieved molecules (and we take an average of the property scores as the final score in the case of multi-property optimization), and then apply the softmax function on these scores to obtain a vector of normalized scores. The first method, termed **softmax-aggregation**, applies a weighted averaging of the retrieved embeddings as the fused embedding, with the weights being set to the normalized scores. Note that since all retrieved embeddings have different lengths, we simply concatenate zero vectors to those embeddings with a smaller length to maintain the same dimensionality before the weighted averaging. The second method, termed **softmax-sampling**, samples one retrieved molecule embedding as the fused embedding, by treating the normalized scores from softmax as probabilities of a multinomial distribution. The rest procedures are kept the same with RetMol.

We summarize the results in Table D.13, where we conduct the QED experiments in Section 10.3.1. Compared with RetMol (¿ 90% success rate), the softmax-aggregate method

Table D.13 : We compare our proposed information fusion module with two parameter-free fusion methods in the QED experiments in Section 10.3.1, where the results demonstrate the importance of our proposed information fusion module.

Method	**Success (%)**
Softmax-aggregate	1.5
Softmax-sampling	53.6
RetMol	**94.5**

has nearly zero success rate. It implies that training the information fusion component is necessary and that simply aggregating the embeddings from the retrieved exemplar molecules results in undesirable results. The softmax-sampling method that does not directly aggregate the information also achieves subpar performance compared to RetMol, which also demonstrates the importance of our proposed information fusion module. These two comparisons together show that our proposed information fusion module with trainable parameters that learns to dynamically aggregate retrieved information is critical for achieving good generation performance.

Computational cost For the memory cost, 1) the model size (in #parameters) does not increase much: The information fusion module contains 460,544 parameters. The base generative model in RetMol contains 10,010,635 parameters. These numbers indicate that the information module adds only less than 5% of the total parameters. 2) The memory cost in the fusion module scales linearly with the number of retrieved molecules K in order to store the K extra embeddings of the retrieved molecules, i.e., O($K\bar{L}D$), where $\bar{L}$ is the average molecule length, and D is the embedding size. Since K is small (i.e., $K = 10$ with $\bar{L} = 55, D = 256$), in experiments we find this additional memory cost is small.

For the time cost, we also observed infinitesimal difference in the runtime between the base generative model (i.e., without the fusion module) and RetMol (i.e., with the fusion module). To verify this, we ran a small experiment to compare their time cost in the encoding step, as RetMol uses the same decoder as the base generative model. Specifically, we gave 100 input molecules to both the base generative model and RetMol with a batch size of 1 and compute their average runtime (in seconds) on NNVIDIA Quadro RTX 8000. The average encoding time for the base generative model and for RetMol is **0.00402** seconds and **0.00343** seconds, respectively. It shows that the additional time cost of the fusion module in RetMol is indeed small.

Similarities of retrieved molecules We first plot in Figure D.16 the similarities between the input molecule and each of its K nearest neighbors (measured by cosine similarity) where $K = 10$, respectively, and take an average across all input molecules in the training set. We see that on average, the most similar molecule has a similarity of 0.66 to its corresponding input, and the second most similar molecule has a similarity of 0.57. The average similarities of the remaining 3rd to the 10th most similar molecules do not drop dramatically, implying the retrieved molecules are indeed similar ones to the input in the case of $K = 10$. Second, we also plot in Figure D.16 the distribution of similarities for the 1st, 2nd, and 10th retrieved molecules to their input molecule, respectively, across the training set. We see that the majority of the top-k similar molecules have a similarity greater than 0.2 with their input, even for the distribution of the 10th retrieved molecules. It implies the possibility of retrieving a largely dissimilar molecule is small. These results both imply that the noise present in the retrieved molecules is modest with a small fixed K.

Attribute relaxation ordering in molecule retriever In molecule retriever, we may need to gradually relax attribute constraints to construct a relaxed feasible set. Our strategy is to relax the harder-to-satisfy constraints first and then the easier-to-satisfy ones. In other words, we first focus more on the simple attributes and then on the hard attributes (e.g.,

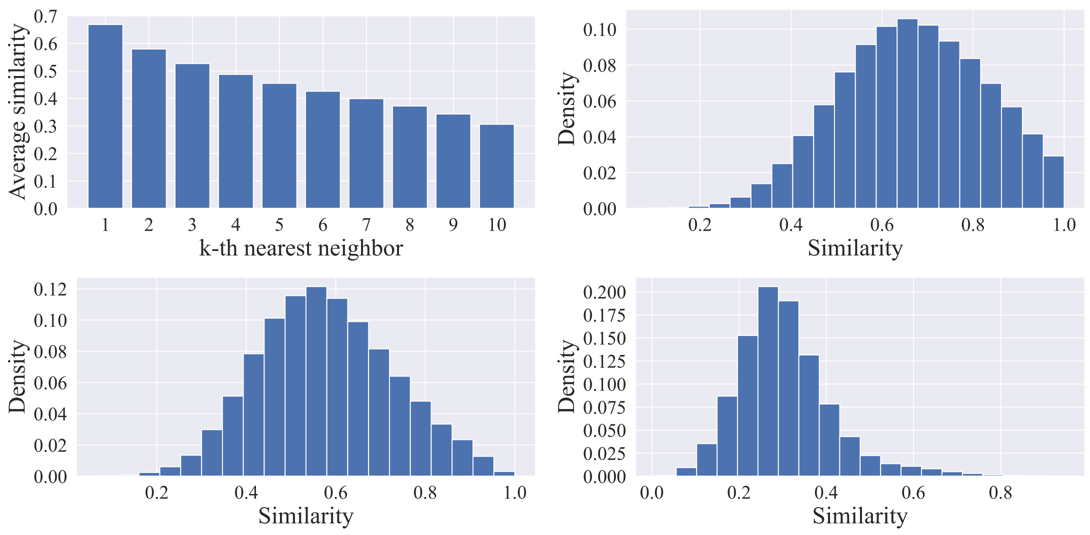


Figure D.16 : Analyses of the similarities between the retrieved molecules and the input molecules. **Top left**: the average similarity between the k-th most similar molecules to their corresponding input molecules; **Top right, bottom left, and bottom right**: the distribution of the similarities of the 1st, 2nd, and 10th most similar molecules to their corresponding input molecules, respectively.

similarity $\rightarrow$ QED $\rightarrow$ docking), which gradually increases the level of inference difficulty over iterations. The order of the constraints being removed is an important consideration during inference, especially when there are many of them. To show this, we perform a sanity check on the impact of attribute relaxation ordering in the QED task in Sec 10.3.1, where, for a quick experiment, we use a subset of 200 molecules and 20 iterations. We see that if we first relax QED and then similarity, we get 89.5% success rate, and if we first relax similarity and then QED, we get 78.5% success rate. It confirms our intuition that it is preferable to relax the harder-to-satisfy constraints first and then the easier-to-satisfy ones.

D.3.7 RetMol with other base generative models

We conduct all experiments in this work until this point with a pre-trained molecule generative model based on the transformer (i.e., BART) architecture as the encoder and decoder

Table D.14 : Penalized logP experiment with HierVAE as the base molecule generative model (encoder and decoder) in the RetMol framework. This result demonstrates that RetMol is flexible and is compatible with models other than transformer-based ones and that the RetMol framework improves controllable molecule generation performance compared to the base model alone.

Method	δ=0.6	δ=0.4
HierG2G	2.49±1.09	3.98±1.46
RetMol (w/ HierG2G)	**3.09±3.29**	**6.25±5.76**

model in the RetMol framework. As a further demonstration that RetMol is flexible and is compatible with other types of molecule generative models, and thanks to the reviewers' suggestions, we conduct the penalized logP experiment with HierG2G [139] as the RetMol encoder and decoder models. The results in Table D.14 suggest that the RetMol framework can also elevate the performance of graph-based generative models, i.e., HierG2G, on controllable molecule generation.