Information Access Using Neural Networks For Diverse Domains And Sources

by

Yuqing Xie

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Hui Jiang
Professor, Dept. of Electrical Engineering and Computer Science,
York University

Supervisor(s): Ming Li
Professor, David R. Cheriton School of Computer Science
University of Waterloo
Jimmy Lin
Professor, David R. Cheriton School of Computer Science
University of Waterloo

Internal Member: Charles Clarke
Professor, David R. Cheriton School of Computer Science
University of Waterloo
Yaoliang Yu
Professor, David R. Cheriton School of Computer Science
University of Waterloo
Wenhu Chen
Assistant Professor, David R. Cheriton School of Computer Science
University of Waterloo

Internal-External Member: James Wallace
Professor, School of Public Health Sciences
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis consists of 5 peer-reviewed publications, 2 manuscripts and experiment results not published. Yuqing Xie is the (co-) first author of 6 papers, and is responsible for the implementation, conducting of experiments, and paper writing if not specified. Following is a list of the publications, manuscripts, and Yuqing Xie's contributions in them.

1. Wei Yang*[1], **Yuqing Xie***, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, Jimmy Lin. End-to-end open-domain question answering with bertserini[281]. *NAACL 2019, Demo.* Discussed in Section 3.1.

2. Zhengkai Tu*, Wei Yang*, Zihang Fu*, **Yuqing Xie**, Luchen Tan, Kun Xiong, Ming Li, Jimmy Lin. Approximate nearest neighbor search and lightweight dense vector reranking in multi-stage retrieval architectures[240]. *ITCIR 2020.* Responsible for the factoid QA task experiment. Discussed in Section 3.2.

3. **Yuqing Xie***, Wei Yang*, Luchen Tan, Kun Xiong, Nicholas Jing Yuan, Baoxing Huai, Ming Li, Jimmy Lin. Distant supervision for multi-stage fine-tuning in retrieval-based question answering[271]. *WWW 2020.* Discussed in Section 3.3.

4. Wei Yang*, **Yuqing Xie***, Luchen Tan, Kun Xiong, Ming Li, Jimmy Lin. Data augmentation for bert fine-tuning in open-domain question answering[282]. *Preprint.* Discussed in Section 3.3.

5. While Yuqing Xie is the main contributor, Aileen Lin and Wei Yang participated in the experiments discussed in Section 4.3.

6. Cynthia Huang*, **Yuqing Xie***, Zhiying Jiang*, Jimmy Lin, Ming Li. Approximating Human-Like Few-shot Learning with GPT-based Compression[80]. *Preprint.* Discussed in Section 4.4.

7. **Yuqing Xie**, Taesik Na, Xiao Xiao, Saurav Manchanda, Young Rao, Zhihong Xu, Guanghua Shu, Esther Vasiete, Tejaswi Tenneti, Haixun Wang. An Embedding-Based Grocery Search Model at Instacart[272]. *SIGIR E-Coms 2022.* Discussed in Section 5.1.

8. Wei Zhong*, **Yuqing Xie***, Jimmy Lin. Applying Structural and Dense Semantic Matching for the ARQMath Lab 2022[307]. *CLEF 2022 Best of Lab.* Responsible for the Math Open QA part experiment design and implementation. Discussed in Section 5.2.

[1]* Equal Contribution

Abstract

The ever-increasing volume of web-based documents poses a challenge in efficiently accessing specialized knowledge from domain-specific sources, requiring a profound understanding of the domain and substantial comprehension effort. Although natural language technologies, such as information retrieval and machine reading compression systems, offer rapid and accurate information retrieval, their performance in specific domains is hindered by training on general domain datasets. Creating domain-specific training datasets, while effective, is time-consuming, expensive, and heavily reliant on domain experts. This thesis presents a comprehensive exploration of efficient technologies to address the challenge of information access in specific domains, focusing on retrieval-based systems encompassing question answering and ranking.

We begin with a comprehensive introduction to the information access system. We demonstrated the structure of a information access system through a typical open-domain question-answering task. We outline its two major components: retrieval and reader models, and the design choice for each part. We focus on mainly three points: 1) the design choice of the connection of the two components. 2) the trade-off associated with the retrieval model and the best frontier in practice. 3) a data augmentation method to adapt the reader model, trained initially on closed-domain datasets, to effectively answer questions in the retrieval-based setting.

Subsequently, we discuss various methods enabling system adaptation to specific domains. Transfer learning techniques are presented, including generation as data augmentation, further pre-training, and progressive domain-clustered training. We also present a novel zero-shot re-ranking method inspired by the compression-based distance. We summarize the conclusions and findings gathered from the experiments.

Moreover, the exploration extends to retrieval-based systems beyond textual corpora. We explored the search system for an e-commerce database, wherein natural language queries are combined with user preference data to facilitate the retrieval of relevant products. To address the challenges, including noisy labels and cold start problems, for the retrieval-based e-commerce ranking system, we enhanced model training through cascaded training and adversarial sample weighting. Another scenario we investigated is the search system in the math domain, characterized by the unique role of formulas and distinct features compared to textual searches. We tackle the math related search problem by combining neural ranking models with structual optimized algorithms.

Finally, we summarize the research findings and future research directions.

Acknowledgements

When I first joined the University of Waterloo, I had limited knowledge of computer science (I majored in Math before) and could barely communicate with people in English. Looking back, it is incredible that I have been doing the most cutting-edge artificial intelligence research and had some progress. Along the way, I had the privilege of meeting and being supported by many remarkable individuals.

To begin with, I want to express my gratitude to my supervisors, Prof. Ming Li and Prof. Jimmy Lin, for their invaluable guidance and support throughout my research journey. I am deeply grateful to Prof. Ming Li for allowing me to conduct research in such an exceptional environment. His insightful ideas have helped me gain a broader perspective beyond my specific work, allowing me to develop a higher level of understanding. I would also like to extend my sincere appreciation to Prof. Jimmy Lin. From the beginning, even before becoming my co-supervisor, Prof. Lin has provided me with hands-on instructions that have been instrumental in shaping my research. His guidance was crucial in initiating my journey as an NLP researcher. I thank Prof. Ming Li and Prof. Jimmy Lin for their instruction, care, and unwavering support. They have not only shaped my research skills but also influenced my academic mindset and my personal development. Their guidance has been invaluable, and I am honoured to have had the opportunity to work under their supervision.

I would like to express my sincere gratitude to my thesis committee: Charles Clarke, Yaoliang Yu, Wenhu Chen, James Wallace and Hui Jiang. Their expertise and thoughtful feedback have played an important role in refining my work. I am grateful for their time, commitment, and invaluable contributions.

During my Ph.D., I have done great internships at RSVP.ai, Amazon AWS and Instacart. I thank my mentors, Luchen Tan and Wei Yang, Yi Zhang and Yi-an Lai, Taesik Na and Tejaswi Tenneti, when I worked there. My internship project at RSVP.ai eventually leads to the BERTserini project, and the project with Instacart led to the database search paper, which became part of this dissertation. I would also like to express my gratitude to the wonderful colleagues during my internship: Wei Yang, Aileen Lin, Ruixue Zhang, Zhengkai Tu, Zihang Fu, Peng Shi, Wuchen Tang, Kun Xiong, Elman Mansimov, Saurav Manchanda.

I am grateful to Prof. Shanfeng Zhu for introducing me to the fascinating field of NLP. I would also like to thank Houcong Lin, who has cultivated my interest in programming since high school. Their encouragement and support have fueled my curiosity and drive in this domain. I am thankful for their valuable foundation in shaping my programming skills.

I would like to express my heartfelt appreciation to the friends who have accompanied me throughout this journey, including Ruixue Zhang, Aileen Lin, Ruoxi Zhang, He Bai, Wei Zhong, Haohan Zhang, Kexin Yang, Xiaojun Su, Qianfeng Shen, Zhao Zhang, Daphne Chen, Cynthia Huang, Mingyang Jiang, Qin Yu and many others. I am thankful for their friendship and countless moments of intellectual exchange and shared experiences.

I want to thank my parents for their unconditional love, support, and understanding throughout this journey. Their belief in me and constant encouragement have been my most significant source of strength. I am truly blessed to have them by my side.

Lastly, I would like to extend my deepest gratitude to Victor for his love and support throughout this journey. Our paths crossed during our internship at RSVP.ai, and little did we know that our collaboration on a project would eventually become the starting point for this thesis, namely BERTserini. Over the past several years, Victor has been more than just a partner, a lab mate, and a best friend. He has become someone I admire for his exceptional qualities of honesty, active attitude, intelligence, and hard work. I am profoundly grateful to Victor for the countless hours of collaboration, brainstorming sessions, and late-night discussions that have shaped my research and enriched my understanding. His encouragement, empathy, and belief in my abilities have been invaluable sources of strength and motivation. I thank him for everything he has done for me.

Dedication

To my parents and Victor, for their unconditional love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Information access systems have been a long-lasting and fundamental tool for knowledge-seeking since the beginning of civilization. From ancient libraries to modern digital platforms, the quest for information and answers has been a constant pursuit of humankind. They play a pivotal role in people's lives by serving as gateways to the vast realm of knowledge and information. In a world filled with data and information, information access systems like search engines provide a sense of order and efficiency, allowing people to navigate the vast digital landscape effortlessly. Whether students conducting research, professionals seeking insights, or individuals looking for practical solutions, search engines enable users to find relevant and reliable information in seconds. By organizing and indexing the vast corpus of human knowledge, information access systems empower individuals to make informed decisions, expand their understanding of the world, and engage in lifelong learning. The retrieval-based information access system is one of the most widely used information access systems.

Navigating through vast information sources, such as the internet or an extensive academic library, poses significant challenges for search. The sheer magnitude and complexity of these repositories make it difficult for people to find specific and relevant information efficiently. With the ever-growing volume of digital information, users often people information overload. For instance, the English Wikipedia [1] dump 2018 contains up to 5.08 million documents with 29.5 million paragraphs. Additionally, platforms like PubMed [2] comprise more than 33 million

[1] https://en.wikipedia.org/

[2] https://pubmed.ncbi.nlm.nih.gov/

articles from biomedical literature MEDLINE, life science journals, and online books, with the article number increasing by hundreds of thousands every year. Seeking specialized knowledge from these documents requires a specialized understanding of the domain and considerable comprehension effort. For example, doctors often search through thousands of medical papers to find specific solutions or cures. The search process can take hours or days, during which the patient's life may be at stake.

Recent advancements in large pre-trained language models have played a pivotal role in boosting the performance of retrieval-based search systems, as demonstrated by examples like Bing [3] and ChatGPT. These pre-trained models have been trained on vast amounts of diverse text data, enabling them to learn rich language representations and acquire a broad understanding of various domains. By leveraging these pre-trained models, search systems can tap into their extensive knowledge and linguistic capabilities to improve the accuracy and relevance of the effectiveness of information access systems. These models capture semantic relationships, understand context, and generate coherent responses. The power of these pre-trained models lies in their ability to leverage the collective knowledge of vast corpora. By applying deep learning models as automatic information-seeking tools, we can leverage their capabilities to locate the most helpful information from this vast corpus in minutes or seconds. This can revolutionize how we access critical information, especially during urgent situations like disease outbreaks affecting millions of people in a short period, such as COVID-19.

However, using pre-trained models for information access in specific domains can pose several challenges. Firstly, pre-trained models are typically trained on large-scale general domain data, which may not adequately capture the nuances and intricacies of specific domains. This lack of domain-specific knowledge can lead to sub-optimal performance when applied to domain-specific tasks. Secondly, pre-trained models may struggle with domain-specific vocabulary, jargon, or terminology not well-represented in their training data. This can result in difficulties in accurately understanding and processing domain-specific texts. Additionally, pre-trained models might encounter issues when dealing with domain-specific structures, formats, or data types that differ from the ones they were trained on. This can hinder their ability to effectively extract relevant information or generate accurate responses in specific domains. Finally, the limited availability of labelled training data for fine-tuning pre-trained models in specific domains can be a significant obstacle. It becomes challenging to adapt the models to perform optimally in the target domain without sufficient domain-specific training data. These difficulties highlight the need for specialized techniques, such as domain adaptation, data augmentation, or fine-tuning strategies, to enhance the performance of pre-trained models in specific domains.

In this thesis, we aim to address the challenges mentioned above and provide potential solutions

[3] https://www.bing.com/

to enhance the performance of pre-trained models in specific domains for information access tasks. We recognize the importance of improving the accuracy and effectiveness of retrieval-based systems in domain-specific contexts, where the challenges of domain knowledge, vocabulary, structures, and limited training data are prevalent. To tackle these challenges, we propose several approaches and techniques that tap into domain adaptation, data augmentation, and fine-tuning strategies. By exploring these avenues, we aim to demonstrate how these methods can mitigate the difficulties of using pre-trained models in specific domains. Through empirical evaluations and experiments conducted in various domains, we seek to showcase the efficacy and applicability of our proposed solutions. By addressing these challenges head-on, we strive to advance retrieval-based information access systems in specific domains, ultimately providing users with more accurate and reliable answers in their domain-specific information-seeking endeavours.

1.2 Thesis Overview

This thesis delves into retrieval-based information access systems and explores various approaches to enhance their performance in specific domains and extend to various formats.

In Chapter 2, we start with a comprehensive introduction to the background of machine learning, neural networks, pre-trained language models, and the fundamentals of retrieval and information access systems. These foundational concepts lay the groundwork for understanding the subsequent chapters. We also discuss some widely used retrieval-based information access systems and their characteristics.

In Chapter 3, we take a typical information access task: open-domain question answering as a starting point. We first investigate the challenges and trade-offs encountered in this task. We address several vital aspects that are crucial for achieving optimal performance in such a system: Firstly, we discuss the integration of a retriever and a neural reader to extract accurate answers. This involves finding the right balance between retrieval accuracy and the efficiency of the reader model, tackle with the distribution difference between the retrieved results and reader inputs. We proposed a simple yet effective integration which enhances the system's overall performance. Next, we examine the trade-off between speed and performance in the retriever component. Retrieving relevant documents from a large corpus can be time-consuming, and balancing the retrieval time and quality of the retrieved documents becomes essential. Furthermore, we address the challenge of training a better neural reader model under the retrieval-based setting. We investigate different approaches to leverage the retrieved documents and utilize them effectively in the reader training process. This includes techniques for data augmentation, fine-tuning strategies, and the selection of training examples. Combining our findings with the broader context of information access systems, we present a comprehensive approach to building a robust and efficient retrieval-based question-answering system.

In Chapter 4, we explore methods of adapting the information access systems to different domains. We recognize that each domain possesses unique characteristics, including variations in text content, vocabulary distribution, and domain-specific knowledge. These distribution differences will lead to difficulties in directly applying general-domain-trained neural models in these domains. We build baseline models which applied previous build general domain models to these new datasets. To address these challenges, we explore various techniques and methodologies to adapt the system to diverse domains effectively: We begin with synthesizing training data in specific domains. Furthermore, we experiment with further pre-training, benefiting from training the models on domain-specific data and the existing pre-training on general domain data. This approach helps the system to acquire domain-specific knowledge and improve its performance. We also investigate the progressive domain-clustered training method, which involves selecting and incorporating data from general domains during training. By carefully scheduling the training data, we aim to balance general domain knowledge and domain-specific expertise, enabling the system to handle queries from various domains effectively. Additionally, we introduce a domain-specific zero-shot ranking method that leverages neural-based compressors. By exploring these techniques and methodologies, we provide insights and practical guidelines for adapting the information access system to different domains. Through these efforts, we strive to enhance the system's performance, enable it to handle diverse domain-specific queries effectively, and ultimately broaden its applicability in real-world information access scenarios.

In Chapter 5, we explore the adaptation of the information access systems to different information sources: First, we focus on the e-commerce domain, which often involves structured data in databases. We present methods for effectively leveraging table data, tackling noisy user logs, and enabling cold start. We demonstrate the effectiveness of our system in ranking relevant products in a grocery e-commerce setting. Next, we tackle the math domain, which presents a distinct challenge with including mathematical formulae. We explore techniques to incorporate formulae understanding and processing into the system, enabling it to retrieve and generate accurate answers to math-related queries. Additionally, we address the challenges introduced by version updates when using neural models in real-world deployment scenarios. We explore the potential issues and discuss strategies to mitigate the impact of version updates, ensuring the robustness and reliability of the system during deployment. Our findings and insights will contribute to developing more versatile and reliable information access systems in real-world applications.

In Chapter 6, we provide conclusions of the thesis and discuss future directions. We summarize the main contributions and findings of the entire thesis and reflect on the accomplishments achieved. Additionally, we explore potential avenues for future research and expansion.

1.3 Contributions

This thesis makes several key contributions to the field of retrieval-based information access systems:

General Domain Retrieval-Based Question Answering System

This section paves the way for future research by introducing BERTserini, the pioneering open-domain question answering system based on a large pre-trained language model, and exploring avenues for enhancing system performance and trade-offs, thus advancing the field of question answering systems.

- In Section 3.1:
  - Introduces an open-domain question answering system that combines BERT with the Anserini information retrieval toolkit.
  - The proposed system demonstrates significant improvements in accuracy compared to previous methods on standard benchmark test collections and achieved state-of-the-art performance when publishing.
- In Section 3.2:
  - Discusses the trade-offs between retrieval effectiveness and efficiency by analyzing approximate nearest neighbour search and light-weighted re-ranker in a multi-stage ranking setting.
- In Section 3.3
  - Explores strategies for gathering distantly supervised training samples to train a better reader under the retrieval-based question-answering setting.
  - Experiment with sampling strategies of negative training examples and effectively integrate existing training data with distant supervision, thereby enhancing the effectiveness of the reader model.

Zero-shot Adaptation to Domains

This section explores adapting open-domain systems for specific vertical domains like biomedical and finance through rapid zero-shot techniques, including auto data generation and domain-specific pre-training, notably featuring the GPT-based compressor's non-parametric adaptation strategy, which calculates information distance between text documents and shows promising experimental results, contributing to future domain-specific adaptation research for open-domain systems.

- In Section 4.3:
  - Explores question generation as a data augmentation that can synthesize domain-specific training data for the reader.
  - Explores further pre-training, which adapts the large neural networks to target domains with the help of unlabeled in-domain corpus.
  - Explores the progressive domain-clustered training method, which utilizes the large general domain labelled dataset by sorting the data according to the semantic similarity to the target domain, to boost model transfer ability.
- In Section 4.4:
  - Proposed a novel non-parametric zero-shot human-like learning method derived from a GPT-based compressor.
  - Experimented with domain zero-shot ranking datasets shows the effectiveness of the proposed method.

Generalization to Diverse Sources

This section addresses the challenge of expanding retrieval systems to encompass a variety of text formats, including tabular data from relational databases and mathematical formulas in documents with mathematical content. By tackling the task of extending retrieval systems to handle diverse text formats like tabular data and mathematical formulas, we provides a roadmap for future investigations in the field.

- In Section 5.1
  - Demonstrates an embedding-based model for grocery search at Instacart, utilizing a two-tower transformer-based encoder architecture.
  - Proposes self-adversarial learning and cascade training methods to train the model on noisy data effectively.
- In Section 5.2
  - Demonstrates a retrieval-based question answering system, which requires the system to conduct a search based on a corpus that contains math formulae as an important component.

In Chapter 6, we encapsulate our discoveries and highlight potential pathways for future exploration.

Chapter 2

Background

In this chapter, we introduce basic knowledge of machine learning, deep learning, natural language processing (NLP), and some widely used retrieval-based information access systems. We cover key concepts and techniques required for understanding the training and architecture of these systems.

2.1 Deep Learning for NLP

Supervised Learning: Supervised learning refers to a type of machine learning where a model learns a function that maps input data to corresponding output labels. In supervised learning, we have a dataset consisting of input-output pairs, denoted as $\{(x_i, y_i)\}_{i=1}^{N}$, where x_i represents the input data and y_i represents the corresponding output labels, and N is the number of training examples. The goal is to learn a function f that can accurately predict the output label y given a new input x. The objective is to find a function f that minimizes the discrepancy between the predicted output $\hat{y} = f(x)$ and the true output y, usually quantified by a loss function L. This can be formulated as an optimization task to find the optimal parameters θ of the function f:

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \theta), y_i). \tag{2.1}$$

Unsupervised Learning: While on the other hand, unsupervised learning is another branch of machine learning that tries to understand and model data without explicit labels or guidance. It encompasses various techniques and algorithms designed to discover hidden patterns, cluster similar data points or reduces dimensionality. By leveraging the inherent structure and statistics of the data itself, unsupervised learning enables us to gain insights and make discoveries in an exploratory manner. We are not going to cover methods using unsupervised learning in this thesis.

Training, Validation and Test Datasets: It is common practice for supervised learning to divide the available data into three sets: the training, validation and test set.

- **The training set** is usually the largest subset of the data. The models learn to adjust parameters to make predictions based on this dataset. It is important to have a sufficiently large and diverse training set to enable the model to learn patterns and relationships in the data.
- **The test set** is a smaller independent subset of the data we use to evaluate the trained model. It measures the model's performance on unseen data and provides an estimate of how the model would perform in the real world.
- **The validation set** is used for hyper-parameter tuning and comparing different versions of trained models on unseen data.

Linear Regression: Linear regression is one of the simplest supervised learning models, and formally:

$$f(x;\theta) = \theta^\top x + b,$$

Mean square error(MSE) is a commonly used loss function to quantify the difference between the models, defined as follows:

$$L_{MSE}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i;\theta))^2$$

Multi-layer Perceptron: The Multi-layer Perceptron (MLP) consists of multiple layers of interconnected nodes, called neurons, organized in a sequential manner. For an MLP with L layers, including the input layer, hidden layers, and output layer. The input layer consists of n neurons, and the output layer consists of m neurons. Each hidden layer has k_i neurons, where $1 \leq i \leq L-2$. Formally, the MLP can be described as follows:

1. Input layer: $a_0 = x$, where x is the input vector.
2. Hidden layers: $a_i = f\left(\sum_{j=0}^{k_{i-1}} w_{ji}^{(k)} a_{i-1} + b_i\right)$, where $w_{ji}^{(k)}$ represents the weight.
3. Output layer:$a_{L-1} = f\left(\sum_{j=0}^{k_{L-2}} w_{j(L-1)}^{(m)} a_{L-2} + b_{L-1}\right)$

Gradient Descent: Gradient descent is an optimization algorithm commonly used to minimize the loss function. It iteratively adjusts the model's parameters by following the derivative of the loss function. The update rule for gradient descent can be represented as:

$$\theta \leftarrow \theta - \alpha \cdot \frac{\partial L(\theta)}{\partial \theta},$$

where α is the learning rate. It aims to find the optimal parameter that minimizes the loss function.

Stochastic Gradient Descent (SGD) is a variation of gradient descent that is particularly suitable for large-scale datasets. Instead of calculating the gradient using the entire training set, SGD randomly selects a subset of training samples, often called mini-batch, to compute the gradient. In SGD, the parameters are updated after each mini-batch, and the process is repeated for multiple epochs until convergence.

Over the years, several variants of SGD have been developed to improve its performance and address challenges associated with the standard SGD algorithm. One variant is Momentum [195], incorporating a momentum term to accelerate convergence. Adaptive learning rates methods, such as Adagrad [45], Adadelta [297], RMSprop [1], Adam [97], AdamW [147], and AdaFactor [221], are other popular variants of SGD. We mainly use mini-batch SGD and Adam for model training in the experiments of this thesis.

Tokenization: Tokenization refers to the process of breaking down a text into individual tokens, such as words or subwords, which serve as the basic units of analysis. Tokenization can be performed at various levels, depending on the specific requirements of the task. For example, the text is segmented into individual words in word-level tokenization. On the other hand, subword tokenization, such as Byte-Pair Encoding (BPE) [216] or WordPiece [225], splits the text into subword units that capture both morphological information and rare or out-of-vocabulary words. Subword tokenization is particularly useful for languages with complex morphology or large vocabularies. This thesis mainly adopts subword-level tokenization for English datasets; some models use character-level tokenization for Chinese datasets.

Word Vector and Embedding: After tokenization, the next step is to represent each token as a numerical vector, known as word embedding. By representing words as vectors, models can perform mathematical operations on these vectors to capture similarities, analogies, and contextual relationships between words. Popular word embedding models, such as Word2Vec [166], and GloVe [188], use unsupervised learning algorithms to learn these representations based on large corpora of text data. Word embedding is often pre-trained on large corpora and can be readily used in various NLP tasks.

[1] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

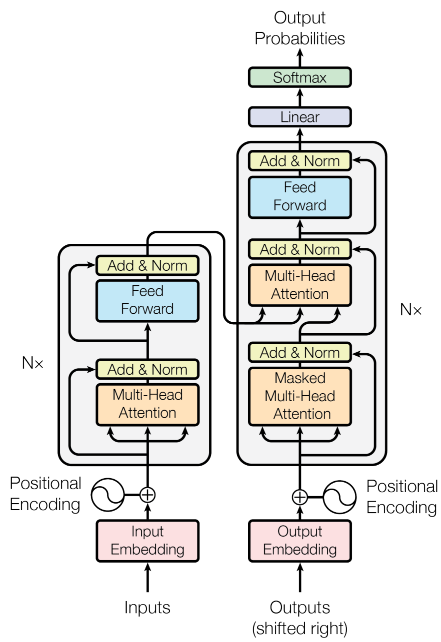


Figure 2.1: Architecture of Transformers.

RNN and LSTM: Recurrent Neural Networks (RNNs) [47] [2] are a class of neural networks that excel at modeling sequential data. They are designed to process inputs of varying lengths by maintaining an internal hidden state that can capture the contextual information from previous inputs.

One popular variant of RNNs is the Long Short-Term Memory (LSTM) [76] network. LSTMs address the vanishing gradient problem often encountered in training RNNs by introducing a gating mechanism that allows the model to selectively retain or discard information at each time step. LSTMs have been widely used in various NLP tasks such as language modeling, machine translation, and sentiment analysis.

Attention and Transformers: Unlike RNN and LSTM, transformers rely on a self-attention mechanism that allows the model to attend to different parts of the input sequence during processing. This attention mechanism enables transformers to capture long-range dependencies and model interactions between all input elements simultaneously. The transformers [245] structure has been prove to be successful in catch long term dependency in sequence data, which is of vital importance in natural language processing. Figure 2.1 demonstrates a typical transformer

[2]The source of these structures dates back to 1982-1986. See https://ai.stackexchange.com/questions/8190/where-can-i-find-the-original-paper-that-introduced-rnns

architecture. Transformers have achieved state-of-the-art performance in various NLP tasks, including machine translation, question answering, and text generation. Most of the models we used in this thesis are based on Transformers.

Given the input word sequence, the model first map them into vectors $\{x_1, x_2, ..., x_n\}$ through a word embedding layer. Following are the transformer encoder layers, each of which is consists of a multi-head attention block followed by a feed forward block. In a single head attention, the input sequence $\{x_1, x_2, ..., x_n\}$ are transformed into corresponding *query*, *key* and *value* vectors through linear projections:

$$q_i = W_q * x_i, k_i = W_k * x_i, v_i = W_v * x_i. \tag{2.2}$$

And then, we compute the self-attention weight matrix for the input sequence, in which each element represents the similarity of the two tokens

$$a_{i,j} = q_i^\top * k_j, \tag{2.3}$$

and normalize with softmax function. We then apply the attention matrix on the value vectors to get the re-weighted word representation:

$$y_i = \sum_{j=1}^{n} a_{i,j} * v_j. \tag{2.4}$$

To form a multi-head attention block, we use k different linear transformation weights and concatenate the final results y_i^k as the new output. We pass y_i into a feed-forward layer to map it back to the same dimension as the input vector. We add a skip-connection with layer normalization after all the layers to help with the convergence of deep neural networks. Transformer decoder layers have the similar structure. We replace the key and value vectors in the attention with the transformed output of the last encoder layer.

2.2 Pre-trained Language Models

Pre-trained Language Models(PLM): To familiarize models with the data distribution, [32] proposed to train the model on un-labeled data before training it on the labeled datasets. This is also know as a pre-train and then fine-tune learning routine. Based on pre-training, several works, including GPT [198, 16], BERT [42], XLNET [285], ELECTRA [29], BART [113], T5 [199] GPT-2, 3, 4 [198, 16, 181], and InstructGPT [182] etc, successfully improve the performance on many NLP downstream tasks. To reduce the high computation cost for industry applications, DistilBERT[211], and MiniLM[260] propose to distill from large pre-trained models into smaller models.

Pre-training has been adopted in numerous deep learning models with the rise of transformer [245] due to its ability of learning task-agnostic representation. In NLP, encoder-only transformers like BERT [42] has achieved impressive performance on GLUE benchmark [253] including tasks like natural language inference and sentiment analysis with only MLP and fine-tuning. Decoder-only transformers like GPT [198, 16] and related models [235, 236] can treat downstream discriminative tasks in a generative way. However, previous works on few-shot learning using language models are either prompt-based [190, 189, 126] or fine-tuning-based [302, 275, 178] while in this work, we propose a new way to leverage pre-trained language models for few-shot learning without fine-tuning or in-context learning.

After the success of BERT and GPT, most models adopt the transformer model. Either take the encoder part like BERT, the decoder part like GPT, or the whole structure like T5 and BART. Yang et al. [277] shows the development of such models over the past several years in Figure 2.2. We mainly adopt BERT, GPT and T5 in this thesis since they are the most widely used, open source, parameter available and well-integrated models.

Now we introduce how to pre-train the stacked transformer models. Taking the most widely used PTM model, BERT, as an example: Figure 2.3 shows the general stacked transformer architecture and how it can adapt to different downstream tasks. At the very beginning, all parameters in the model are randomly initialized. Pre-training aims to update model parameters on unsupervised corpora. BERT pre-train the model with context prediction tasks: mask language model (predict the missing words) and next sentence prediction (whether two sentence will appear in sequence order). After the stacked transformer layers, they add a feed-forward layer to map the representation vector sequence into desired output formats. There are also other training signals can be used to pre-train the model, most of them can be easily gathered from the natural text, and are aimed to let the model learn the language distribution.

In the context of neural models, transfer learning involves pre-training a model on a large corpus of text and then fine-tuning it on a target task or domain-specific dataset. The idea behind transfer learning is that the pre-training phase enables the model to learn general language representations, capturing syntactic, semantic, and contextual information. The transfer learning process typically involves two stages: pre-training and fine-tuning. During pre-training, the model is trained on a large corpus, such as a collection of books, articles, or web pages. The objective is to predict masked or corrupted words within a sentence, forcing the model to learn meaningful representations. In the fine-tuning stage, the pre-trained model is further trained on a task-specific dataset with labeled examples. Fine-tuning allows the model to adapt its representations to the specific task and domain, leveraging the knowledge gained during pre-training.

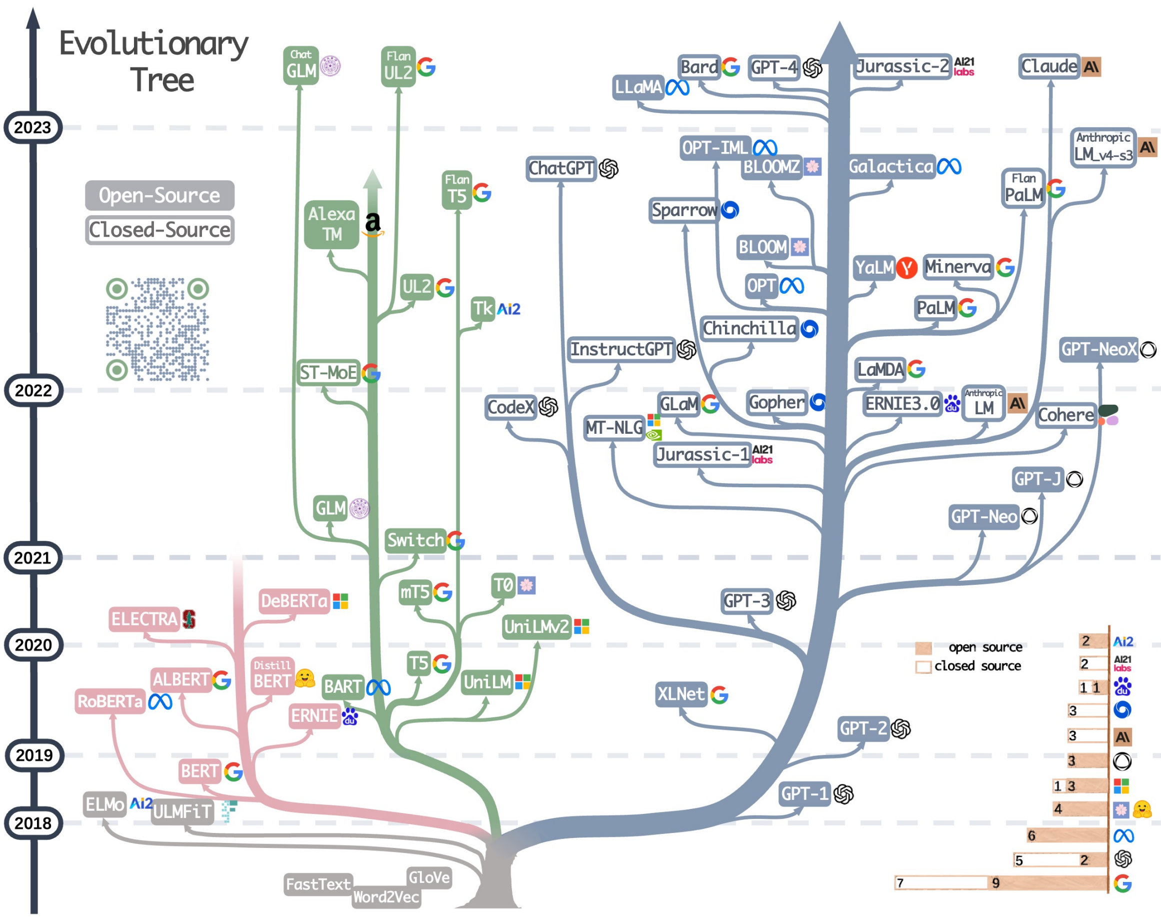


Figure 2.2: Evolution of Transformer-based large language models [277].

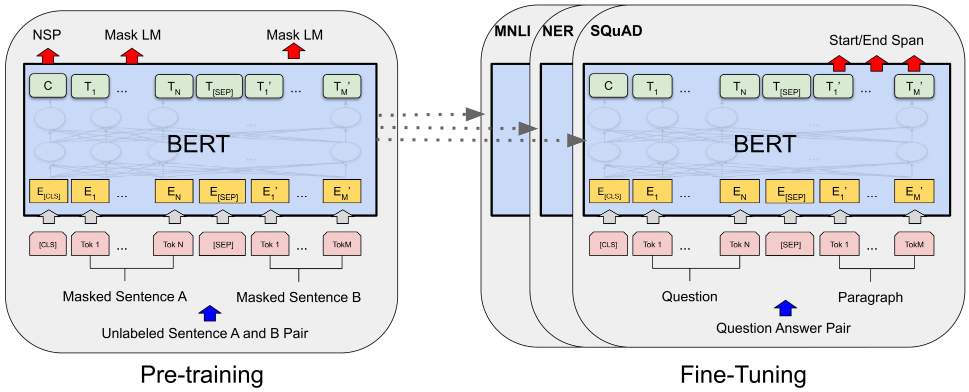


Figure 2.3: Architecture of BERT.

Codebase

HuggingFace[3] provides several easy to use and well maintained repositories . For example **Transformers**[4] provides thousands of pre-trained models to perform tasks on different modalities such as text, vision, and audio. Text models can be applied for tasks like text classification, information extraction, question answering, summarization, translation, text generation, in over 100 languages. Our main experiments are based on their code base. We mainly conduct experiments on PyTorch backend with Tesla V100 GPUs.

2.3 Information Access Systems

In this section, we briefly introduce the information access system, specifically in the context of open-domain question answering. We build a retrieval-based question-answering system, which involves two major components: the retriever and the reader. The retriever is responsible for retrieving relevant documents or passages from a large corpus given a question. After gathering top relevant documents from the retriever, the reader processes and understands the retrieved documents to identify or generate the answers. We provide an overview of the problem formulation, datasets, and evaluation methodologies commonly used in retrieval-based question-answering systems. Additionally, we discuss the techniques and models engaged in the retrieval and the reader.

[3] https://huggingface.co/
[4] https://github.com/huggingface/transformers

2.3.1 The Task: Open-Domain Question-Answering

While the origins of question answering date back to the 1960s, the modern formulation can be traced to the Text Retrieval Conferences (TRECs) in the late 1990s [249]. With roots in information retrieval, it was generally envisioned that a QA system would comprise pipeline stages that select increasingly finer-grained segments of text [232]: document retrieval to identify relevant documents from a large corpus, followed by passage ranking to identify text segments that contain answers, and finally answer extraction to identify the answer spans.

As NLP researchers became increasingly interested in QA, they emphasized the later stages of the pipeline to emphasize various aspects of linguistic analysis. Information retrieval techniques receded into the background and became altogether ignored. Most popular QA benchmark datasets today—for example, TrecQA [286], WikiQA [283], and MSMARCO [173]—are best characterized as answer selection tasks. The system is given the question and a candidate list of sentences to choose from. Of course, those candidates have to come from *somewhere*, but their source lies outside the problem formulation. Similarly, reading comprehension datasets such as SQuAD [200] eschew retrieval since there is only one document from which to extract answers.

In contrast, what we refer to as open-domain question-answering (ODQA) begins with a large corpus of documents. Since it is impractical to apply inference exhaustively to all documents in a corpus with current models (primarily based on neural networks), this formulation necessarily requires some retrieval technique to restrict the input text under consideration—and hence an architecture quite like the pipelined systems from over a decade ago. There has been a resurgence of interest in this task, the most notable of which is Dr.QA [22]. Other papers have examined the role of retrieval in this end-to-end formulation [255, 101, 108], some of which have, in essence, rediscovered ideas from the late 1990s and early 2000s.

The SQuAD [200] dataset was initially designed for machine reading comprehension, meaning the answers should be within the given golden paragraph. Dr.QA [22] proposed to ignore the paragraphs and directly evaluate ODQA on SQuAD, mainly due to the high quality. However, because the design goal is machine reading comprehension, some questions cannot be answered under the "open-domain" setting. For example, there might be a co-reference in the question, making the question only applies to the specific paragraph. Furthermore, the dataset was constructed based on hundreds of Wikipedia documents, leading to limited coverage of the topics. TriviaQA [90] is also widely used for ODQA but was initially designed for machine reading comprehension.

Later, Kwiatkowski et al. [103] construct the NaturalQuestions (NQ) dataset. The NQ corpus contains questions from real users, and it requires QA systems to read and comprehend an entire Wikipedia article that may or may not contain the answer to the question. NQ is a more realistic and challenging task than prior QA datasets because of the inclusion of real user questions and the requirement that solutions should read an entire page to find the answer.

2.3.2 Problem Formulation and Evaluation

Problem Formulation: The ODQA problem can be formulated as follows: given a question q and a corpus of documents or passages $\mathcal{D}$, the task is to find a text answer a^*, which answers the problem in query q:

$$a^* = \text{System}(q|\mathcal{D}), \tag{2.5}$$

Evaluation Metrics

To evaluate the ODQA systems, we usually have labelled answer strings; in some cases, there will be more than one reference answer for a single query. Matching any one of the reference answers will be an acceptable return. The most common evaluation metrics include:

1. **Exact Match:** The most strict metric is to check if the predicted answer string is exactly the same as one of the reference answers after string cleaning:

$$\text{EM}(a_{\text{gold}}, a_{\text{predict}}) = \begin{cases} 1, \text{ if } a_{\text{gold}} = a_{\text{predict}}, \\ 0, \text{ otherwise }, \end{cases} \tag{2.6}$$

   where $a_{\text{gold}} = (w_1^g, w_2^g, ..., w_{n_g}^g)$ is the golden answer and $a_{\text{predict}} = (w_1^p, w_2^p, ..., w_{n_p}^p)$ is the model prediction, with w_i^* represents the words in the answers. $a_{\text{gold}} = a_{\text{predict}}$ means $n_g = n_p$ and $w_i^g = w_i^p, \forall i, < i \leq n_g$.

2. **F1-score:** Another commonly used metric is F1-score measure the word overlap ratio in the answers:

$$\text{F1}(a_{\text{gold}}, a_{\text{predict}}) = \frac{2 * P * R}{P + R} \tag{2.7}$$

$$\text{P}(a_{\text{gold}}, a_{\text{predict}}) = \frac{|\{w_i | w_i \in a_{\text{gold}}, \text{ and } w_i \in a_{\text{predict}}\}|}{|a_{\text{predict}}|}, \tag{2.8}$$

$$\text{R}(a_{\text{gold}}, a_{\text{predict}}) = \frac{|\{w_i | w_i \in a_{\text{gold}}, \text{ and } w_i \in a_{\text{predict}}\}|}{|a_{\text{gold}}|} \tag{2.9}$$

   where $|\cdot|$ represents the number of words in the answers.

3. **BLUE, Rouge and their variants:** We can also consider QA as a text-generation problem. Typical generation metrics such as BLEU [183] and ROUGE [127] can also be applied as evaluation metrics.

Dr.QA [22] was the most notable system of applying machine learning methods to the ODQA task. It breaks the task down into two parts: retrieval and reader. The retriever is responsible for gathering the most relevant materials: documents or paragraphs from the corpus, focusing more on relevance and recall. The reader takes the selected materials to give a precise answer, focusing more on precision. Other methods [199, 16] enable closed-book question answering, which directly generates answers without referring to the corpus. These methods mainly rely on the pre-trained phase for memorizing the knowledge. We mainly follow the first line of research: the retrieval-based question-answering (RBQA) systems.

2.3.3 Information Retrieval

In the RBQA systems, we first retrieve the most relevant ones to acquire knowledge from a large corpus, which usually contains millions of documents. Typical retrieval systems compute a relevance score for each document given the question, and then the top-ranked documents are selected as the final results.

Task Formulation

The core function in the retrieval systems is a model to measure the similarity between the question q and the document or passage d: $\text{Sim}(q, d|\theta_{IR})$. Formally, the goal of a retriever is to generate an ordered list of texts retrieved from a corpus $\mathcal{D}$ in response to a query q:

$$\text{IR}(\mathcal{D}, q, k) = [d_1, d_2, ..., d_k], d_i \in \mathcal{D}, \tag{2.10}$$

$$\text{Sim}(q, d_i|\theta_{IR}) > \text{Sim}(q, d_{i+1}|\theta_{IR}), \tag{2.11}$$

$$\text{Sim}(q, d|\theta_{IR}) > \text{Sim}(q, d'|\theta_{IR}),\ \text{for } d \in \text{IR}(\mathcal{D}, q) \text{ and } d' \notin \text{IR}(\mathcal{D}, q). \tag{2.12}$$

where k is the number of documents to return. Note that in actual implementation, we allow some missing retrievals caused by efficiency reasons, meaning $\text{IR}(\mathcal{D}, q)$ might not be optimal, and Equation (2.12) might not be satisfied.

Evaluation

To evaluate the quality of a retrieval system, we compute the following metrics based on labelled test sets. Each test example contains the query and candidates in the corpus, Usually, there should be positive(relevant) candidates, and sometimes with, negative(non-relevant) candidates. The positive candidates sometimes have scaled relevancy labels or are presented as a golden ranked list. We introduce the most widely used metrics for a single query q:

1. **Precision and Recall** of a single query are defined similarly as in the classification tasks:

$$\text{Precision@k}(R_k, q) = \frac{\sum_{d \in R_k} \text{rel}(q, d)}{|R_k|}, \tag{2.13}$$

$$\text{Recall@k}(R_k, q) = \frac{\sum_{d \in R_k} \text{rel}(q, d)}{\sum_{d \in C} \text{rel}(q, d)}, \tag{2.14}$$

   where $\text{rel}(q, d) = 1$ if q and d are relevant and otherwise $\text{rel}(q, d) = 0$, and $R_k = \text{IR}(\mathcal{D}, q, k)$ contains the top k ranked results from the model, C is the whole corpus.

2. **Reciprocal Rank(RR)** measures the rank of the top one positive document:

$$\text{RR@k}(R_k, q) = \frac{1}{argmin_{\{d_i \in R_k, \text{rel}(q, d_i) = 1\}} i}. \tag{2.15}$$

3. **Average Precision(AP)** measures an averaged precision for different i:

$$\text{AP@k}(R_k, q) = \frac{\sum_{i=1}^{k} \text{rel}(d_i, q) \cdot \text{Precision@i}(R_i, q)}{\sum_{d \in C} \text{rel}(q, d)}. \tag{2.16}$$

4. **Normalized Discounted Cumulative Gain(NDCG)** is usually used for datasets with multi-scale relevance labels or ranked lists:

$$\text{DCG@k}(R, q) = \sum_{i=1}^{k} \frac{2^{\text{rel}(q, d_i)} - 1}{log_2(i + 1)}, \tag{2.17}$$

$$\text{nDCG@k}(R_k, q) = \frac{\text{DCG@k}(R_k, q)}{\text{IDCG@k}(R_k, q)}, \tag{2.18}$$

   where IDCG represents the DCG score of the ideal ranked list.

Methods

Now we introduce the widely used methods for the retriever.

Unsupervised Sparse Retrievers: A straightforward way to determine whether a document is relevant to the query is to check the overlapped words. One typical such retrieval system used in DrQA [22] computes the tf-idf weighted bag-of-word vector for all the documents and queries:

$$\overrightarrow{V(d_i)} = (t_1, t_2, ..., t_m)^\top, \tag{2.19}$$

where m is the vocabulary size and t_j is the tf-idf score for word w_j in document d_i:

$$\text{tf-idf}(w_j, d_i, \mathcal{D}) = \text{tf}(w_j, d_i) \cdot \text{idf}(w_j, \mathcal{D}), \tag{2.20}$$

$$\text{tf}(w_j, d_i) = \frac{f_{w_j,d_i}}{\sum_{w \in d_i} f_{w,d_i}}, \ \text{idf}(w_j, \mathcal{D}) = log\frac{|\mathcal{D}|}{|\{d \in \mathcal{D}|t \in d\}|}, \tag{2.21}$$

where $f_{w,d}$ is the raw count of word w in document d, and $|\{d \in \mathcal{D}|t \in d\}|$ is the number of documents where the term w appears. DrQA takes the dot product $\overrightarrow{V(d_i)} \cdot \overrightarrow{V(q)}$ as the score for the query-document pairs and adds n-gram features to catch local word order.

BM25 [209] is the most successful and widely used sparse scoring function, usually performing better than the tf-idf score. It is defined as:

$$\text{BM25}(d, q) = \sum_{w_i \in q} \text{idf}(q, d) \cdot \frac{f_{w_i,d} \cdot (k_1 + 1)}{f_{w_i,d} + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})}, \tag{2.22}$$

where $|d|$ is the document length in words, $avgdl$ is the average document length in the corpus, k_1 and b are hyper-parameters. It can generalize well to different corpus without training and is still a strong baseline compared with neural network-based methods.

While the sparse retrievers provide strong performance, they also suffer in many aspects:

- The usually requires stemming the text before indexing. A small difference in the word will lead to a mismatch.
- They cannot match words semantically.
- They fail to catch word order information and drop it at the beginning.

Supervised Dense Retrievers: With the success of the PLMs in many tasks, people begin to apply them to re-ranking and retrieval. The PLMs-based systems can benefit from the generalization ability and better catch the semantic relevance and word order information. PLMs usually encode all the input in a single sequence to catch the relation between the input parts. To be able to infer within a reasonable time, people start with applying PLMs as re-rankers: such systems first use the heuristic retrievers such as BM25 to select a large number of documents and then apply the fine-tuned PLM-based rankers to form better ranking lists. The PLM-based methods have been shown effective in passage re-ranking tasks [176, 82] Conneau et al. [30] showed that training with SNLI [14] would produce sentence representation models that are generalizable and transferable to other tasks. The Universal Sentence Encoder (USE) [19, 26] went a step further

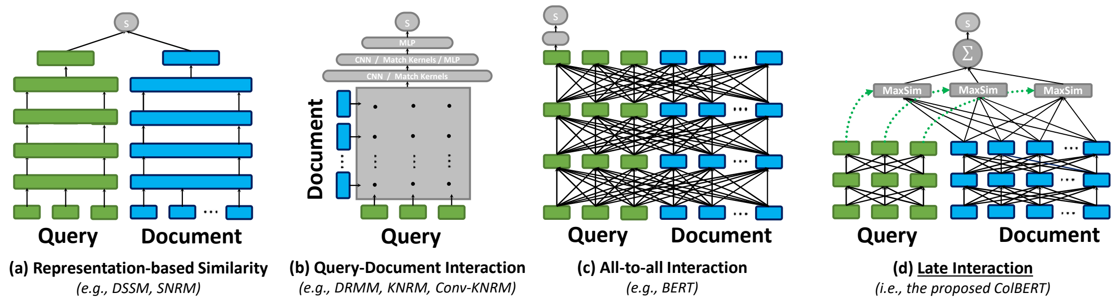


Figure 2.4: Different query-document matching paradigms in neural IR. (Figure from [94])

to propose a transformer encoder trained on multiple tasks; this work adopts USE as the vector representation for documents.

However, such methods need to encode the query and document together, which will be time-consuming if we apply them as the first-stage retrievers. To catch the context-aware semantic importance of a term with a PLM and enable indexing documents before inference, HDCT [35] proposed to stack a linear layer after the BERT model to predict the term importance score and then aggregated the scores into a document-level bag-of-words representation, which can be stored into a standard inverted index for efficient retrieval. Later, ColBERT [94] and DPR [92] proposed to train PLM-based models as first-stage retrievers to generate dense representations of queries and documents separately. Instead of storing an inverted index, such methods store the dense vectors for each document or token, taking advantage of the approximate nearest neighbour search for inference.

The learned retrievers can be categorized into several types according to when will the query and document interact:

- **Representation-based Similarity**: first transform texts into representation vectors, then take the vector similarity as the relevance scores [292].
- **Query-document interaction**: model word and phrase-level relationships across queries and documents and match them using a deep neural network [67, 273, 37].
- **All-to-all interaction**: concatenate the document and query together, and rely on the transformer's all-to-all attention to get the interaction between the query and the document [42].
- **Late interaction** stores the token level representation and applies late interaction at inference with a MaxSim function [94].

Other Retrievers: There are also other retriever methods not falling in these two categories, such as docTTTTquery [177], COIL [59], DeepCT [36], DeepImpact [156], uniCOIL [131]. They all show promising performance in balancing the time-precision space. However, in this thesis, we focus mainly on the RBQA system and take the representative methods: BM25 and DPR.

Training, Indexing and Searching

For large corpus, it is usually too expensive to compute all the documents every time during inference. This makes query-document interaction and all-to-all interaction methods not feasible for first-stage retrievers. Most first-stage dense retrievers follow the representation-based or late interaction methods. Typical construction of neural network-based retrievers has three sub-process: training the scoring model, indexing the corpus, and inference.

Encoder Training: Training the scoring model is usually based on labelled query-document pairs. Such methods usually encode the query and document with separate encoders. Depending on what label we have, we can apply different loss functions as the encoder training objectives:

- **Point-wise loss**: The training data contains (query, positive or negative document) pairs. A cross entropy loss is widely used for training: $L = -\sum_{(q,d^+)} log(s(q,d)) - \sum_{(q,d^-)} log(1 - s(q,d))$, where d^+, d^- represents the positive and negative documents respectively, and $s(q,d)$ is the similarity score predicted by the models.
- **Pair-wise loss**: The training data contains (query, positive-document, negative-document) triplets. People usually train the models with a margin loss: $L = \sum_{(q,d^+,d^-)} max\{0, d(q,d^+) - d(q,d^-) + m\}$, where m is a hyper-parameter represents the margin.
- **List-wise loss**: Listwise approaches directly look at the entire list of documents and try to come up with the optimal ordering for it, for example, SoftRank [231], AdaRank [274] and ListNet [18], ListMLE [105].

Indexing: After the scoring function is trained, we index the whole corpus. The purpose is to optimize speed and performance in finding relevant documents for a search query. We store each document as a sparse vector in the heuristic retrievers, usually a vector of dimension as the vocabulary size. We will store the documents in an *inverted index*, a list of occurrences of each atomic search criterion [312, 313]. Typically, the index is stored as a hash table or binary tree [53, 107]. While for the representation-based retrievers, we represent each document as a dense vector, usually only hundreds of dimensions. Then the document search problem is converted to a vector search in a high-dimensional space. People usually apply an approximate nearest neighbour search to retrieve a "good guess" of the nearest neighbour in return for improved speed or memory savings.

Searching: During serving, the systems will first compute the query representation. Then, the systems will apply the query function of the indexing system, either a lookup of the inverted index or an approximated nearest neighbour search in high-dimensional space.

Codebase

Since the retriever contains several different components, which is more complicated than the other neural machine learning systems, we adopt several tools for easier experiments:

- **Anserini** [135, 279, 280] represents recent efforts by researchers to bring academic IR into better alignment with the practice of building real-world search applications, where Lucene has become the *de facto* platform used in industry. Through an emphasis on rigorous software engineering and regression testing for replicability, Anserini codifies IR best practices today. Lin [128] showed that a well-tuned Anserini implementation of a query expansion model proposed over a decade ago still beats two recent neural models for document ranking.
- **Pyserini** [134] is a Python toolkit for reproducible information retrieval research with sparse and dense representations. Retrieval using sparse representations is provided via integration with the Anserini IR toolkit. Retrieval using dense representations is provided via integration with Facebook's Faiss library. Pyserini is primarily designed to provide effective, reproducible, easy-to-use first-stage retrieval in a multi-stage ranking architecture.
- **Tevatron** [61] is a simple and efficient toolkit for training and running dense retrievers with deep language models. The toolkit has a modularized design for easy research; command line tools are also provided for fast development and testing. A set of easy-to-use interfaces to Huggingface's state-of-the-art pre-trained transformers ensures Tevatron's superior performance.

2.3.4 Question Answering

The reader part originates from the question-answering task, where a system is designed to answer certain questions. Question answering can contain the broadest format of natural language tasks: we can convert any problem into a question and answering format. We mainly consider knowledge-seeking questions, and the answers should be in the related documents. There are also some other question-answering tasks, for example, visual question answering and common knowledge reasoning, which are not in the scope of this proposal.

Following DrQA's structure in an RBQA, we adopt a machine reading comprehension model as the reader. In their paper, this part of the reader follows a Bi-directional LSTM structure, predicting the answer span within the provided paragraphs. Before the success of BERT-based models, the most successful machine reading comprehension structures are based on LSTM, such as BiDAF [217] and R-Net [259].

Methods

There are mainly two formations of the question-answering models: extractive or generative. Extractive question answering means the answers are a sub-string of the given documents, while generative QA means the systems can generate any format of text answers.

Extractive QA: In extractive question answering, the systems take the spans or sentences from the given documents as the answers. We usually take passages as the input unit of the readers. It can be formulated as a sequence labelling problem: Given a question q and a passage $p = (w_1, w_2, ..., w_n)$, where w_i are the words or tokens in the passage and n represents the total number of words, find a span $(i_{\text{start}}, i_{\text{end}})$, where $1 \le i_{\text{start}} \le i_{\text{end}} \le n$ indicates the start and end position of the answer in the given passage. The most common model architectures are stacking the pointer network after the backbones [254, 42]. The mentioned system often trains the model based on a token-wise cross-entropy loss: whether the current position is the start/end of the answer.

Generative QA: Generative question answering considers the answers as generated texts, where the answer string does not have to be a sub-string of the background documents. People usually apply sequence-to-sequence models to tackle such tasks, for example, T5 [199], BART [113] and GPT-based models [181]. The mentioned methods train the models based on the generative loss, usually negative log-likelihood, between the model's output and the referenced answers.

Chapter 3

The Retrieval-Based Question Answering System

3.1 System Architecture: BERTserini

In the work BERTserini [281], we introduce such a simple yet effective RBQA system to deal with the ODQA task based on Wikipedia. We demonstrate an end-to-end question answering system that integrates BERT with the open-source Anserini information retrieval toolkit. Our system integrates best practices from IR with a BERT-based reader to identify answers from a large corpus of Wikipedia articles in an end-to-end fashion. We report large improvements over previous results on a standard benchmark test collection, showing that fine-tuning pretrained BERT with SQuAD v1.1 is sufficient to achieve high accuracy in identifying answer spans. The simplicity of this design is one major feature of our architecture.

3.1.1 Method

The architecture of BERTserini is shown in Figure 3.1 and is comprised of two main modules, the Anserini retriever and the BERT reader. The retriever is responsible for selecting segments of text that contain the answer, which is then passed to the reader to identify an answer span. To facilitate comparisons to previous work, we use the same Wikipedia corpus described in Chen et al. [22] (from Dec. 2016) comprising 5.08M articles. In what follows, we describe each module in turn.

Anserini Retriever: For simplicity, we adopted a single-stage retriever that directly identifies segments of text from Wikipedia to pass to the BERT reader—as opposed to a multi-stage retriever

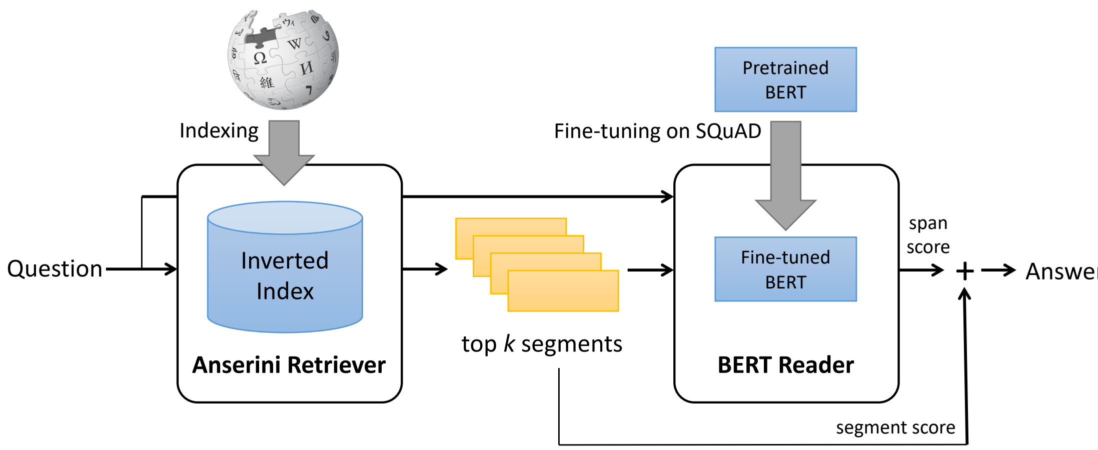


Figure 3.1: Architecture of a RBQA system: BERTserini.

that first retrieves documents and then ranks passages within. However, to increase flexibility, we experimented with different granularities of text at indexing time:

- **Article**: The 5.08M Wikipedia articles are directly indexed; that is, an article is the unit of retrieval.
- **Paragraph**: The corpus is pre-segmented into 29.5M paragraphs and indexed, where each paragraph is treated as a "document" (i.e., the unit of retrieval).
- **Sentence**: The corpus is pre-segmented into 79.5M sentences and indexed, where each sentence is treated as a "document".

At inference time, we retrieve k text segments (one of the above conditions) using the question as a "bag of words" query. We use a v0.3.0 of Anserini,[1] with BM25 as the ranking function (Anserini's default parameters).

BERT Reader: Text segments from the retriever are passed to the BERT reader. We use the model in Devlin et al. [42], but with one important difference: to allow comparison and aggregation of results from different segments, we remove the final softmax layer over different answer spans.

Our BERT reader is based on Google's reference implementation [2] (TensorFlow 1.12.0). For training, we begin with the BERT-Base model (uncased, 12-layer, 768-hidden, 12-heads, 110M

[1]http://anserini.io/
[2]https://github.com/google-research/bert

Model	EM	F1	R
Dr.QA [22]	27.1	-	77.8
R^3 [255]	29.1	37.5	-
[101]	29.8	-	-
Par. R. [108]	28.5	-	83.1
MINIMAL [168]	34.7	42.5	64.0
BERTserini (Article, $k = 5$)	19.1	25.9	63.1
BERTserini (Paragraph, $k = 29$)	36.6	44.0	75.0
BERTserini (Sentence, $k = 78$)	34.0	41.0	67.5
BERTserini (Paragraph, $k = 100$)	**38.6**	**46.1**	**85.8**

Table 3.1: BERTserini results on SQuAD1.1 development questions.

parameters) and then fine tune the model on the training set of SQuAD (v1.1). All inputs to the reader are padded to 384 tokens; the learning rate is set to 3×10^{-5} and all other defaults settings are used.

At inference time, for retrieved articles, we apply the BERT reader paragraph by paragraph. For retrieved paragraphs or setences settings, we directly apply inference over the entired retrieved snippets. In all cases, the reader selects the best text span and provides a score.

We then combine the reader score with the retriever score via linear interpolation:

$$S = (1 - \mu) \cdot S_{\text{Anserini}} + \mu \cdot S_{\text{BERT}}, \tag{3.1}$$

where $\mu \in [0, 1]$ is a hyper-parameter. We tune μ on 1000 randomly selected question-answer pairs from the SQuAD v1.1 training set, considering all values in tenth increments.

3.1.2 Experiments

For evaluation, we mainly show the Exact Match (EM) score, F1-Score of the answers and the Recall (R) of the retrieved passages, as explained in Section 2.3.1. Our main results are shown in Table 3.1, where we report metrics with different Anserini retrieval conditions (article, paragraphs, and sentences). We compare article retrieval at $k = 5$, paragraph retrieval at $k = 29$, and sentence retrieval at $k = 78$. The article setting matches the retrieval condition in Chen et al. [22]. The values of k for the paragraph and sentence conditions are selected so that the reader considers approximately the same amount of text: each paragraph contains 2.7 sentences on average, and

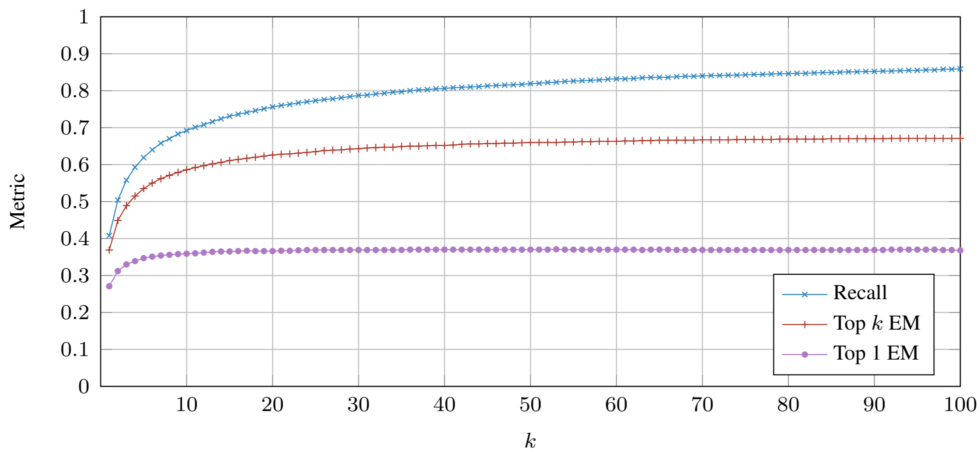


Figure 3.2: BERTserini effectiveness with different numbers of retrieved paragraphs on SQuAD v1.1.

each article contains 5.8 paragraphs on average. We found that setting passages as the retrieve unit provides the best system performance. We were the first to apply BERT-based reader model to the ODQA task, which beats previous systems to a great extend.

We see that article retrieval under-performs paragraph retrieval by a large margin: the reason, we believe, is that articles are long and contain many non-relevant sentences that serve as distractors to the BERT reader. Sentences perform reasonably but not as well as paragraphs because they often lack the context for the reader to identify the answer span. Paragraphs seem to represent a "sweet spot", yielding a large improvement in exact match score over previous results.

Effects of varying k: Our next experiment examined the effects of varying k, the number of text segments considered by the BERT reader. Here, we focus only on the paragraph condition, with $\mu = 0.5$ (the value learned via cross validation). Figure 3.2 plots three metrics with respect to k: recall, top k exact match, and top exact match. Recall measures the fraction of questions for which the correct answer appears in any retrieved segment, exactly as in Table 3.1. Top k exact match represents a lenient condition where the system receives credit for a correctly-identified span in any retrieved segment. Finally, top exact match is evaluated with respect to the top-scoring span, comparable to the results reported in Table 3.1. Scores for the paragraph condition at $k = 100$ are also reported in the table: we note that the exact match score is substantially higher than the previously-published best result that we are aware of.

We see that, as expected, scores increase with larger k values. However, the top exact match score doesn't appear to increase much after around $k = 10$. The top k exact match score continues growing a bit longer but also reaches saturation. Recall appears to continue increasing all the way up to $k = 100$, albeit more slowly as k increases. This means that the BERT reader is unable to take advantage of these additional answer passages that appear in the candidate pool.

These curves also provide a failure analysis: The top recall curve (in blue) represents the upper bound with the current Anserini retriever. At $k = 100$, it is able to return at least one relevant paragraph around 86% of the time, and thus we can conclude that passage retrieval does not appear to be the bottleneck in overall effectiveness in the current implementation. The gap between the top blue recall curve and the top k exact match curve quantifies the room for improvement with the BERT reader; these represent cases in which the reader did not identify the correct answer in any paragraph. Finally, the gap between the red curve and the bottom top exact match curve represents cases where BERT did identify the correct answer, but not as the top-scoring span. This gap can be characterized as failures in scoring or score aggregation, and it seems to be the biggest area for improvement—suggesting that our current approach (weighted interpolation between the BERT and Anserini scores) is insufficient. We are exploring re-ranking models that are capable of integrating more relevance signals.

In this Section, we introduced BERTserini, our design of the RBQA system that integrates BERT and the Anserini IR toolkit. With a simple two-stage pipeline architecture, we are able to achieve large improvements over previous systems. BERTserini has become a reference point for researchers working on this problem [261, 48, 112, 101]. Furthermore, error analysis points to room for improvement in retrieval, answer extraction, and answer aggregation—all of which represent ongoing efforts.

3.2 Trade-offs in the Retriever

A multi-stage pipeline architecture was a widely-adopted design for building retrieval systems, both in real-world deployments [186] as well as in the academic literature [3]. In a common scenario, keyword-based retrieval is utilized to generate a preliminary list of documents. This list is subsequently re-evaluated and re-arranged through one or more subsequent stages. These stages often utilize progressively more computationally intensive methods to iteratively improve the ranking. For instance, they may involve extracting intricate features from the input or applying deep neural networks for inference. However, these approaches are applied to smaller subsets of the initial candidates, ensuring a balance between effectiveness and efficiency. The primary goal of candidate generation (and the early stages) is to maximize the recall of retrieved documents while minimizing query latency. In this context, we explored two designs in Tu et al. [240]:

- **Candidate generation with approximate nearest-neighbors (ANN) search.** We explore whether inverted indexes can be replaced by developments based on ANN search.

RBQA Deployment: We have deployed BERTserini as a chatbot that users can interact with RSVP.ai's intelligent platform that allows businesses to construct natural dialogue services easily and quickly. A screenshot of the RSVP.ai chat platform is shown in Figure 3.3. The current interface uses the paragraph indexing condition, but we return only the sentence containing the answer identified by the BERT reader. The answer span is highlighted in the response [132]. In the screenshot we can see the diversity of questions that BERTserini can handle—different types of named entities as well as queries whose answers are not noun phrases. One important consideration in an operational system is the latency of the responses. Informed by the analysis in Figure 3.2, in our demonstration system we set $k = 10$ under the paragraph condition. While this does not give us the maximum possible accuracy, it represents a good cost/quality tradeoff. To quantify processing time, we randomly selected 100 questions from SQuAD v1.1 and recorded average latencies; measurements were taken on a machine with an Intel Xeon E5-2620 v4 CPU (2.10GHz) and a Tesla P40 GPU. Anserini retrieval (on the CPU) averages 0.5s per question, while BERT processing time (on the GPU) averages 0.18s per question.

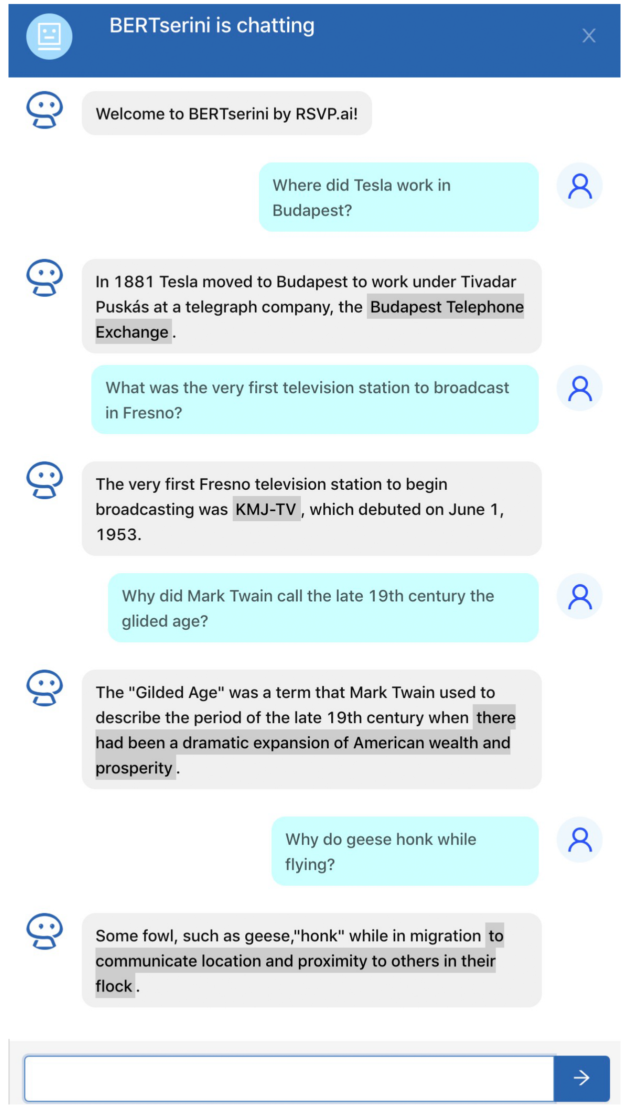


Figure 3.3: Deployment of BERTserini in a chatbot interface.

- **Lightweight dense vector re-ranking.** We examine whether work in representational learning can be deployed as a lightweight re-ranker to achieve better effectiveness–efficiency trade-offs. By lightweight, we mean representations that can be pre-computed and approaches that do not depend on costly neural network inference over candidate documents.

A naïve brute-force approach to nearest-neighbor search quickly becomes impractical as the size of the collection grows. Multi-dimensional indexes (e.g., KD-trees) are not suitable for sparse vectors with very large feature spaces (such as text), but *approximate* solutions based on Local Sensitive Hashing [64] and quantization-based methods [84, 71] have proved workable. However, previous applications to document retrieval [15] are mixed at best. Recently, we have seen a revival in graphical methods for ANN search with the proposal of Hierarchical Navigated Small World (HNSW) graphs [155], which achieves impressive performance in a widely-cited ANN Benchmark [3]. We build on this latest technique for ANN search.

3.2.1 Method

We assume a multi-stage ranking architecture comprising n stages, S_1 to S_n. The initial stage S_1 is responsible for generating a list of k_1 candidates directly from the document collection, and all subsequent stages perform re-ranking given the output of the previous stage. Specifically, stage S_p receives a ranked list of k_{p-1} documents from the previous stage, applies internal re-ranking logic, and passes along a ranked list k_p to the next stage, with the only constraint that $k_p \leq k_{p-1}$.

Candidate Generation

In most multi-stage ranking architectures, the initial candidate generation stage S_1 leverages inverted indexes, typically with bag-of-words queries, to produce documents that are fed to the subsequent re-rankers. This approach is compared against an alternative based on approximate nearest-neighbor (ANN) search on Hierarchical Navigated Small World (HNSW) graphs, constructed from the sparse BM25 representation of the documents. At retrieval time, the query is converted to a tf-idf vector in a similar way as the documents and used to rank the most similar documents based on inner-product distance.

[3]http://ann-benchmarks.com/

Lightweight Re-ranking

Following the candidate generation stage S_1, we designed a lightweight re-ranking stage S_2 based on cosine similarity between dense vector representations. Adopting advances reported in the literature, we selected the Universal Sentence Encoder (USE) [19, 26] for representing documents. This approach can be considered as a prototype of the learned dense retrievers.

Of course, the effectiveness of this re-ranking approach depends on the quality of the document representations. Evidence from the literature suggests that the quality of vector representations degrades, across multiple tasks, as we attempt to encode longer segments of text. We note that USE is recommended for encoding sentences [4], but we wondered if the approach can be extended to paragraphs. In our experiments in Tu et al. [240] show that:

1. For S_1, ANN is faster but less effective; this holds for our sparse BM25 representations, on both sentences and paragraphs.
2. For S_2, our USE encoder is very effective for re-ranking sentences; this isn't surprising, since that was the recommended use case. USE can still be useful for re-ranking paragraphs, but is not as effective, and thus the ANN + USE combination is preferred for a smaller range of operating points, although USE can still benefit index-based approaches;

3.2.2 Experiments

To get an overall picture of how the end-to-end performance is affected by the retrieval stage, we performed the passage retrieval under the open-domain question answering setting. We tested on SQuAD v1.1 dataset [200], following exactly the evaluation procedure of BERTserini [281].

Index-based S_1 stage: For factoid QA, we used the Anserini IR toolkit, matching the setup of Yang et al. [281]. We use BM25 with the settings of $k_1 = 0.9$ and $b = 0.4$. In our results, we refer to this condition as "BM25 (index)".

ANN-based S_1 stage: We used Lucene and `gensim`[5] to generate BM25 sparse document representations, on which HNSW indexing was performed with `nmslib`[6] using inner product distance. We limit the vocab size to 200k. Following previous work, HNSW indexing parameters were set as $M = 35$ and $\text{ef}_{\text{construction}} = 2500$, similar to Fu et al. [55]; $\text{ef}_{\text{search}}$ was kept at 1000. In our results, we refer to this condition as "BM25 (ANN)".

[4] https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3
[5] https://radimrehurek.com/gensim/
[6] https://github.com/nmslib/nmslib

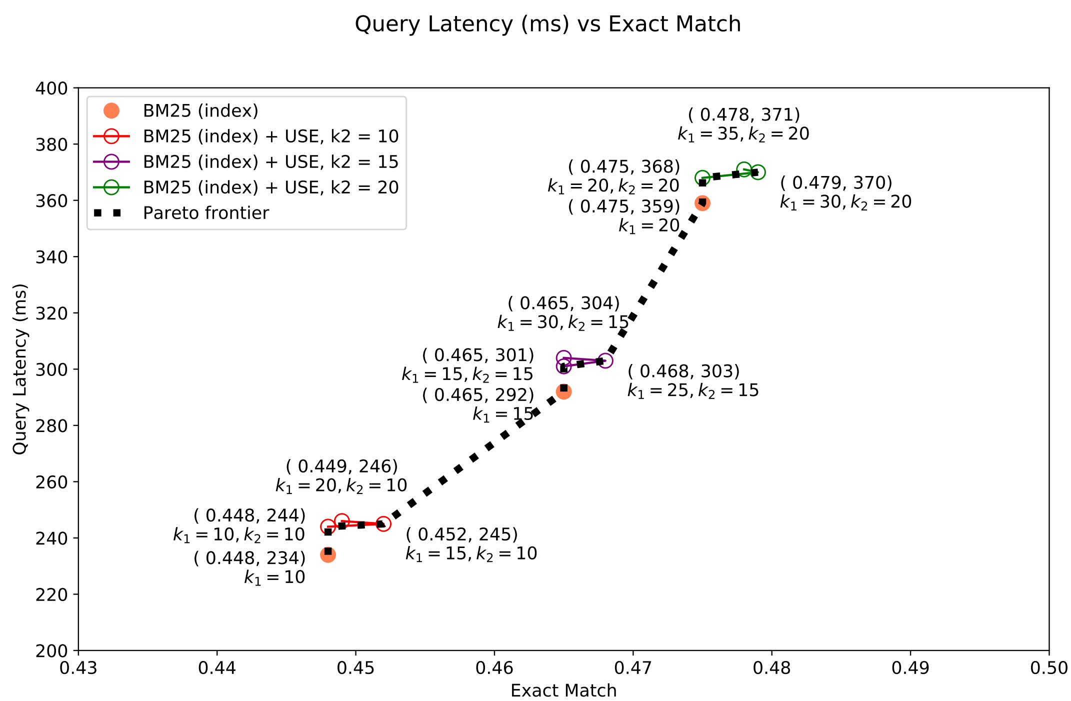


Figure 3.4: The retriever trade-off space for SQuAD v1.1.

USE re-ranking S_2 stage: We use `USE-multilingual-large` as our encoder and feed documents as strings directly without any pre-processing. We did not explicitly differentiate between passages and sentences. Queries are processed in exactly the same way.

The trade-off space for the RBQA system is shown in Figure 3.4. The starting point for comparisons is the two-stage approach of Yang et al. [282]: keyword search with BM25 followed by a BERT reader to extract the exact answer spans, shown with orange circles, sweeping different values of k_1. Here we focus on an operating region with latency that are practical for real-world deployment; higher EM is possible but with latency measured in seconds.

In this context, the index-based approaches are preferred to ANN because the slight latency advantages of the latter are negligible given costly BERT inference. Nevertheless, with each single stage setup there is a comparable setup where we inject USE re-ranking (i.e., retrieve k_1 paragraphs with BM25, re-rank with USE, then feed top k_2 paragraphs to the BERT reader). These are shown as the short colored lines above each solid orange circle. We see that, in each case, we can increase EM slightly at small cost in latency, which provides the developer with an alternative of simply varying k_1 in a single-stage architecture.

3.3 Training the Reader

Now, let's move our focus to the training of the reader models. We fix the underlying system and focus on data augmentation techniques to explore how to best fine-tune BERT in a RBQA system. Following BERTserini [281], we first use passage retrieval to identify relevant paragraphs from Wikipedia and then pass the paragraphs to a BERT reader for answer span extraction (see Figure 3.1). We adopted the "paragraph" setup in BERTserini [281].

One main shortcoming of BERTserini is that it only fine-tune BERT on the original SQuAD dataset, means that the BERT reader is exposed to an impoverished set of examples; all SQuAD data come from a total of only 442 Wikipedia documents. This contrasts with the diversity of paragraphs that the model will likely encounter at inference time in the retrieval-based setting, since they are selected from potentially millions of articles. Also, the output of passage retriever may contain many negative examples (i.e., paragraphs that don't contain answers), which does not accurately match the prevalence of answers in the original training data. Additionally, when performing QA on Wikipedia using a machine-reading comprehension dataset, we do not have labels for most paragraphs (since they are not from the original source documents). The solution, of course, is to fine-tune BERT with labeled paragraphs of the type that it is likely to see at inference time. Distant supervision can provide a bridge.

We further build on BERTserini's basic design and explores how much further we can improve effectiveness by data augmentation alone. To illustrate the robustness of our methods, we also demonstrate gains on another English QA dataset and present results for two additional Chinese QA datasets (which have not to date been evaluated in a retrieval-based setting). The primary contribution is a thorough exploration of the design space of distant supervision techniques for question answering.

The roots of the distant supervision techniques we use trace back to at least the 1990s [287, 207], although the term had not yet been coined. Such techniques have recently become commonplace, especially as a way to gather large amounts of labeled examples for data-hungry neural networks and other machine learning algorithms. Specific recent applications in question answering include Bordes et al. [13], Chen et al. [22], Lin et al. [140], as well as Joshi et al. [90] for building benchmark test collections. However, as we explain below, our approach aims to address a number of issues that are specific to the retrieval-based setting. While such approaches have been proposed previously, we examine several novel aspects:

- **Negative Samples:** Most previous work on distant supervision focuses on generating positive examples. However, under a RBQA system, where the reader consumes the output of passage retrieval, the model will encounter many non-relevant passages, which means that a data collection strategy focused only on positive examples will be inadequate. We show that using existing datasets to identify negative training examples is critical to effectiveness, and explore different strategies for gathering negative examples.

- **Data Fusion:** Another unexplored area in previous work is how to integrate existing training data with data gathered via distant supervision. We show that a naïve strategy of simply combining all data may not be the best approach. Instead, we propose a stage-wise approach to fine-tuning BERT with heterogeneous data, beginning with the dataset that is "furthest" from the test data and ending with the "closest".

3.3.1 Method

Starting from a source dataset comprising question–answer pairs (for sample, SQuAD), we can create additional training examples by fetching paragraphs from the corpus using the system's own passage retrieval algorithm (with the question as the query) and give these paragraphs labels based on the ground truth answers provided in the source dataset. Using the system's own passage retrieval algorithm ensures that the model is exposed to the types of input it will receive at inference time.

Sample Selection: We denote a paragraph as a positive example if the ground truth answers appear in it; otherwise we consider it a negative example. We keep the best positive example for each question (i.e., the highest BM25 score), and examined three different methods for selecting negative examples:

- **Top-down**: We choose negative examples with the highest paragraph scores from the retrieved paragraphs.
- **Bottom-up**: We choose negative examples with the lowest paragraph scores from the retrieved paragraphs.
- **Random**: We randomly sample negative examples from the retrieved paragraphs.

Our intuition is that top-down sampling selects "hard" examples, since the paragraphs have high BM25 scores, while bottom-up sampling selects "easy" examples. Random sampling attempts to obtain diversity. One important hyper-parameter is the number of negative examples to select for each positive instance; ideally, this distribution should match the actual prevalence of correct answers that the reader encounters at test time, giving the model an accurate prior for answer correctness.

Training Set Fusion: Another important design decision is how to make use of both the source data (SRC) and the distantly-supervised data (DS). We refer to the training set of only positive augmented examples as DS(+) and refer to the training set contains both positive and negative examples as DS(±). Other than different datasets we can use, there are three possibilities when considering the fine-tuning order:

- **SRC + DS**: Fine-tune BERT with all data, "lumped" together as a single, larger training set. In practice, this means that the source and augmented data are shuffled together.
- **DS → SRC**: Fine-tune the reader in stages, first on the augmented data and then the source dataset.
- **SRC → DS**: Fine-tune the reader in stages, on the source dataset and then the augmented data.

3.3.2 Experiments

To show the generalizability of our data augmentation techniques, we conducted experiments on two English datasets: SQuAD (v1.1) and TriviaQA [90] (the unfiltered version). The 2016-12-21 dump of English Wikipedia is taken as the document collection from which we are retrieving answers, following Chen et al. [22]. We also examine two Chinese machine reading comprehension datasets: CMRC [31] and DRCD [220]. CMRC is in simplified Chinese while DRCD is in traditional Chinese. For Chinese, we use the 2018-12-01 dump of Chinese Wikipedia, tokenized with Lucene's `CJKAnalyzer` into overlapping bigrams. We apply hanziconv [7] to transform the corpus into simplified characters for CMRC and traditional characters for DRCD. Since we don't have access to the ground truth annotations of the test sets of the four datasets, we use the development sets in each dataset as the test sets. We use BM25 retriever under the same setting ($k_1 = 0.9, b = 4$) as in BERTserini. The retriever returns 100 paragraphs and then feeds them into the BERT reader.

Statistics for the datasets are shown in Table 3.2 DS(+) and DS(±) refer to our augmented dataset with positive and positive as well as negative examples, respectively. For reference, we also provide statistics for our distantly-supervised data; these represent the best settings, corresponding to our main results in Table 3.3, which uses a positive/negative ratio of 1:7, 1:3, 1:7, and 1:6, respectively, using random sampling.

For model training, we begin with the pre-trained BERT-Base model (uncased, 12-layer, 768-hidden, 12-heads, 110M parameters). We use the pre-trained Chinese BERT-Base for the

[7] https://pypi.org/project/hanziconv/0.2.1/

	SQuAD	TriviaQA	CMRC	DRCD
Train	87,599	87,622	10,321	26,936
Test	10,570	11,313	3,351	3,524
DS(+)	64,244	264,192	8,596	41,792
DS(±)	447,468	789,089	68,696	246,604

Table 3.2: Statistics of distant supervision augmented question answering datasets.

Chinese datasets. All inputs are padded to 384 tokens; the learning rate is set to 3×10^{-5} and all other default settings are used. For all datasets, when applying stage-wise fine-tuning, we first fine-tune on the source dataset (SRC) for two epochs, and then on the augmented dataset for one epoch.

Main Results: Our main results on all four datasets are shown in Table 3.3. These results are obtained from the dataset denoted DS(±) in Table 3.2; the training regime is the best stage-wise configuration, which will be introduced in the next section. In later parts we present a number of detailed analyses and contrastive conditions under slightly different experimental procedures, and so there is no exact one-to-one correspondence between results in the main results table and subsequent results. We updated the Anserini retriever in this version, so the performance improved on SQuAD v1.1 from the first BERTserini version. We denote the new implementation as BERTserini*. To the best of our knowledge, there is no previous work on these two Chinese datasets in the retrieval-based setting, and therefore BERTserini* is the only baseline available. We can see that our data augmentation leads to large improvements across all datasets, both in terms of exact match as well as F_1. On all four datasets, these results represented the state of the art.

Effects of Varying k: In Figure 3.5 and Figure 3.6, we show the effects of varying the number of paragraphs k that is fed into the BERT reader on SQuAD v1.1 and CMRC. We plot the following metrics:

- **Top 1 Exact Match (EM)**: we feed the top k paragraphs from the retriever to the reader, the reader extracts the best phrase in each paragraph, and then the aggregator selects the best answer among all the candidates. If the best answer is the same as the ground truth, the system receives credit.
- **Top k Exact Match (EM)**: we feed the top k paragraphs from the retriever to the reader, then the reader extracts the best phrase in each paragraph. If any one of these phrases matches the ground truth answer, the system receives credit.

SQuAD			
Model	EM	F_1	R
Dr.QA [22]	29.8	-	-
R^3 [255]	29.1	37.5	-
Kratzwald and Feuerriegel [101]	29.8	-	-
Par. R. [108]	30.2	-	-
MINIMAL [168]	34.7	42.5	64.0
ORQA [112]	34.7	-	64.0
RankQA [102]	35.8	-	-
DenSPI-Hybrid [218]	36.2	44.4	-
MUPPET [48]	39.3	46.2	-
RE^3QA-Large [79]	41.9	50.2	-
Multi-passage BERT [261]	51.2	59.0	-
BERTserini*	41.8	49.5	86.3
Ours	**51.2**	**59.4**	86.3
TriviaQA			
Model	EM	F_1	R
ORQA [112]	47.2	-	-
R^3 [255]	47.3	53.7	-
DS-QA [140]	48.7	56.3	-
Evidence Agg. [256]	50.6	57.3	-
BERTserini*	51.0	56.3	83.7
Ours	**54.4**	**60.2**	83.7
CMRC			
Model	EM	F_1	R
BERTserini*	44.5	60.9	86.5
Ours	**48.6**	**64.6**	86.5
DRCD			
Model	EM	F_1	R
BERTserini*	50.7	65.0	81.5
Ours	**55.4**	**67.7**	81.5

Table 3.3: Effectiveness of distant supervised reader on two English and two Chinese datasets.

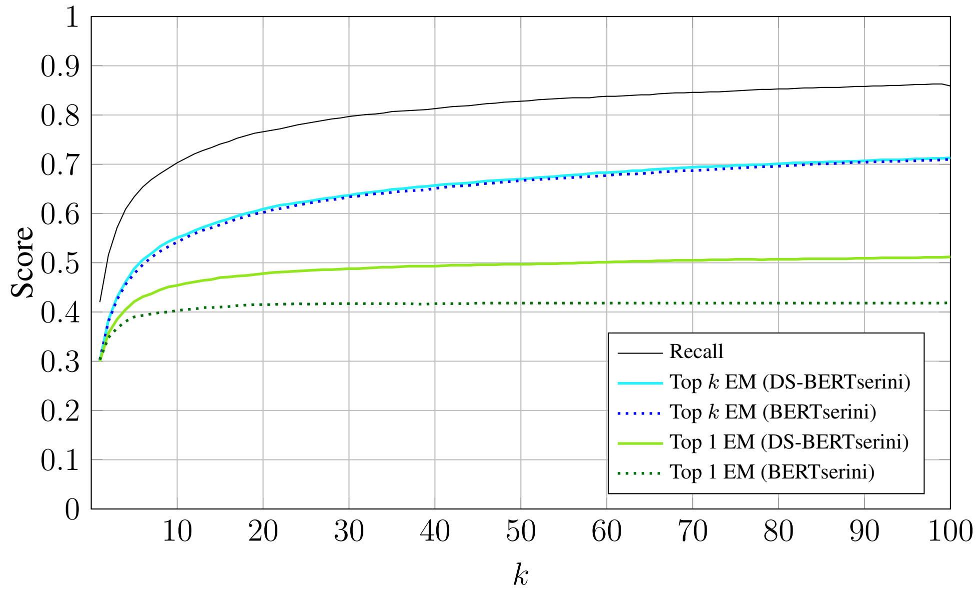


Figure 3.5: Effects of the number of retrieved paragraphs k on SQuAD.

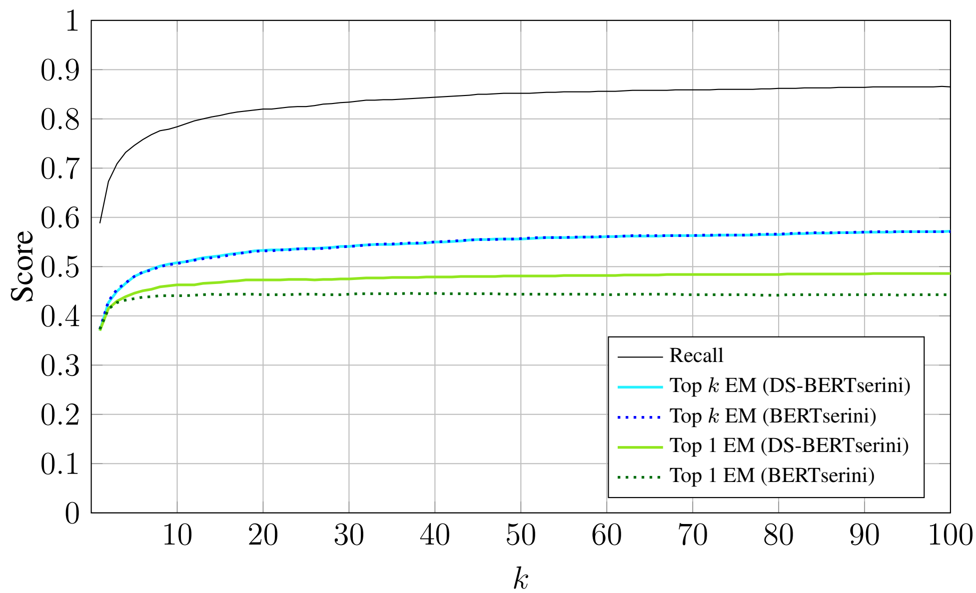


Figure 3.6: Effects of the number of retrieved paragraphs k on CMRC.

- Recall: the fraction of questions in which at least one retrieved passage contains the answer (which provides an upper bound on scores with the current passage retrieval approach).

What is interesting here is to compare the effects of data augmentation, which we show with dotted vs. solid lines; note that the passage retriever (hence, recall) remains the same. We see that top 1 EM improves quite a bit, although top k EM changes little. This suggests that distantly-supervised data is helping the model extract better answer phrases—likely due to improved context modeling—but isn't fundamentally expanding the coverage of the reader. We note the same phenomenon in Chinese. However, the gap here between top k EM and upper bound recall is much larger, which means that the passage retriever is finding relevant paragraphs, but the reader model is unable to identify the correct answer spans. We suspect that the Chinese BERT model is weaker than the English BERT model. The SRC-trained BERT-Base model achieves 80 EM and 88 F_1 on SQuAD 1.1, while only 65 EM and 83 F_1 on CMRC. These results are comparable to Sun et al. [227]). Also, the two Chinese datasets lack diversity in both questions and answers compared to English datasets.

Ablation Studies

Fine-tuning Order: Having acquired the augmented data with our distant supervision techniques, there are a number of options for fine-tuning the model using all available data. Experiments exploring these possibilities are presented in Table 3.4 for each dataset. The rows marked SRC refer to fine-tuning with the source data only, which is the same as the BERTserini* baseline in Table 3.3. While training with positive examples, denoted DS(+), improves effectiveness as expected, an even larger gain comes from leveraging positive and negative examples, denoted DS(±). Note that negative sampling here was performed using the top-down approach (see next), which is slightly less effective than the random sampling approach presented in the Table 3.3; thus, these numbers are not directly comparable.

The natural question is how to take advantage of both types of data (source and augmented). The most obvious approach is to simply lump both the source and augmented data together to create a single, larger training set. This is shown in Table 3.4 as SRC + DS(±), and for three out of the four datasets, it is not the best approach (we discuss the exception below). In fact, the SRC + DS(±) condition performs worse than just using the augmented data alone on the English datasets.

We propose that heterogeneous datasets should be leveraged in a stage-wise manner, starting with the dataset that is "furtherest" away from the test data. That is, we wish to take advantage of all available data, but the last dataset we use to fine-tune BERT should be "most like" the test data

	SQuAD		TriviaQA		CMRC		DRCD	
	EM	F_1	EM	F_1	EM	F_1	EM	F_1
SRC	41.8	49.5	51.0	56.3	44.5	60.9	50.7	65.0
DS($\pm$)	48.7	56.5	**54.4**	**60.2**	48.3	63.9	53.2	66.0
SRC+DS($\pm$)	45.7	53.5	53.1	58.6	49.0	64.6	**55.4**	**67.7**
DS($\pm$) $\rightarrow$ SRC	47.4	55.0	49.8	55.9	45.6	61.9	53.4	67.1
SRC $\rightarrow$ DS($\pm$)	**50.2**	**58.2**	53.7	59.3	**49.2**	**65.4**	54.4	67.0

Table 3.4: Results of approaches to combining source and augmented training data under retrieval-based question answering datasets: SQuAD v1.1, TriviaQA, CMRC and DRCD.

the model will see at evaluation time. Since the augmented data is drawn from the same passages that the system is ultimately evaluated on, we should fine-tune on those data last. This condition is denoted SRC $\rightarrow$ DS($\pm$), which yields the best results for two of the datasets. Note that if we swap the tuning order, DS($\pm$) $\rightarrow$ SRC, effectiveness drops (and quite a bit in three of the four datasets), which lends credence to our heuristic.

However, two of the datasets appear to behave differently. With TriviaQA, the best condition is to simply disregard the source dataset, i.e., fine-tune with the augmented data only. We believe this may be an artifact of how the dataset was constructed to begin with: TriviaQA source paragraphs are already the product of distant supervision (on noisy web texts), which means that they are of lower quality of begin with. Our data augmentation procedure actually yields *higher quality* training data, since our examples are from Wikipedia and thus better match the passages the BERT reader encounters at evaluation time. For DRCD, we see that SRC + DS($\pm$) is the most effective overall, which we explain by how the dataset is constructed. According to Shao et al. [220], the dataset was assembled by searching Wikipedia (10k paragraphs from 2.1k pages). In this case, the augmented data are similar enough to the source data that lumping them both together to create a single, larger dataset is the best strategy. This observation is affirmed by the fact that flipping the order of fine tuning, DS($\pm$) $\rightarrow$ SRC, yields only a small drop in accuracy (much less than on the other datasets).

While the best conditions appear to be idiosyncratic for two of our datasets (artifacts of how they were constructed), we believe that our proposed stage-wise tuning approach makes intuitive sense. Another way to think about using different datasets is in terms of a very simple form of transfer learning. The stage-wise fine-tuning strategy is in essentially trying to transfer knowledge from labeled data that is not drawn from the same distribution as the test instances. We wish to take advantage of transfer effects, but limit the scope of erroneous parameterization. Thus, it makes sense not to intermingle heterogeneous, qualitatively different datasets, but to fine-tune the

	SQuAD		CMRC	
	EM	F1	EM	F1
Top-down	49.2	57.2	48.8	64.5
Bottom-up	46.8	54.9	48.6	65.2
Random	49.6	57.6	48.6	64.7

Table 3.5: Effects of different negative sampling strategies on SQuAD and CMRC.

model in distinct stages.

Negative Sampling Strategies: One key insight of our work is that negative sampling is critical to the effectiveness of distant supervision techniques, particularly in our retrieval-based setting where the reader is only exposed to the output of the passage retriever. We present results that explore different negative sampling strategies.

Table 3.5 shows the effectiveness of different negative sampling strategies on two datasets: SQuAD for English and CMRC for Chinese. In order to better isolate the impact of data augmentation, these experiments do not take advantage of the source data, and so the figures are not directly comparable to the main results in Table 3.3. Recall that the top-down negative sampling strategy selects "difficult" examples (i.e., those with high BM25 scores); for English, this is more effective than bottom-up sampling, which selects "easy" examples. However, results suggest that random sampling beats both approaches, showing the importance of selecting diverse examples. Interestingly, though, for Chinese, the sampling strategy doesn't make much of a difference. This was a curious finding that prompted us to examine the retrieved paragraphs manually; as it turns out, passage retrieval in Chinese does not produce many paragraphs that are relevant to the question, but do not contain the answer because of our simple bigram-based retrieval approach. In other words, all of the negative examples are "easy", regardless of sampling strategy, and hence we observe few effectiveness differences.

The ratio between positive and negative examples that are generated via our distant supervision technique is another important hyper-parameter. In Figure 3.7, we plot the effect of this ratio d; i.e., for every positive example, we select d negative examples. We used a similar setup as the experiment above, and therefore these results are also not directly comparable to Table 3.3. Furthermore, in order to experiment with $d < 1$, we applied a slightly different approach to negative sampling: Instead of pairing negative examples with positive examples, we first gathered all positive and negative examples across all questions, and then performed the sampling on those two groups given the ratio d. These results appear somewhat noisy, but suggest that $d > 2$ yields reasonable results—and from the perspective of minimizing training time, we desire the smallest d value that allows us to fully exploit distant supervision. The same experiments on the other

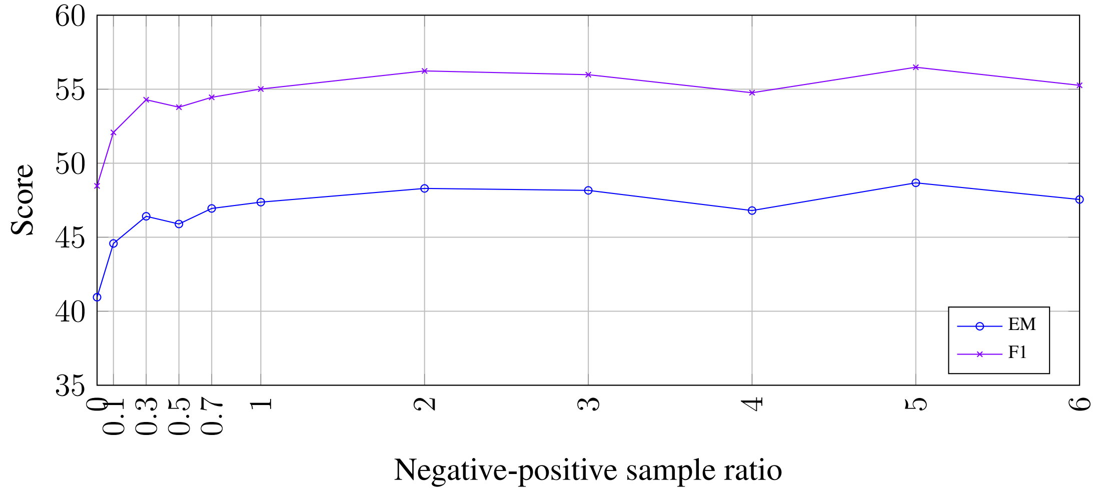


Figure 3.7: Effects of varying d, the positive–negative ratio of examples, on SQuAD v1.1.

datasets yield similar conclusions.

Analysis

Performance by Question Types: In Table 3.6, we further break down effectiveness by question type for SQuAD and CMRC, comparing the BERTserini* baseline and the model trained with data augmentation (corresponding to the main results in Table 3.3). Question classification in English is performed simply based on the initial question word. For consistency, we translated all Chinese questions into English with a state-of-the-art machine translation system[8], and applied the same classification scheme as the English questions. We manually examined translations of the Chinese questions and confirmed that they look reasonable.

For SQuAD, we see that effectiveness on "how" and "why" questions are lower than the other categories (which makes sense since they are more difficult), but surprisingly the scores for "where" questions are quite low as well. We see that data augmentation improves performance across the board, although gains from "why" questions are quite limited—this is likely because the exact phrasing of the ground truth answers for these questions are difficult to find. For Chinese, "why" questions appear more difficult than the other types (not surprising). Interestingly, we see that data augmentation actually decreases effectiveness for some question types; however, most categories do see large gains.

[8]Google Translate API in 2019

Question		BERTserini*		Ours	
Type	**percentage**	**EM**	**F1**	**EM**	**F1**
SQuAD					
Who	13.0%	43.1	48.6	52.9 (+22.7%)	59.3 (+22.0%)
When	7.9%	46.8	54.7	56.9 (+21.6%)	65.0 (+18.8%)
Where	4.7%	28.7	38.2	41.1 (+43.2%)	51.4 (+34.6%)
What	54.8%	42.9	50.6	51.8 (+20.7%)	60.3 (+19.2%)
Which	6.3%	46.1	53.5	58.6 (+27.1%)	65.8 (+23.0%)
How	11.0%	36.6	44.5	45.0 (+23.0%)	52.5 (+18.0%)
Why	1.4%	34.2	51.4	36.2 (+5.8%)	54.2 (+5.4%)
Others	0.9%	28.9	49.5	40.0 (+38.4%)	52.6 (+6.3%)
CMRC					
Who	8.0%	47.9	62.6	46.3 (−3.2%)	63.4 (+1.3%)
When	6.0%	50.0	67.5	58.8 (+17.5%)	73.3 (+8.6%)
Where	10.5%	43.6	61.3	52.2 (+19.7%)	67.1 (+9.4%)
What	48.0%	40.8	55.8	44.3 (+8.6%)	59.2 (+6.2%)
Which	13.2%	57.6	71.3	63.8 (+10.6%)	76.3 (+7.0%)
How	8.4%	43.4	63.8	48.5 (+11.9%)	69.8 (+9.5%)
Why	5.3%	34.3	55.5	32.6 (−5.1%)	53.9 (−2.8%)
Others	0.4%	35.7	53.0	35.7 (+0.0%)	45.3 (−14.5%)

Table 3.6: Breakdown of effectiveness by question type on SQuAD (top) and CMRC (down).

Question	Answer	Answers from BERTserini*	Answers from augmented model
Super Bowl 50 decided the NFL champion for what season?	2015, the 2015 season	Super Bowl XXXVII was an American football game between the American Football Conference (AFC) champion Oakland Raiders and the National Football Conference (NFC) champion Tampa Bay Buccaneers to decide the National Football League (NFL) champion for the **2002** season.	Super Bowl 50 decided the **2015** NFL Champion and was played at Levi's Stadium in Santa Clara, California on Sunday, February 7, 2016.
How did Tesla know he was being struck by the particle?	He could feel a sharp stinging pain (where it entered his body)	On 11 July 1934 the "New York Herald Tribune" published an article on Tesla, in which he recalled an event that would occasionally take place while experimenting with his single-electrode vacuum tubes; a minute particle would break off the cathode, pass out of the tube, and **physically** strike him. "Tesla said he could feel a sharp stinging pain where it entered his body, and again at the place where it passed out."	On 11 July 1934 the "New York Herald Tribune" published an article on Tesla, in which he recalled an event that would occasionally take place while experimenting with his single-electrode vacuum tubes; a minute particle would break off the cathode, pass out of the tube, and physically strike him. "Tesla said **he could feel a sharp stinging pain where it entered his body**, and again at the place where it passed out."

Table 3.7: Sample questions and answers comparing distantly supervised model with BERTserini*.

An Example: How is data augmentation improving the reader? At a high-level, we believe that greater diversity in answers gives the reader a more accurate model of answer contexts. We present two examples that highlight our intuitions in Table 3.7, which provides answers (bolded) in the source paragraphs before and after data augmentation. In the first example, we notice that the model before augmentation extracts the phrase right before the word "season" in the answer paragraph. We believe this is an artifact of how some questions are constructed in SQuAD, where a question that contains "... what *noun*" is directly answered by "... X *noun*" in a text span (i.e., the model learns pattern matching).

Thus, a model trained solely on SQuAD is likely to over-fit; via distant supervision, this is avoided because of exposure to sufficient answer diversity. In the second example, the baseline model frequently answers "how" question with an adverb, in part because there are relatively few examples to learn from (note that the question is best categorized as a "why" question, which has even fewer examples). Here, we believe that distant supervision provides more answer instances that allow the model to better learn answer contexts.

Discussions

A noteworthy advantage of our approach is its simplicity and the fact that our gains are demonstrated on a simple reader model. This means that, essentially, gains come "for free". Furthermore, our techniques should be applicable to other neural models as well, including improvements to the basic BERTserini model architecture proposed by other researchers [261, 48, 112, 101, 79, 218, 218]. These findings confirm perhaps something that machine learning practitioners already know too well: "there no data like more data".

Chapter 4

Zero-shot Adaptation to Domains

Training the retrieval-based information access system requires a large amount of labelled data. For the retrieval part, we need query-paragraph pairs with their relevance labelled. For the reader part, we need labelled question-answer pairs. There are many such datasets for general domain information retrieval and question answering. However, to apply this technology to real-life scenarios, we need to twit the models to fit the specific domains. It is not possible to construct individual datasets for each application scenario.

This chapter first discusses methods to distinguish between domains and how zero-shot systems would perform on them. We then introduce several methods that transfer the RBQA system learned in the general domain to applications in specific domains.

4.1 Diverse Domains and Their Challenges

We start with an introduction to several specific domains, which have different vocabulary distribution and language usage from the general domain.

4.1.1 Method to Distinguish Domains

We adopt the more general definition of domains. We treated the texts gathered from a certain source and used them for a specific purpose in the same domain. The text in the same domain usually shares similar background knowledge and language style. We consider the corpus that targets the general public as the general domain. For example, Wikipedia, Common Crawl and the Book Corpus are all considered in the general domain. In contrast, we consider the corpus that targets people with specialized knowledge as specific domains. We give several examples following.

To understand what a domain is quantitatively, Pogrebnyakov and Shaghaghian [193] proposed to apply several methods to compute the domain similarity to predict the success of domain adaptation in text similarity, including:

- **Uni-gram coverage**, the overlap between corpus' vocabulary, using Jaccard similarity.
- **KL-divergence** between the vocabulary distributions.
- **Language model perplexity** [33]. First, train a tri-gram language model in each domain and evaluate the perplexity in other domains.
- **Word vector variance** [33], the word vector variance under word2vec skip-gram trained embeddings.

We conduct a similar analysis from the retrieval side as in BEIR [233] to demonstrate the corpus cluster. In Figure 4.1, we first tokenize the corpus with word piece tokenizer, which maps all the corpus into the same token space as in the most widely used backbone "bert-base-uncased" [1]. Then we compute the Jensen–Shannon divergence between each corpus token distribution and the query set of each dataset. Red represents low divergence, meaning highly related dataset, while green represents high divergence, meaning less related dataset. The corpus based on Wikipedia generally has a higher overlap with other domains, and the bio-medical domain has a high overlap with the scientific domain.

4.1.2 Datasets

The Biomedical Domain: Science domains are the ones that contain many specialized words, and the language style is usually more formal. It often contains many proper nouns, which are rare in the general domain. This also includes abbreviations referring to different entities, common words with uncommon meanings or uncommon word phrases. We choose the BioASQ dataset [239] as an example, which is a large-scale biomedical semantic indexing and question answering (QA) dataset based on PubMed documents[2]. It aims to assess the ability of systems to index large numbers of biomedical scientific articles semantically and to return concise and

[1] https://huggingface.co/bert-base-uncased

[2] https://pubmed.ncbi.nlm.nih.gov/

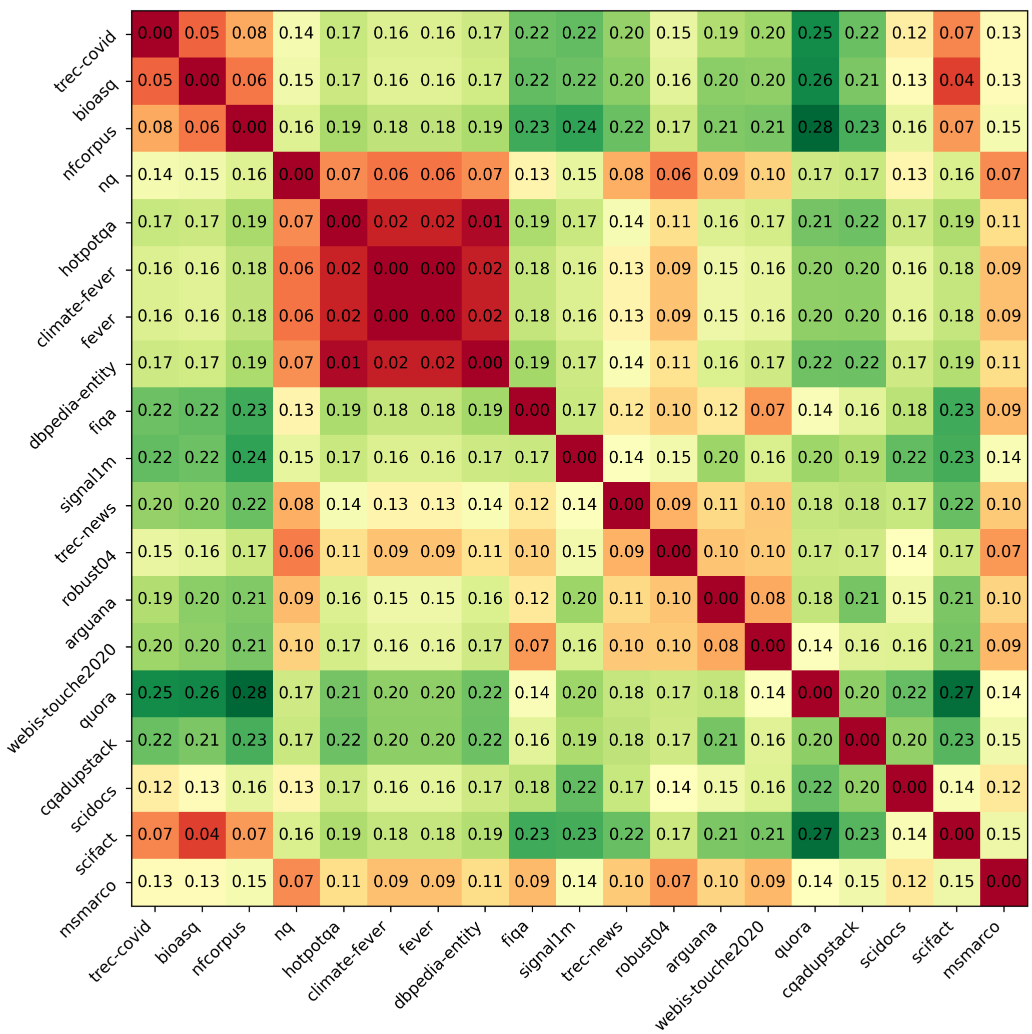


Figure 4.1: Jensen–Shannon divergence between corpus in BEIR

user-understandable answers to given natural language questions by combining information from biomedical articles and ontologies. A team of biomedical experts constructs the answers.

We mainly focus on Task B: Biomedical Semantic QA, which involves IR, QA, and summarization. The questions can be split into four categories: yes/no, factoid, list and summary. The factoid and list questions have the exact answer label, while all the questions have an ideal answer label, a sentence from the provided paragraphs. We unified the answer format into question, answer and evidence. Besides taking the answers for exact match evaluations, we also take the one-sentence reference in the original dataset as evidence and evaluate the models' F1 score for generative models. Following is an example of a yes/no question:

- Question: *Are long non coding RNAs spliced?*
- Answers: *Yes*
- Evidence: *Long non coding RNAs appear to be spliced through the same pathway as the mRNAs.*

The Finance Domain: The finance domain search is another interesting and meaningful application of the RBQA system. Automatic analysis of financial news and opinions relies on interpreting multiple unstructured and structured data sources, demanding fast and comprehensive decision-making. We take the FiQA [154] as our main experiment data sets. FiQA focuses on advancing state-of-the-art of aspect-based sentiment analysis and opinion-based QA for the financial domain. Specifically, we focus on Task 2: Opinion-based QA over financial data – Given a corpus of structured and unstructured text documents from different financial data sources in English (microblogs, reports, news), build a Question Answering system that answers natural language questions. However, the answers in this dataset are mainly paragraphs, so we only need the retrieval part of our system, and we also evaluate from the retrieval perspective [3]. An example from FiQA is shown below:

- Question: *Why are big companies like Apple or Google not included in the Dow Jones Industrial Average (DJIA) index?*
- Answers: *That is a pretty exclusive club and for the most part they are not interested in highly volatile companies like Apple and Google. Sure, IBM is part of the DJIA, but that is about as stalwart as you can get these days. The typical profile for a DJIA stock would be one that pays fairly predictable dividends, has been around since money was invented, and are not going anywhere unless the apocalypse really happens this year. In summary, DJIA is the boring reliable company index.*

[3] The original dataset only evaluates the pair-wise classification accuracy, which is a relatively easy evaluation.

Model (→)	Lexical	Sparse			Dense				Late-Interaction	Re-ranking
Dataset (↓)	BM25	DeepCT	SPARTA	docT5query	DPR	ANCE	TAS-B	GenQ	ColBERT	BM25+CE
MS MARCO	0.228	0.296‡	0.351‡	0.338‡	0.177	0.388‡	0.408‡	0.408‡	0.401‡	**0.413‡**
BioASQ NDCG@10	0.465	0.407	0.351	0.431	0.127	0.306	0.383	0.398	0.474	**0.523**
BioASQ R@100	**0.714**	0.699	0.351	0.646	0.256	0.463	0.579	0.627	0.645	**0.714**
FiQA-2018 NDCG@10	0.236	0.191	0.198	0.291	0.112	0.295	0.300	0.308	0.317	**0.347**
FiQA-2018 R@100	0.539	0.489	0.446	0.598	0.342	0.581	0.593	**0.618**	0.603	0.539

Table 4.1: Domain and zero-shot performances on BioASQ and FiQA retrieval datasets [233].

Model	Rouge-2(R)	Rouge-2(F1)	Rouge-SU4(R)	Rouge-SU4(F1)
Best Supervised Model[4]	0.58	0.36	0.58	0.35
BM25 + BERT(open-squad)	0.13	0.05	0.17	0.05
DPR+BERT(open-squad)	0.06	0.02	0.09	0.03

Table 4.2: RBQA system performance on BioASQ TaskB.

4.2 Baselines

4.2.1 Direct Application of General Domain Systems

The BEIR benchmark [233] has comprehensively analyzed the performance of out-of-the-box retrieval models on zero-shot domain retrieval datasets. These models are initially trained on the MS MARCO dataset and then evaluated on other domain-specific datasets, such as BioASQ and FiQA, which are also interesting for our evaluation. The results of this evaluation are presented in Table 4.1, showing the performance of different retrieval models on these diverse domains.

Our primary focus is on evaluating the performance of RBQA systems. We consider applying RBQA systems trained on the general domain to in-domain datasets to establish a baseline. Our baselines consist of two widely-used retrievers, BM25 and DPR, combined with a classic BERT reader. The results, presented in Table 4.2, compare the performance of these out-of-the-box RBQA models with the current state-of-the-art supervised model on the BioASQ dataset. It is evident that directly applying the retriever and reader trained on the SQuAD-Open dataset yields relatively low scores compared to domain fine-tuned models. This observation underscores the significant gap when applying deep neural models to specific domains, which is the primary motivation for our research.

Model	Shots	NDCG@10	R@100	Learning Rate	Batch Size	Train Steps
BM25	0	0.236	0.539	-	-	-
DPR	0	0.228	0.342	-	-	-
DPR	50	0.239	0.302	3e-7	32	128
DPR	100	0.243	0.301	3e-6	64	94
DPR	1000	0.247	0.310	3e-6	64	313

Table 4.3: Few-shot DPR fine-tune performance on FiQA.

4.2.2 Few-shot Training

In this section, we aim to address the research question: "How much labelled training data is required to fine-tune a pre-trained model for specific domains effectively?" According to the Vapnik-Chervonenkis (VC) dimension theory [244], the amount of labelled training data needed is typically around ten times the dimensionality of the data. Considering that our pre-trained models often have several hundred embedding dimensions, we need at least several thousand labelled samples. However, since we have already pre-trained the model, the question arises as to whether we still require such a large volume of labelled domain-specific data. OpenAI's API guide provides some insights into this matter, suggesting that "the more training examples you have, the better". They recommend having at least a couple of hundred examples, and they have observed that each doubling of the dataset size leads to a linear increase in model quality. We conducted a few-shot fine-tuning experiment using the DPR model to gain a deeper understanding of the data volume required for fine-tuning. By varying the size of the labelled training dataset, we aim to evaluate the impact on the model's performance and determine the threshold at which further improvements become marginal.

To investigate the impact of labelled training data on fine-tuning the DPR model for domain-specific retrieval, we conducted experiments using the FiQA dataset. We randomly sampled k examples from the FiQA training set as the training data and fine-tuned the DPR checkpoint after pre-training on MS MARCO. The results of the experiments are presented in Table 4.3.

For NDCG@10, we observed that the original DPR model, fine-tuned on MS MARCO, had slightly lower retrieval performance than the BM25 baseline, with a difference of 0.008. However, after fine-tuning with just 50 training samples, the model surpassed the BM25 baseline. With increasing training samples, the NDCG@10 score continued to improve, surpassing even the best zero-shot method on the FiQA dataset listed on the BEIR benchmark. And notably, fine-tuning with 1000 samples led to further performance gains. On the other hand, when evaluating Recall@100, we observed a different trend. The fine-tuned models exhibited lower recall scores compared to the zero-shot version. This suggests that the fine-tuning process caused overfitting due to limited training examples.

In additional experiments not included in the table, we found that using a lower learning rate can help mitigate overfitting when working with smaller training sets. Additionally, we observed that larger batch sizes generally resulted in better performance.

4.3 Parametric Adaptation Methods

4.3.1 Generation as Augmentation

One approach in domain transfer for RBQA systems involves generating synthesized data [149]. The process typically begins by training a question generation model in a general domain and then performing inference on the target domain corpus to obtain labelled question-paragraph-answer triplets as augmented data. Notably, Thakur et al. [233] demonstrate a GenQ baseline for various datasets, showing promising performance.

However, employing such methods faces challenges, particularly in evaluating the quality of the generated questions in terms of their contribution to the training of question-answering models. Additionally, a key difference exists when generalizing the question generation method for readers. In this case, we want the question generation model to focus on generating questions aligned with the desired answers, thereby ensuring the correctness of the supervision signal. In our exploration of domain transfer for RBQA systems, we investigated two key directions:

- **Paraphrase Question Generation:** In this approach, we focus on generating paraphrase questions for the general domain dataset. By diversifying the question language formats, we aim to enhance the reader model's ability to handle a broader range of question variations. This helps improve the generalization capability of the model across different domains.
- **In-Domain Question Generation:** The goal is to augment the in-domain supervision signals by generating additional labelled question-paragraph-answer triplets. By incorporating domain-specific questions into the training process, we aim to enhance the model's performance and adapt it more effectively to the target domains.

Paraphrase Question Generation

In our experiment, we set out to investigate the impact of question paraphrases on training question-answering models. To achieve this, we employed the T5 model and utilized the QQP

Original Train(88k)	Augmented data	EM	F1	Epoch
100%	0k	80.08	87.62	2
100%	190k	79.78	87.59	1
100%	60k	79.75	87.40	1
10%	0k	68.33	78.96	1
10%	27k	67.91	77.91	1
10%	27k	67.67	77.87	3
1%	0k	54.75	64.71	20
1%	0k	53.17	63.61	3
1%	0k	32.56	43.68	1
1%	2.7k	55.30	66.01	20
1%	2.7k	51.04	61.90	1

Table 4.4: Paraphrase question generation augmentation results on SQuAD.

(Quora Question Pairs) dataset as the training data for question paraphrasing. Then, we generated paraphrases for the questions in the SQuAD training set. For each paraphrased question, we assigned the original paragraph and corresponding answer to create a question-paragraph-answer triplet.

We mixed the augmented dataset, consisting of the paraphrase-augmented questions, with the original SQuAD training set during the training process. This approach allowed the model to learn from both the original and paraphrased questions, leveraging the additional training data to improve its question-answering performance potentially.

The results of our experiment are presented in Table 4.4. Surprisingly, we observed that incorporating the augmented questions into the training set had a negative impact on the question-answering performance. Across the majority of the training group experiments, the addition of the augmented dataset resulted in decreased performance. One potential explanation for this outcome is that the quality of the generated paraphrases may not have been sufficiently high to contribute positively to the model's learning.

However, a slight performance improvement was observed in the 1% training group, where the augmented dataset constituted a relatively larger proportion of the training data. This suggests that when the original training dataset is relatively small, augmented questions can provide some enhancement to the supervision signal and contribute to improved performance.

In-Domain Question Generation

In this experiment, we investigated the potential benefits of incorporating generated in-domain questions to improve question-answering performance. We employed the T5 model as our question generation model and trained it on the Natural Question (NQ) and SQuAD datasets. We explored several variants of prompt formats to guide the question-generation process. The formats we tested were as follows:

- V1 "**Context:** context **Answer:** answer **Question:** question":
  In this format, we directly connected the three parts of the prompt using the respective keywords. However, with this version, the generated questions sometimes did not effectively target the desired answer.

- V2 "**Answer:** answer **Context:** context **Question:** question":
  In this format, we placed the answer at the beginning of the prompt to highlight its importance. Typically, important information in natural language is often conveyed at the beginning of a sentence or text.

- V3 "context with answer highlighted with <a> and **Question:** question":
  In this format, we marked the answers within the context by enclosing them with special tags (<a> and). This approach aimed to help the model better understand the task by providing explicit context about the answers.

Based on these prompt formats, we trained the question-answering model using the NQ baseline. However, we later discovered that this general domain question-answering dataset hurt the performance of our model when applied to the BioASQ dataset, possibly due to the difference in question format distribution. Despite this limitation, we believe that the relative comparisons between the different prompt formats can still provide valuable insights and contribute to drawing reasonable conclusions.

Based on the results presented in Table 4.5, we have made several key observations:

- The highlight version (V3) of the prompt format tends to have the most significant impact on the Exact Match performance metric. This suggests that marking the answers within the context using special tags helps the model produce more accurate and precise questions that directly target the desired answer.

- Despite various augmentation experiments, we did not observe any improvements in the F1 score. This indicates that the generated in-domain questions did not enhance the model's ability to capture relevant information beyond what was already achieved by the T5 model.

	Train Data	Epoch	EM	F1	QG BLEU
Baseline	-	2	67.48	76.94	-
	NQ	2	64.42	76.14	-
	Bioasq	2	68.10	75.93	-
Augment	V2	2	65.64	74.30	**8.66**
	V2	5	65.64	**74.44**	**8.66**
	V3	2	**66.26**	74.34	5.99

Table 4.5: Question generation results on BioASQ, with T5-base.

- It is worth noting that the T5 model already performs well, and the augmentation with additional questions only resulted in marginal improvements.

Furthermore, we encountered a significant challenge in evaluating the Question Generation (QG) model in this experiment. We found that the BLUE score, commonly used to evaluate the generated text's quality, did not correlate strongly with the performance of the fine-tuned reader model. This highlights the difficulty in assessing the quality and relevance of generated questions, specifically in the context of the question-answering task. Despite these challenges, our findings provide insights into the limited impact of question generation augmentation in the retrieval-based information access system. The results suggest that while certain prompt formats can improve performance on specific metrics, the overall benefits may be modest, and evaluating the QG model remains an ongoing challenge.

4.3.2 Domain Further Pre-training

In Section 4.1, we discussed the significant difference between specific domains and the general domain, particularly regarding vocabulary distribution. To address this data distribution gap, researchers have explored the effectiveness of further pre-training techniques, which have consistently shown performance improvements [226].

Our experiments also investigated the effectiveness of these pre-training methods in specific domains. We explored combining domain pre-trained models with fine-tuning downstream tasks in the general domain. We started with a pre-trained model that had already undergone fine-tuning in the general domain datasets. Subsequently, we conducted further pre-training (using the Masked Language Modeling objective for BERT) on the specific domain corpus. This additional pre-training aimed to better align the model's understanding with the vocabulary and characteristics of the target domain.

# epoch	BM25	DRP	1	2	3	4	5	6	7
NDCG@10	0.236	0.228	0.241	-	0.248	0.254	0.257	0.263	**0.265**
R@100	0.539	0.342	0.362	-	0.392	0.388	0.380	**0.394**	0.389
perplexity	-	8.573	8.312	-	7.388	7.218	**7.054**	7.118	6.940

Table 4.6: Effects of further pre-training to retrieval performance on FiQA.

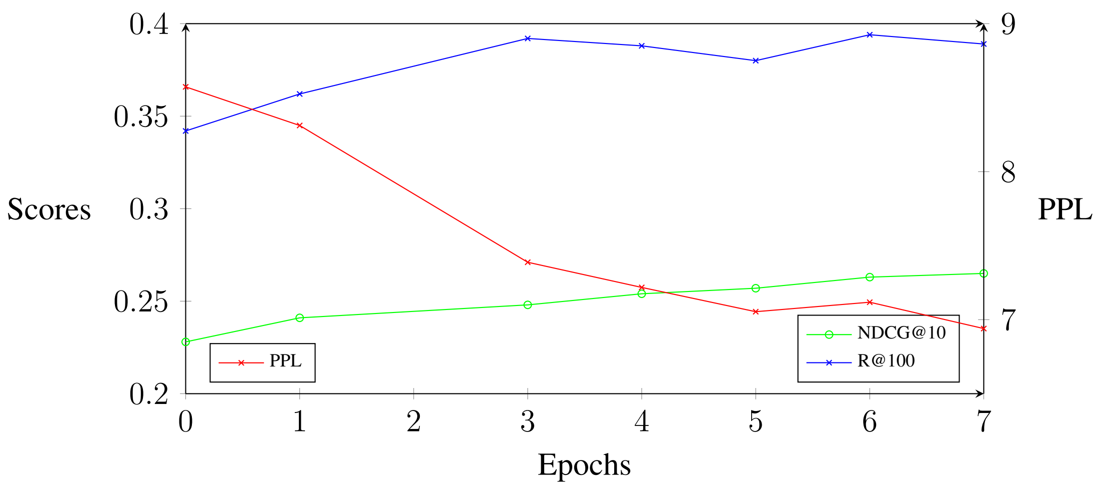


Figure 4.2: Effects of further pre-training, the retrieval performance based on further pre-trained model, on FiQA.

Once further pre-training was completed, we fine-tuned the model on general domain tasks for a few epochs. This fine-tuning step allowed the model to retain its task-specific capabilities while benefiting from the domain-specific knowledge acquired during further pre-training.

By combining domain pre-training with general domain fine-tuning, we aimed to leverage the advantages of both approaches and achieve improved performance on specific domain tasks. The following sections will present the results and insights gained from these experiments.

The results of our experiments are presented inTable 4.6, and we also provide a visual representation of the findings in Figure 4.2. The results generally indicate that further pre-training can improve retrieval performance on the specific domain task. It can be observed that there is not a strong correlation between the NDCG@10 scores and the perplexity of the further pre-trained model. However, lower perplexity values may have a negative impact on the Recall@100 metric.

Similarly to the challenges discussed in Section 4.3, one of the difficulties encountered in this method is the need for a reliable indicator for evaluating the performance of the further pre-trained model on the target downstream task. Despite the challenge of evaluating the performance of the further pre-trained model, our experiments suggest that optimizing for lower perplexity may not necessarily translate to better performance in the downstream task. Further investigation and analysis are needed to understand better the relationship between pre-training objectives, perplexity, and task-specific performance.

4.3.3 Progressive Domain-Clustered Training

We explore a clustering and scheduling approach to fully leverage the wealth of labelled datasets available in the general domain. The goal is to identify training examples more closely related to the target domain and schedule their training in a remote-to-close order. The MS MARCO dataset offers various topics, including bio-medical, finance, and computer science. By clustering the passages and queries in the embedding space, we can prioritize training on samples more similar to the target domain, thereby improving the performance on the specific domain task.

To measure sentence similarity, we utilize SentenceBERT [202], which provides sentence embeddings. Initially, we attempted to use the averaged corpus embeddings but found that this approach led to poor clustering performance. Instead, we conducted preliminary experiments using individual queries or document similarity. It is important to note that this should not be considered zero-shot learning, as we utilize in-domain training samples as references.

To implement the clustering-based scheduling approach, we divided the training set into three portions:

- Initially, the model is trained on the 80% to 95% least relevant examples for one epoch to familiarize it with the retrieval task.
- Subsequently, the model undergoes training on the 5% to 20% relatively similar examples for multiple epochs, inducing a domain shift.
- Finally, the top 1% most domain-relevant samples are utilized for extensive training conducted over the maximum number of epochs.

By progressively training the model, starting with the least similar samples and gradually incorporating the most similar ones, we aim to improve the model's ability to handle the target domain effectively.

The results presented in Table 4.7 indicate that the cluster-based training method does not significantly enhance the performance of the DPR model on the FiQA dataset. Despite the

Model	Slice Score	Epoch Schedule	NDCG@10	R@100
FiQA				
BM25	-	-	0.236	0.539
DPR	-	-	0.233	0.301
	0.03, 0.07	1,3,20	0.191	0.248
	0.03, 0.07	1,5,20	**0.237**	0.303
	0.03, 0.08	1,5,10	0.235	0.303
	0.03, 0.08	1,3,20	0.235	**0.304**
SciFact [251]				
BM25	-	-	0.665	0.908
DPR	-	-	0.50	0.48
DPR Supervised	-	-	0.747	0.732
	0.03, 0.1	1, 5, 20	**0.53**	**0.51**
	0.03, 0.1	1, 5, 40	0.52	0.50
	0.03, 0.1	1, 3, 10	0.52	0.50
	0.02, 0.08	1, 3, 10	0.51	0.49
	0.03, 0.1	1, 3, 20	**0.53**	**0.51**
	0.05, 0.15	1, 3, 20	0.52	0.50

Table 4.7: Results of progressive domain-clustered training on FiQA and SciFact

progressive training approach is based on domain relevance, the improvements in retrieval performance are marginal. Further analysis and investigation are required to understand the reasons behind the limited impact of this method on the model's performance.

4.4 Non-Parametric Adaptation: GPT-based Compressor

In this section, we delve into another line of research that tackles an even more challenging learning scenario: human-like few-shot learning. Large labeled datasets are often scarce in the real world where annotation is expensive and time consuming. This has prompted the development of few-shot learning, where the model is learned using only a few annotated samples [49]. One resort to the few-shot scenario is to utilize the pre-trained models like Generative Pre-trained Transformers (GPTs) [197, 198, 16, 181] with in-context learning [16, 302], fine-tuning [142] or the combination [8]. However, in-context learning requires heavy engineering to achieve a high accuracy [144], and its ability to generalize to different tasks is constrained by the input size and the need for precise formatting. Fine-tuning also has limitations, notably its inability to generalize to rare out-of-distribution datasets with limited labeled samples [291, 178].

Contrary to the data-hungry nature of machine learning, humans demonstrate exceptional ability in zero-shot and few-shot learning, where only a handful of labeled examples are needed to grasp a concept. Inspired by this, [86] proposed a human-like few-shot learning framework and boils it down to the ability of compressing data at inference time, measured by the universal information distance. Derived from a simple and accepted assumption in thermodynamics [9], the universal information distance emerges as the key component in taking the effective usage of few labeled data. Hence, accurate approximations of the information distance can lead to improved learning, though its incomputability has posed significant challenges. This information distance, consisting of Kolmogorov complexity [118], is both data-type-agnostic and distribution-agnostic. Additionally, its parameter-free usage enables the metric's applicability across diverse applications. Inspired by this theory, we propose a novel method GPT-AC, that leverages the knowledge GPTs learned during pre-training. Our method tackles tasks traditionally challenging for prompt engineering or fine-tuning, including semantic similarity, zero-shot text classification and ranking.

Kolmogorov complexity, also known as the length of the shortest computer program for a target output, is often approximated by the compression length. At the same time, entropy coding algorithms attempt to approach the code length lower bound declared by Shannon's source coding theorem: the entropy of the source. By using rich priors, such as pre-trained language models, that more accurately predict the source distribution, we can optimize the compression ratio and more closely approximate Kolmogorov complexity. Despite the potential of large language model-based compressors, their direct application to downstream tasks is nearly infeasible due to speed constraints. In addition to the inference speed required by the language model itself, the overhead of the compressor is even more substantial. Fortunately, the information distance only requires the compressed length instead of actual compression of the text sequence. We demonstrate an equivalence of the compression length under arithmetic coding to the negative log probability of text tokens when using GPT as the prior. This easy-to-compute compression length enables us to efficiently approximate the universal information distance without the overheads of the actual compression. By approximating normalized information distances [27, 119, 120] using GPT-based compression, we significantly enhance GPT's ability to quantitatively capture text similarities, which forms the foundation for its application in downstream NLP tasks. We show an example application in domain-specific text re-ranking in this thesis.

4.4.1 Related Work

Few-shot Learning

Prior to the emergence of large pre-trained models, the majority of previous works on few-shot learning can be divided into two streams: meta/transfer-learning based methods [246, 46, 223, 229] and data augmentation based methods [104, 191, 44, 56]. However, the former relies on constraining the hypothesis space by prior knowledge from other tasks or support datasets while the latter depends on the heuristics of the datasets, often accompanied with the smoothness assumption [243] (i.e., close data points in the input space share the same label). Pre-trained models, on the other hand, have incorporated prior knowledge during the pre-training stage and are proved to be excellent few-shot learners [16]. However, pre-trained models suffer from (1) high computational cost and (2) unsatisfactory performance in out-of-distributed datasets [291]. The problem of computational cost is especially prominent for large language models like GPT-3 where it is infeasible to fine-tune locally. We utilize pre-trained language model for one-shot and zero-shot classification tasks with no fine-tuning required.

Kolmogorov Complexity and Compression Distance

Information distance was first proposed by [9] as a universal similarity metric between two objects. It was proved to be universal in a way that it is optimal and can discover every possible metric [9]. Due to the problem of incomputability, [23, 120, 27] have derived computable version of information distances for tasks like clustering and plagiarism detection, shedding light on the possibility of using real-world compressors to approximate Kolmogorov complexity. These prior works utilize traditional compressors, indicating that the performance gain on downstream tasks mainly comes from the compressor-based distance metric.

Recently, [85] propose non-parametric learning by compression with latent variables (NPC-LV) where neural networks are incorporated into information distance. They demonstrate that trained generative models like variational autoencoders can be used directly with zero parameters for downstream few-shot image classification. Also, [87] employ this method in text classification using GZIP, achieving competitive results compared to several widely-used deep learning approaches. However, it remains open in how to incorporate pre-trained language models into this framework, which we aim to address in this paper. A recent study [270] explores the use of compression length for in-context example selection. They rely on a large candidate set and applied the model under generation setting. In contrast, we focus on adapting generative models to zero/one-shot learning for text similarity tasks.

Neural Compression

Our GPT-based compressor falls under the category of neural compression where neural networks are used for data compression. Shannon's source coding theorem [219] establishes the lower bound of the lossless compression on random variables with probability distribution. With near-optimal coding schemes, the bottleneck is the entropy approximation. Fortunately, deep generative models with explicit density estimation serve as the entropy model that can learn adaptively. [237] propose Bits-Back with Asymmetric Numeral Systems (BB-ANS), a lossless compression algorithm based on VAE. Bit-Swap [98] further improves BB-ANS by incorporating multiple latent variables and hierarchical networks. In addition to autoencoders, Flow [206, 257]-based lossless compression [78] outperform Bit-Swap and achieve the state of the art compression ratio of images. The development of deep neural networks also benefits lossless text compression. [65] use recurrent neural networks [215] combining with arithmetic coding [266] to achieve higher compression ratio than GZIP. Recent advancements, such as the fast transformer-based general-purpose lossless compressor TRACE [163], have demonstrated promising results in enhancing compression performance with transformer architecture. NNCP [7] v3.1, adaptively encodes the source message with Transformers, achieves state-of-the-art performance on the Matt Mahoney's Large Text Compression Benchmark[5].

4.4.2 Method

Human-Like Few-Shot Learning

Formally: we consider the following generalized human-like few-shot learning setting. Assume a universe of objects Ω comprising various concepts. Given an unlabelled subset $U \subset \Omega$ (or a pre-training corpus), a hypothesis ϕ is formulated. For any concept c, we have information

[5] http://mattmahoney.net/dc/text.html

D_c representing a description or representations derived from a few examples. The goal is to determine the concept for any new instance $x \in \Omega$ under a computable metric $\mathcal{M}$:

$$argmin_{c \in \mathcal{C}} \mathcal{M}(x, D_c | \phi(U)). \tag{4.1}$$

Here, ϕ can be a pre-trained model learned distribution from the unlabeled data U. For instance, GPT can be seen as the hypothesis ϕ learned from a large unlabeled corpus. Moreover, we can consider $\mathcal{C}$ as the selection space during inference. For example, if we are doing a classification task, then $\mathcal{C}$ is the finite set of all possible classes; if we are doing text re-ranking, then $\mathcal{C}$ is an infinite set of all selected candidates to be ranked.

This learning scenario differs from traditional machine learning settings as it permits extensive pre-training using unlabeled data but provides very limited labelled data for learning concept classes. Under this framework, the optimal metric $\mathcal{M}$ for few-shot learning algorithms is proven to be the universal information distance, defined by Kolmogorov complexity, based on the von-Neuman-Landauer Principle. Formally, the universal information distance $\mathcal{M}_{UID}$ between two objects x and y is defined as:

$$\mathcal{M}_{UID}(x, y) = \max\{K(x|y), K(y|x)\}, \tag{4.2}$$

$K(x|y)$ is the Kolmogorov complexity of x given y, or informally, the length of the shortest program that outputs x given input y. Since the Kolmogorov complexity is not computable [314], it is often approximated via compression in practice.

GPT-based Compression

We proposed GPT-based Arithmetic Coding (GPT-AC), where GPT is integrated into adaptive arithmetic coding, an entropy-based compressor. Consider a text $T = (t_1, t_2, ..., t_n)$, composed of a sequence of tokens. Let ϕ represent a GPT model, where $\phi(t_{1:(i-1)}) = P_i(t_i | t_1, t_2, ..., t_{i-1})$ models the probability distribution P_i of the next token t_i. The function $\phi(T)$ outputs all next-token probability distributions $(P_2, \cdots, P_{n+1})$. To derive the distribution for P_1, an EOS (End Of Sentence) token is added at the start of the text as t_0. For each token t_i, the associated P_i is the entropy model for encoding and decoding in the compressor.

In practice, since we cannot have float number with infinite precision. We always output the left several bits and shift the current interval by same bits once we reach the float number overflow limit.

GPT-based Arithmetic Encoding: In the encoding phase, under the schema of adaptive arithmetic coding, we start with an initial interval $I^0 = [0, 1)$. For each token t_i in the sequence,

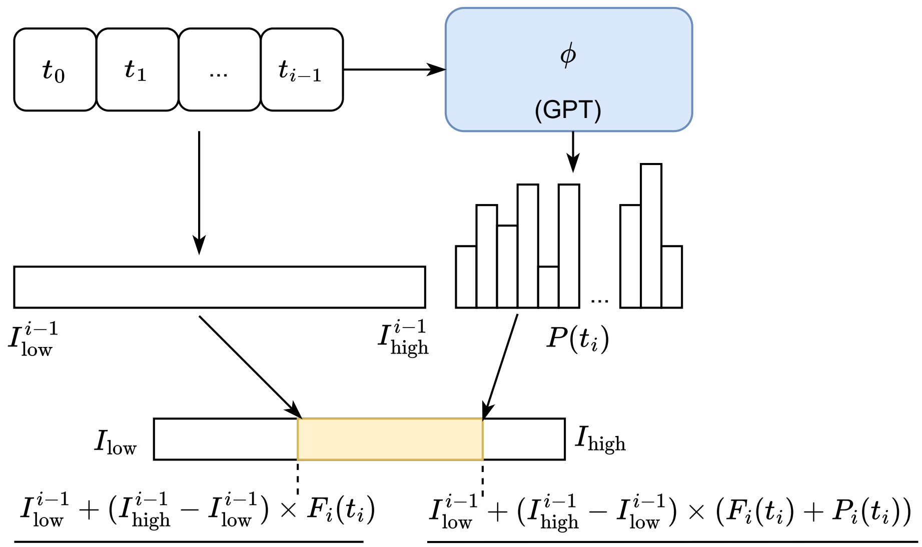


Figure 4.3: Illustration of GPT-based arithmetic encoding

Algorithm 1 GPT-AC Encoding

Input: $T = \{t_0, t_1, t_2, ..., t_n\}$ where t_0 is the EOS token, GPT model ϕ.
Initialize range $I_{low} = 0, I_{high} = 1$
for $t_i \in T, i = 1, \cdots, (n+1)$ **do**
  Obtain $F_i(t_i)$ and $P_i(t_i)$ based on $\phi(t_{1:(i-1)})$
  $I_{low-previous} \leftarrow I_{low}$
  $I_{high-previous} \leftarrow I_{high}$
  $I_{low} \leftarrow I_{low-previous} + (I_{high-previous} - I_{low-previous}) * F_i(t_i)$
  $I_{high} \leftarrow I_{low-previous} + (I_{high-previous} - I_{low-previous}) * (F_i(t_i) + P_i(t_i))$
end for
Output a single number $I_{low} \leq r < I_{high}$ as the encoded text.

Algorithm 2 GPT-AC Decoding

Input: encoded message r, GPT model ϕ.
Initialize decoded text $D = \{t_0\}$ where t_0 is an EOS token. $i = 0$
while $i = 0$ or t_i is not EOS token **do**
    Calculate $P_{i+1} = \phi(D)$
    Find index j such that $F_{i+1}(v_j) \leq r < F_{i+1}(v_j) + P_{i+1}(v_j)$ where v_i represents the i-th token in GPT vocabulary
    Update $t_{i+1} \leftarrow v_j$
    Update $D = D + \{t_{i+1}\}$
    Update $r \leftarrow \frac{r - F_{i+1}(v_j)}{P_{i+1}(v_j)}$
    Update $i \leftarrow i + 1$
end while
Output D as the decoded message

we calculate the cumulative distribution function $F_i(t_i)$ and $P_i(t_i)$ based on $\phi(t_{1:(i-1)})$, where $F_i(t_i) = \sum_{w|w \leq t_i} P_i(w)$ and $w < t_i$ means v appears before t_i in the vocabulary. Then, the interval $I = [I_{\text{low}}, I_{\text{high}})$ is updated according to the range assigned to t_i:

$$\begin{aligned} I^i_{\text{low}} &= I^{i-1}_{\text{low}} + (I^{i-1}_{\text{high}} - I^{i-1}_{\text{low}}) \times F_i(t_i), \\ I^i_{\text{high}} &= I^{i-1}_{\text{low}} + (I^{i-1}_{\text{high}} - I^{i-1}_{\text{low}}) \times (F_i(t_i) + P_i(t_i)) \end{aligned} \tag{4.3}$$

After updating I for each token in the sequence, we can pick any number, say x, within the final interval to represent the entire text sequence. See detailed encoding algorithm in Algorithm 1.

GPT-based Arithmetic Decoding: When decoding the message x, the token t_1 can be identified by finding the range $[F_1(t_1), F_1(t_1) + P_1(t_1)]$ that includes x. The value of x is then updated by normalizing it within the range of t_1, using the formula: $x \leftarrow \frac{x - F_1(t_1)}{P_1(t_1)}$. With this updated x and the next-token probability distribution $\phi(t_2)$, we can decode the next token t_2. This process is repeated until an EOS token is encountered, indicating the end of the text. The text can be losslessly decoded using x and ϕ alone. See detailed decoding algorithm in Algorithm 2.

Negative Log Probability as Compression Length: During the arithmetic encoding, the length of the interval I^i equals $I^{i-1} * P_i(t_i)$. From an initial interval of length 1, the entire message's encoding results in a final interval with a length of $\prod_{i=1}^{n} P_i(t_i)$. The number of bits required to represent this final interval, and thus the message T, is $\sum_{i=1}^{n} - \log_2 P_i(t_i)$. Interestingly, the optimization target during pre-training for auto-regressive models such as GPT is defined as:

$$L(T|p_{model}) = -\log p_{model}(T) = -\log p_{model}(t_1, t_2, ..., t_n) = \sum_{i=1}^{n} - \log_2 P_i(t_i).$$

Consequently, the pre-training phase of GPT models is essentially an exercise in learning a compressor that optimizes the coding length.

The Rank Coding Variant: In the abovementioned method, we primarily employ arithmetic coding for text compression. An alternative variant of lossless coding uses ranks instead of probability [73]. After the GPT model predicts the distribution of the next token, we can rank all tokens according to their probability. The target token is assigned the corresponding rank index, with tokens of higher probability having a smaller rank index. In this variant, we approximate the compression length as $\sum_{i=1}^{n} \log_2(rank_i)$ where $rank_i$ denotes the rank for token i.

Information Distance Approximation

Having computed the compression length using the GPT-AC method, we can now utilize it to approximate the information distance. Let $x = \{x_1, \cdots, x_n\}$ and $y = \{y_1, \cdots, y_m\}$ denote two tokenized text sequences, where each x_i or y_i represents a token in the sequence. We approximate $K(x)$ using the compression length $C(x) = \sum_{i=1}^{n} -\log_2 P_i(x_i)$ where P_i represents the probability distribution for x_i based on the GPT model prediction. As in Equation (4.2), we also need $K(x|y)$, approximated as follows: let $P_i = \phi(y, x_{1:i-1})$ denotes the probability distribution for token x_i outputed by the GPT model, given $y = (y_1, \cdots, y_m)$ and previous tokens in x, $K(x|y)$ can be estimated as $C(x|y) = \sum_{i=1}^{n} -\log_2 P_i(x_i)$. We denote all compressed-based approximations in $C(\cdot)$.

However, compression lengths vary when the lengths of the input text differ. We need a normalized version to enable comparison across diverse object pairs. There are several normalized measures:

- **The max distance, NID:** Li et al. [120] introduced a normalized version, referred to as the Normalized Information Distance (NID):

$$\mathcal{M}_{max}(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \tag{4.4}$$

- **The min distance:** To tackle challenges such as partial matching, situations where only portions of two objects match each other, Li [117] proposed the following variants of the universal distances for broader application scenarios:

$$\mathcal{M}_{min}(x, y) = \frac{\min\{K(x|y), K(y|x)\}}{\min\{K(x), K(y)\}}. \tag{4.5}$$

- **The mean distance, CDM:** Keogh et al. [93] proposed the Compression-Based Dissimilarity Measure (CDM) for data mining applications, demonstrating its effectiveness in practice. We re-scale it to fit within the range $[0, 1)$:

$$\mathcal{M}_{mean}(x, y) = \frac{C(x|y) + C(y|x)}{C(x) + C(y)} = 2 * CDM - 1, CDM = \frac{C(xy)}{C(x) + C(y)} \tag{4.6}$$

Compression-based distance for text re-ranking: To apply the above distances to text re-ranking, we first compute their compression lengths $C(x), C(y)$, where x is the document and y is the query. We also calculate the joint and conditional compression lengths $C(xy), C(x|y), C(y|x)$. Subsequently, we use these values to compute the distance metrics defined in Section 4.4.2 as $\mathcal{M}$.

4.4.3 Experiments

Our experimental evaluation consists of four key components: lossless text compression and three downstream tasks, namely semantic textual similarity, text classification, and text re-ranking. For downstream applications, we mainly conduct experiments with the GPT-2 small (124M), comparing to GPT-2 embedding or prompt tuning baselines and BERT-base-uncased (110M) models from HuggingFace Hub[6]. We take GPT-2 as an example due to its light weights and availability. However, the proposed method is not limited to GPT models. It can be readily applied to more advanced large language models, such as LLAMA[235, 236], provided that the output probabilities are available.

Lossless Text Compression

In the Lossless Text Compression task, we assess our method on the Enwik9 [222] and the first gigabyte of BookCorpus [311] datasets. In addition to GPT-2 models, we tested our method on LLAMA2-7B [235]. GPT-AC is benchmarked against both traditional methods, such as GZIP [40], and contemporary neural network-based techniques like DZIP [66] and TRACE [163]. In the actual implementation, GPT-AC processes chunks of 2,500 characters for GPT-2 and 10,000 characters for LLAMA2-7B independently. Although this would slightly compromise the compression ratio, it enables parallel computing.

As shown in Table 4.8, GPT-AC significantly outperforms conventional methods like GZIP and 7z in compression ratio. Even with the GPT-2 small model, GPT-AC achieves a more than 2-fold enhancement in compression ratio compared to the widely-used GZIP, on both the Enwik9

[6] https://huggingface.co/

Table 4.8: **Compression Ratio by Compression Method.** Note the compression ratio equals to *Original text length / Compressed text length*.

Model →	GPT-AC (Ours)				GZIP	7z	DZIP	TRACE	NNCP
Dataset ↓	Llama2-7B	GPT2-L	GPT2-M	GPT2-S					
Enwik9	**15.56**	8.05	7.71	6.53	3.09	4.35	4.47	5.29	9.33
BookCorpus	**10.55**	8.34	7.89	7.22	2.77	3.80	3.95	4.58	-

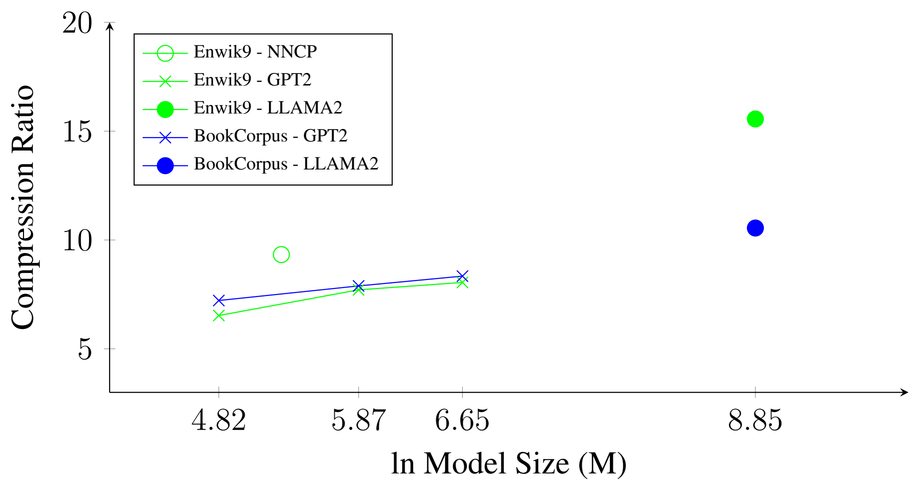


Figure 4.4: Relation between Compression Ratio and Model Size.

and Book datasets. On Enwik9, GPT-AC with Llama2-7B records a compression ratio of 15.56, a 67% enhancement over the previous state of the art of 9.33, based on NNCP [7] [7]. As the language model increases, the compression ratio consistently improves, suggesting that larger and better-trained language models will further amplify these results.

Note that NNCP involves updating the parameters of a large transformer model (v3.1, 199M) during encoding. We can also achieve an even higher compression ratio with encode-time optimization. However, the encode-time optimization process only enables an increase in compression ratio as the input message length goes up, and would overfit to the specific message. With random initialization, the compression ratio will be around 1.0 for the beginning of the input message, offering no benefits to similarity measurement.

[7] Refer to: http://mattmahoney.net/dc/text.html

Dataset ↓	# Test	Model → Domain	GPT-AC(Ours)	BM25	GZIP	(DPR)
TREC-COVID	50	COVID	**0.694**	0.656	0.447	(0.332)
TREC-NEWS	57	News	0.225	**0.398**	0.142	(0.161)
SciFact	300	Scientific	0.648	**0.665**	0.053	(0.318)
BioASQ	500	Bio-Med	**0.517**	0.465	0.157	(0.127)
FiQA-2018	648	Finance	**0.239**	0.236	0.032	(0.112)
ArguAna	1406	Argument	**0.327**	0.315	0.073	(0.175)

Table 4.9: GPT-AC text re-ranking dffectiveness (NDCG@10).

Text Re-ranking

We mainly conduct experiments with the GPT-2 small (124M) and BERT-base-uncased(110M) models from HuggingFace Hub[8]. We opted for this version of GPT-2 mainly due to the need for complete access to the token probability distribution in the APIs of newer versions. To show the effectiveness of zero-shot domain adaptation in re-ranking, we evaluate the models on different domain-specific zero-shot text retrieval datasets, including Trec-Covid [248], Trec-News [224], SciFact [252], BioASQ [239], FiQA-2018 [154], and ArguAna [250]. Given a query, we first retrieve the top relevant document with BM25 [209] with Elastic Search API [9]. We then re-rank the documents with the models. We compare our system with the original BM25 ranking and the Dense Passage Retrieval (DPR) [92] model, a BERT-based model already fine-tuned on the MS MARCO [173] for ranking, and a text GZIP compressor. We retrieved the top 100 relevant passages using BM25 and then re-ranking using GPT-AC.

As shown in Table 4.9, our proposed method outperforms the BERT-based DPR model across all settings. Our performance remains superior despite the DPR model being fine-tuned on the massive labelled MS MARCO dataset. We also benchmark our model against the GZIP compression method. The improvements observed indicate that GPT-AC can provide important semantic information, leading to improved ranking results. Notably, BM25 is a strong baseline in that the domain-specific texts can contain many out-of-distribution terms, potentially hampering the performance of the language model. Despite this, our method demonstrates comparable performance across most datasets and surpasses BM25 in certain domains, particularly in Bio-Medical and Finance, which requires much domain-specific understanding.

[8] https://huggingface.co/gpt2, https://huggingface.co/bert-base-uncased

[9] https://github.com/elastic/elasticsearch

			Log-Prob			Log-Rank		
Dataset	$len(x)$	$len(y)$	$\mathcal{M}_{max}$	$\mathcal{M}_{min}$	$\mathcal{M}_{mean}$	$\mathcal{M}_{max}$	$\mathcal{M}_{min}$	$\mathcal{M}_{mean}$
TREC-COVID	303	17	0.459	0.655	0.467	0.473	**0.694**	0.489
TREC-NEWS	808	16	0.173	0.161	0.167	0.184	0.161	**0.186**
BioASQ	304	14	0.306	**0.517**	0.364	0.267	0.507	0.330
FiQA-2018	247	15	0.128	**0.239**	0.154	0.118	0.222	0.143
ArguAna	276	247	0.277	0.257	0.301	0.282	0.262	**0.307**
SciFact	345	21	0.389	**0.648**	0.519	0.351	0.635	0.478

Table 4.10: Distance metric analysis.

Comparison of Distance Metrics

In Table 4.10, we compare the results of all distance metrics with two different coding variants. For better understanding, let's take a concrete example from FiQA in re-ranking.

- **Query:** *How to Deduct Family Health Care Premiums Under Side Business?*
- **Document:** ... *Health insurance premiums paid or reimbursed by the S corporation are shown as wages on Form W-2. The insurance plan must be established under your business. Your personal services must have been a material income-producing factor in the business. If you are filing Schedule C, C-EZ, or F, the policy can be either in your name or in the name of the business.*

As stated in Section 4.4.2, we always take the longer sequence results with $\mathcal{M}_{max}$, the shorter sequence results with $\mathcal{M}_{min}$, and the average over both of them with $\mathcal{M}_{mean}$. If we aim to avoid bias towards any single sequence when both texts contain similar or equally important information, suggesting that $\mathcal{M}_{mean}$ is the most suitable. Yet, this is different for re-ranking. Typically, queries contain fewer than 20 tokens, contrasting with documents that often contain hundreds. We thus want to focus the measurement on the query part without being disturbed by the extra or non-relevant information in the document. Therefore, $\mathcal{M}_{min}$ is the preferred choice. ArguAna, however, is a unique case as both the query and the document contain about 250 tokens with similar content, making $\mathcal{M}_{mean}$ more suitable.

Our experiments employ older versions of pre-trained language models due to computational constraints and limited access - for instance, the GPT-3+ API only provides the top 5 token probabilities. We surmise that improved base models would enhance our method since a superior

model	NDCG@10	R@100
BM25	0.236	0.539
DPR	0.228	0.342
Further Pre-training	0.263	0.394
Clustering	0.237	0.303
GPT-AC	0.239	-

Table 4.11: Summary of zero-shot transfer performance on FiQA.

prior could yield a better compressor and, subsequently, a more accurate distance measure. Our method only works for forward-only attention-based models to utilize arithmetic coding for lossless compression. However, it can generalize to all other models by utilizing a sliding window.

4.5 Discussion

Comparing across different adaptation methods we have discussed in this chapter (see Table 4.11), we find the most effective method is further pre-training. Based on this conclusion, we can further hypothesis that the model pre-trained on large in-domain corpus could boost the transfer learning performance a lot. We recommend starting from in domain pre-trained models, when they are are available, such as BioBERT [109], FinBERT [1], MathDPR [308] etc.

Upon comparing the various adaptation methods discussed in this chapter and reviewing the summary presented in Table 4.11, we observe that further pre-training emerges as the most effective approach for improving transfer learning performance. Building upon this finding, we hypothesize that utilizing pre-trained models specifically trained on large in-domain corpora could significantly enhance the transfer learning process. Consequently, we recommend leveraging in-domain pre-trained models, such as BioBERT [109], FinBERT [1], MathDPR [308], and others, when available, to maximize performance gains in specific domains.

Chapter 5

Generalization to Diverse Sources

An additional challenge when applying information access systems to specific domains arises from the presence of context that goes beyond plain text, as observed in math formula search and database search tasks. These domains introduce unique complexities. In database search, the data involved often consists of semi-structured information, which can include fields, tables, and relations. This type of data may have missing elements or incomplete information, adding an extra layer of complexity to the search task. For e-commerce applications, the labeling of data can be noisy, making it challenging to accurately interpret and retrieve relevant information. In math search, formulas are involved, which expands the vocabulary beyond plain text and requires additional logical reasoning.

5.1 Retrieval from Database

In this section, we focus on the dense retrieval model as the first-stage search module for e-commerce. A high-quality search, or retrieval, system is critical to e-commerce system efficiency and user experience. We build an embedding model to learn the semantic relationship between queries and products for Instacart. Both the product information and our learning signal come from the database, usually in a table format. We present a simple yet effective method that enables the application of recent advanced pre-trained language models to our application. We concatenate special tokens and the row element in the table as the input to the models. The trained embeddings also serve as features for the later stages or other downstream modules, but are out of our scope in this thesis.

Beyond formatting considerations, applying retrieval-based systems to e-commerce applications poses several other challenges. These challenges arise from the specific characteristics of e-commerce data and the unique requirements of the domain.

Challenge 1, Cold-start: New products or un-seen queries historically don't perform well due to the lack of user engagements. Furthermore, according to our user research, new users tend to send unseen queries because they are not familiar with the e-commerce search engine. It's critical to show high-quality results for these queries so that we can retain these new users. Additionally, retailers expect good ranking results for newly introduced products. We can manually boost some of the new products to meet business needs, but manual boosting doesn't scale at all.

Previous works either include historical features, such as clicks, cart adds (conversions), purchases in the model input, or highly rely on these data in offline evaluation. If a system takes historical features as inputs, it will favor the products that have higher engagement. Continuous training on a regular cadence using these historical features will make the cold-start problem even worse, entering a self-fulfilling circle. Similar behavior will occur if the offline evaluation highly relies on the user engagement log.

To address this challenge, we propose to use only content-based features such as query text itself and product information including product name, brand name, categories and attributes from our catalog. This way, the model will focus on semantic relationships rather than historical engagement relationships. We also propose to use permutations of product information as synthetic queries for a product, and include those in the training dataset. This helps the model generalize on cold-start products when given attribute related queries. For offline evaluation, instead of relying on user engagement log data, we propose to evaluate the system on a scaled human annotated dataset, which contains a set of hot and cold products.

Challenge 2, Noisy Data: We follow previous approaches [146, 125], and use the user engagement data to collect positive query-product examples. We treat converted [1] products for queries as positive training samples. However, for grocery search, user engagement data could be particularly noisy. People come to Instacart to shop for multiple items. When irrelevant items show up in the search results, people may add these items to their cart just because they happen to need them regardless of the original search intent. When we take these conversions as training examples, it results in false positive labels. Another source of noise is that users may have very different tolerances for search results: Some users may add items that don't fully satisfy their search intent, while others may be very strict due to allergies or religious reasons. Therefore, conversions can imply a wide range of relevance, ranging from somewhat relevant to strongly relevant.

[1]In our case, a user converts a product in a search query means the user adds the product presented in the search results for the query to cart.

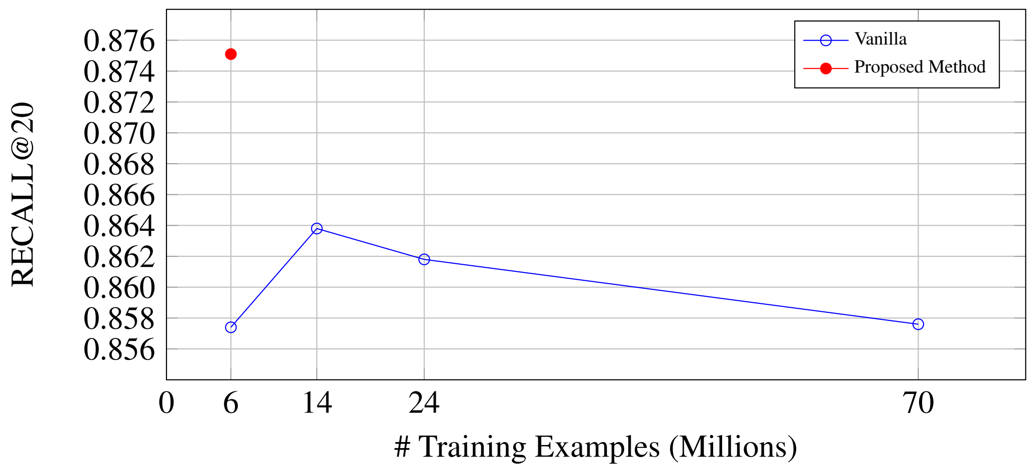


Figure 5.1: Effect of training dataset size for e-commerce search.

To show the effect of noise in the training set, we trained embedding based retrieval models with different sizes of training data. As the training set gets larger, we potentially add more relevant but cold examples at the expense of including more noisy examples. As shown in Figure 5.1, the performance of the model decreases after a point. This shows training with more data doesn't necessarily give better model performance due to the inclusion of noise.

To address the noisy data problem, we propose cascade training: We first warm-up the model with a noisy but large dataset. This way, we transfer the pretrained model's knowledge to the grocery domain by letting the model see as much data as possible. And then we train the model on a smaller yet high quality data. In this step, we best utilize learned knowledge to achieve better performance by training on the high quality data. We also show that cross architecture training, first training the two-tower shared encoder and then training with un-tied parameters, is useful in cascade training. Shared encoder allows the model to learn common grocery knowledge, and two towers with un-tied parameters allows the model to best fine-tune for each expertise (query and product knowledge).

We also propose a self-adversarial negative sampling/re-weighting method to select informative negative samples. Arguably, the straight-forward approach to generate negative samples is to draw them from a uniform distribution. Such uniform negative sampling is not efficient enough because many negative samples can be easily distinguished from the positive ones after a few training epochs. The proposed self-adversarial learning method adjusts the loss term weights for individual negative samples according to their scores in the current training step. This way, the model will be punished more, when it predicts high scores for the negative query product pairs, resulting in more efficient training. As shown in Figure 5.1, our methods utilize the knowledge learned from the vanilla training by performing cascade training with self-adversarial loss.

5.1.1 Related Work

Dense Retrieval Systems for E-commerce: Many embedding based retrieval approaches, such as Nigam et al. [174], Wu et al. [269] have been introduced to solve e-commerce search problems. Facebook systematically shared their knowledge in modeling, serving and full-stack optimization in Huang et al. [81], Liu et al. [146]. Taobao [125] and JD [298] shared their exploration in personalized product search systems. We also apply embedding based methods for product search, while we focus more on building a semantic relevance model.

Effective Negative Sampling Strategies: Training large models often relies on good optimization strategies, of which negative sample strategies are one of the most important for representation learning. Yih et al. [289] introduces the in-batch negative sampling, taking other samples within a mini-batch as negative samples. This method doesn't require explicitly labeled negative data and can reduce computation by re-using other in-batch sample representations. However, uniformly selecting negative samples is not the optimal choice. Wieting et al. [264] proposes self-adversarial training, which takes the in-batch negative with the highest score as a hard negative, thus, achieving improved results for paraphrastic sentence embeddings. Later, self-adversarial negative sampling is proven useful for relation-aware graph attention models [196] and knowledge graph (KG) embeddings [228]. In this paper, we introduce a self-adversarial negative sampling strategy to help improve the training efficiency with noisy data.

5.1.2 Method

Cascade Training

We collect conversions as positive training query-product samples. We gather one years worth of user engagement log, and only keep the query-product pairs that are converted by at least two users. As shown in Figure 5.1, including more pairs does not necessarily improve the model performance due to data noise. To filter out samples that are likely noise, we rank the products by the conversion rate in descending order and remove ones below a threshold. On one hand, we need to provide the model with more accurate relevance signals, so that it can learn a better relevance representation. On the other hand, we also need to train the model on as many unique queries or products as possible to transfer the pre-trained model's ability to our domain.

To meet both requirements, we propose cascade training: Starting from a pre-trained model, we first train the model on a warm-up dataset to transfer the model to the grocery domain. Then we

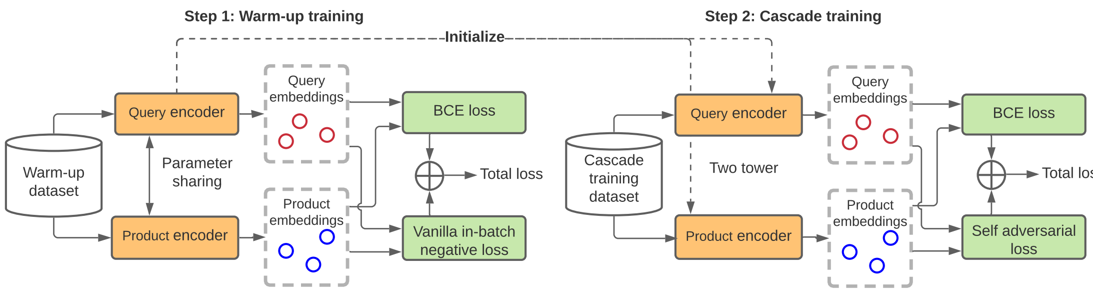


Figure 5.2: Cascade training overview.

train it on a cascade training dataset to achieve better performance. The entire training procedure is summarized in Figure 5.2.

- **The warm-up training step:** We first collect a relatively large *warm-up dataset* with lower conversion rate threshold. For a query q_i, we select top $k_{q_i}\%$[2] products as positive pairs. To balance the training examples, we apply smaller/larger $k_{q_i}\%$ s for queries with higher/lower frequencies in the log. We finally gathered 14 million unique query-product pairs for the warm-up dataset.
- **The cascade training step:** We then collect a smaller but less noisy *cascade training dataset*. We repeatedly include positive pairs with higher conversion rate threshold. We start with threshold $k^0_{q_i}\%$, which is much smaller than $k_{q_i}\%$ in the *warm-up dataset*, so that there are less noisy conversion signals. At step n, we add all the products ranked above top $k^n_{q_i}\%$ into the dataset. Then we decrease $k^n_{q_i} = k^{n-1}_{q_i} * \theta$ and repeatedly add sample pairs until there are no positive pairs. We finally gathered 6 million training samples for cascade training. Details of constructing the *cascade training dataset* can be found in Algorithm 3.

Augmentation with Catalog Data

Our users often combine different product keywords to create the queries. Suppose a user wants to by a product that has product name ”Organic 2% Reduced Fat Milk”, product brand name ”GreenWise”, category ”Milk”, and attributes ”organic, kosher, gluten free”, the user might send one of the following queries: *“GreenWise Milk”, “Organic”, “Organic 2% Reduced Fat Milk”,*

[2]Using k_{q_i} % = 100 % means we include all the conversions for query q_i.

Algorithm 3 Cascade Training Dataset Construction

$Q \leftarrow$ the query set
$C_{<q_i,p_j>} \leftarrow$ the number of conversions of $< q_i, p_j >$,
$P_{q_i} \leftarrow$ the converted products of q_i, sorted by $C_{<q_i,p_j>}$,
$n \leftarrow 0$, $D_{cascade} \leftarrow \emptyset$, $k_{q_i} \leftarrow k^0_{q_i}$, $\theta \leftarrow 0.5$,
repeat
  $D' \leftarrow \emptyset$
  for $q_i \in Q$ **do**
    D' append $\{< q_i, p_j > | p_j \in P_{q_i}, j \leq len(P_{q_i}) * k^n_{q_i}\%\}$,
    $k^{n+1}_{q_i} \leftarrow k^n_{q_i} * \theta$
  end for
  $n \leftarrow n + 1$
  $D_{cascade} \leftarrow D_{cascade} + D'$,
until $D' = \emptyset$
return $D_{cascade}$

"GreenWise". Our catalog contains these product meta data. We use hierarchical taxonomy, and map products into one of taxonomy nodes. So we can apply this heuristic to augment the training dataset to help address the cold start problem.

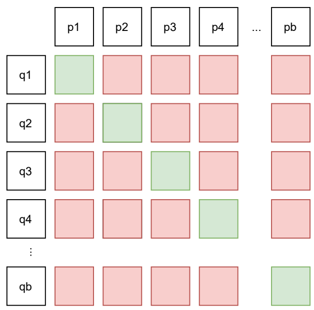
We synthesize positive queries using the combination of one or more product feature sources, including product name, product attributes, categories and brand names. We randomly sample synthetic data, and merge these with the dataset created from the user engagement data. Although the synthetic data can be clean, we keep the ratio of the synthetic data relatively low because it changes the actual distribution.

Features

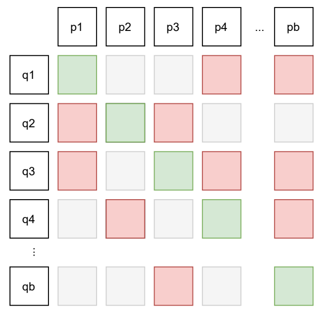
We use the query text as the input of the query encoder. For the product encoder, we input the concatenation of product name, brand name, size information, categories and attributes. In this way, the model can only capture semantic relationships and generalize to cold start products or queries. We use special tokens to explicitly indicate the start of different features. This provides a grammar rule so that a model can easily identify different features.

An example of the query and product features would be:

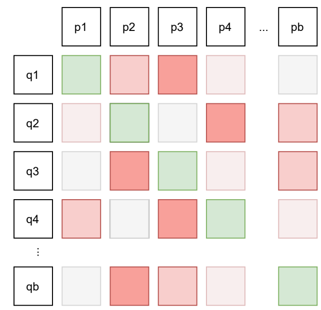
- **Query:** "*[QRY] milk*"



(a) Vanilla In-batch Negative.



(b) In-batch Negative with Random Sampling.



(c) In-batch Negative with Self-adversarial Re-weighting

Figure 5.3: Figure of different in-batch negative sampling.

- **Product:** "*[PN] Organic 2% Reduced Fat Milk [PBN] GreenWise [PCS] Milk [PAS] organic, kosher, gluten free*" respectively.

In the above examples, we use *[QRY]*, *[PN]*, *[PBN]*, *[PCS]* and *[PAS]* tokens to indicate the next token is a query, product name, product brand name, product categories and product attributes respectively.

Negative Sampling Strategies

In grocery search, many products can satisfy the same query intent but the users just favor some of the them. So, unlike in Huang et al. [81], we do not collect products that are presented in the search results but not converted as hard negatives. We only collect converted pairs and use in-batch negative samplings during training. We experimented on adding hard negative products (presented in search results but not converted). But this gives worse performance on our human evaluation dataset due to the bias we manually introduce.

Vanilla In-Batch Negatives: The vanilla in-batch negative strategy is to take all other in-batch products as negative samples, following [289]. Figure 5.3(a) demonstrates the samples we included in a mini-batch. For a training batch, we have $(q_1, q_2, ..., q_b)$ and $(p_1, p_2, ..., p_b)$, where b represents the batch size, and q_i and p_i are query and product embeddings for a positive training example i respectively. We take the dot products of the embeddings followed by a sigmoid activation $\sigma(\cdot)$ as the relevance score, and then we apply binary cross entropy loss $BCELoss(\cdot)$[3]

[3]We also experiment with other loss functions, such as Cosine similarity, Marginal loss, however, the BCE gives best performance.

to train the model. The objective is defined by: $L_{rel} = L_{pos} + \lambda_{neg} * L_{neg}$, where $L_{pos,neg}$ are binary cross entropy losses, and λ_{neg} is a hyper-parameter to adjust negative loss weight. We illustrate this in Figure 5.3(a), positive samples are colored in green and negative ones are colored in red.

Sampled In-Batch Negatives: For a certain query, instead of adding all negative loss terms into the final loss, we also try uniform sampling of negative products S_i Figure 5.3(b). Each negative example i has a sampling probability of $P(p_j \in S_i, i \neq j) = \frac{1}{b-1}$. The loss function becomes:

$$L_{rel} = L_{pos} + \lambda_{neg} * L_{sampled\_neg}, \tag{5.1}$$

$$L_{sampled\_neg} = \sum_{i} \sum_{p_j \in S_i} BCELoss(\sigma(q_i \cdot p_j), 0). \tag{5.2}$$

Self-Adversarial Negatives: Uniform negative sampling is not an optimal strategy: many negative samples can be easily distinguished after a few epochs of training. We hope the model can focus more on the difficulty samples. The difficulty of a certain negative sample to the current model can be measured by the model's prediction score. We adopt self-adversarial negative product sampling or re-weighting, shown in Figure 5.3(c), different levels of red represents the different weight applied to the samples. We take the prediction score of the current model for the negative sample as a weight term. We then apply either loss re-weighting or re-sampling based on the weight.

- **Self-Adversarial Negative Sample Re-weighting:** We first compute the dot product of the query and product embedding vectors and pass the score through an activation function $act()$ to get the relevance score. We multiply each sample's loss with its relevance score. Take the BCELoss version for example, the new loss function becomes:

$$L_{rel} = L_{pos} + \lambda_{neg} * L_{self\_adv\_reweight}, \tag{5.3}$$

$$L_{self\_adv\_reweight} = \sum_{i} \sum_{p_j \in S_i} w_{i,j} * BCELoss(\sigma(q_i \cdot p_j), 0), \tag{5.4}$$

$$w_{i,j} = act(q_i \cdot p_j). \tag{5.5}$$

  We experiment with various activation functions $act()$ including identity, sigmoid and ReLU functions. Identity activation function gives the best performance.

- **Self-Adversarial Negative Sampling:** Another way of applying the self-adversarial learning is to take the similarity score as the sampling probability:

$$L_{rel} = L_{pos} + \lambda_{neg} * L_{self\_adv\_sampling}, \tag{5.6}$$

$$L_{self\_adv\_sampling} = \sum_{i} \sum_{p_j \in S_i} BCELoss(\sigma(q_i \cdot p_j), 0), \tag{5.7}$$

$$P(p_j \in S_i) = \begin{cases} softmax(w_{i,j}/T) * (T^2), & i \neq j, \\ 0, & i = j, \end{cases} \tag{5.8}$$

where T is the self-adversarial sampling temperature and $w_{i,j}$ is the similarity score as in Equation (5.5).

5.1.3 Experiments

We base all our experiments on a pre-trained language model. We choose to use MiniLM-L3-v2[4] since it achieves relatively good performance and also provide the fastest speed, which can be beneficial for online inference.

We initialize both the query and the product encoders with the MiniLM-L3 model. As described in before, we train the model with two steps:

- **Warm-up training:** Initially, we train the model with shared parameters in both towers in the first warm-up training step. We train the model on the "warm-up dataset" with binary classification loss, using all other in-batch products as negative samples.
- **Cascade training:** In the second cascade training step, we continue training the two towers but with un-tied parameters. We train the model on the "cascade training dataset" with self-adversarial negative re-weighting training strategy.

We experiment with different pre-trained models including distilbert [5], MiniLM [6], TinyBERT [7]. We also experiment with their semantically fine-tuned checkpoints, for example the ones fine-tuned on sts-b or QQP in GLUE [253], or other NLI and paraphrase datasets.

Our implementation is based on the sentence transformer [202] [8]. The base model outputs a hidden vector for each input sentence. We project the hidden vector into a lower (100) dimension with dense layers to meet the memory requirements.

[4]https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L3-v2
[5]https://huggingface.co/distilbert-base-uncased
[6]https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L3-v2
[7]https://huggingface.co/cross-encoder/stsb-TinyBERT-L-4
[8]https://www.sbert.net/

Offline Evaluation Dataset

We create an offline evaluation dataset to measure the model performance so we don't have to launch online testing for every training trial. The user engagement log is not an ideal testing distribution because: first, it comes from the same source as the training data; second, it just reflects the conversion relationships instead of relevance representation ability; and third, when evaluating on the test data collected from the user engagement log, we get 0.999 NDCG@5 for most training trials.

Other than the previous proposed evaluation metric RECALL@K in Huang et al. [81], Liu et al. [146], we propose to also monitor NDCG@K on a human evaluation dataset as offline evaluation. We collect sampled search log to human annotators, query product pairs are labeled into 5 categories:

- **Strongly relevant**: The product is exactly the (type of) product the query is looking for.
- **Relevant**: The product is the type of the product the query is looking for, but there are likely others that fit better.
- **Somewhat relevant**: The product is not exactly what the query is looking for but I understand why it was shown.
- **Not relevant**: The product is not what the query is looking for, and I can't imagine why it was shown.
- **Offensive**: The product is unacceptable and creates a bad experience.

After filtering out records that raters do not achieve agreement on, meaning 3 out of 5 raters gives the same rate, we got 158k query product pairs. We take "strongly relevant" as a score of 3, "relevant" as 2, "somewhat relevant" as 1 and the other levels as zero. We then compute NDCG@K based on these scaled scores. Also, we take the levels with positive scores as positive labels and compute RECALL@K as other metrics.

Offline Evaluation Results

We show how the proposed method outperforms various baseline models in Table 5.1. We include BM25 [208] algorithm, Wide & Deep [25] model (our previous production model), and Que2Search [146] as baseline models. Wide & Deep uses historical conversion features and text matching scores as features for the wide component, and product raw text data and query text as

	NDCG @5	NDCG @10	NDCG @20	RECALL @5	RECALL @10	RECALL @20
BM25 [208]	0.5413	0.5785	0.6389	0.6365	0.6372	0.6371
Wide & Deep [25]	0.7609	0.7714	0.7936	0.8326	0.8112	0.7955
Que2Search [146]	**0.8158**	0.8242	0.8367	0.8938	0.8772	0.8650
Proposed	0.8150	**0.8246**	**0.8372**	**0.9013**	**0.8863**	**0.8751**

Table 5.1: Offline evaluation results.

	MRR	CAPS	GMV
Wide & Deep [25]	-	-	-
+ Vanilla Embedding	+0.7%	+2.0%	+0.5%
+ Proposed	+1.2%	+4.1%	+1.5%

Table 5.2: Online A/B testing results.

features for the deep component. The deep component is LSTM-based [76] network followed by 2 stage dense layers. For Que2Search [146], we tried symmetrical scaled cross entropy loss in the 1st stage and applied margin ranking loss in the 2nd stage training. We performed extensive hyper parameter search for fair comparison. We tried *scale* between 15 and 20 for the symmetrical cross entropy loss, and *margin* between 0.1 and 0.2 for the margin ranking loss as in Liu et al. [146]. We didn't observe improvement using margin ranking loss, thus we only report the best results obtained with the symmetrical scaled cross entropy loss.

BM25 performs worst compared to any other models which shows the effectiveness of representation learned from the deep learning models. Compared with the previous production model (Wide & Deep), the proposed method has shown significant improvement. Que2Search shows pretty good results, but our proposed method outperforms Que2Search especially for RECALL@K metrics that are critical when used in retrieval.

Online Evaluation Results

We deploy our system to production and conduct A/B testing to prove the effectiveness of the proposed methods. In Table 5.2 we show the relative improvement of the proposed methods compared with our baseline wide and deep model. Here, *MRR* means the Mean Reciprocal Rank of the first converted item, *CAPS* means the Cart Adds Per Search, and *GMV* means the Gross

	Data	NDCG			RECALL		
		@5	@10	@20	@5	@10	@20
Vanilla	Cas	0.8051	0.8160	0.8301	0.8855	0.8692	0.8574
+Data	Warm	0.8121	0.8215	0.8347	0.8915	0.8756	0.8638
+Self-Adv	Cas(Adv)	0.8151	0.8232	0.8362	0.8921	0.8748	0.8623
+Data +Self-Adv	Warm(Adv)	0.8074	0.8181	0.8319	0.8880	0.8724	0.8607
+Cascade	Warm(Adv)→Cas(Adv)	0.8102	0.8210	0.8342	0.8989	0.8845	0.8728
+Cascade*	Warm→Cas(Adv)	0.8139	0.8242	0.8367	0.9006	0.8853	0.8739
+Two Tower	Warm→Cas(Adv)	0.8150	**0.8246**	**0.8372**	**0.9013**	**0.8863**	**0.8751**

Table 5.3: Ablation study for embedding based search system

Merchandise Value. When we use the embedding model trained with vanilla in-batch negative samples (Vanilla Embedding in Table 5.2), we observe 0.7 % improvement in MRR, 2.0% gain in CAPS, and 0.5% of GMV increase. With all the proposed methods applied, we achieve 1.5 % improvement in MRR, 4.1% gain in CAPS, which turns into 1.2% of GMV increase.

Analysis

The Self-Adversarial Negative Learning Method Helps: Comparing row *Vanilla Embedding* and row *+Self-Adv*, we can see the proposed self-adversarial method can boost the model's performance, especially on ranking metrics (NDCG). Comparing row *Vanilla Embedding* and *+Data* in Table 5.1, we find vanilla all in-batch negative training can benefit from adding more data, by changing the training set from the the cascade training set (6M) to the warm-up set (14M). However, the model could not gain further benefit while we combine with a self-adversarial training strategy, comparing row *+Self-Adv* and *+Data&+Self-Adv*. This is because although the warm-up dataset is larger, it is also nosier, thus is not an ideal training ground truth.

Effect of Cascade Training: After adding the cascade training, row *+Cascade* outperforms previous rows with a large gap. Another interesting point is that row *+Cascade** outperforms row *+Cascade*. The second cascade training experiment first makes full use of all the in-batch negative samples at the warm-up stage and then learns more accurate information in the cascade training stage through self-adversarial negative re-weighting.

Effect of Cross Architecture Cascade Training: We also observe that two tower model architecture in the cascade training stage gives better performance than Siamese network architecture, see row *+Two Tower*. In the cascade training stage, further fine-tuning model parameters individually makes the model to efficiently learn customized information for each encoder.

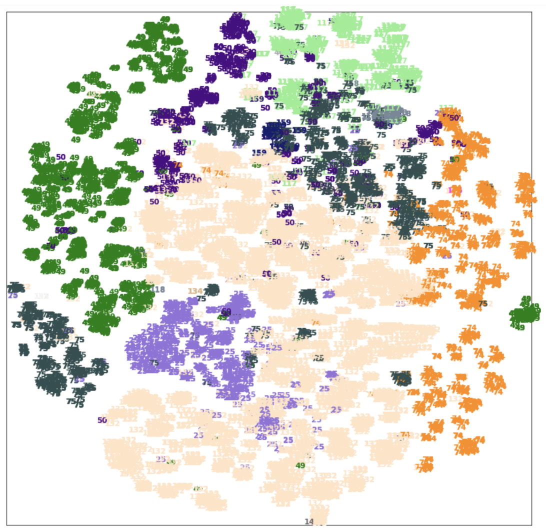


Figure 5.4: Embedding visualization.

Semantic Relevance: To analyze how the embedding model performs, we visualize the product embedding vectors. We randomly sampled product embeddings and colored them according to their pre-labeled categories. We apply T-SNE [242] to project the embedding vectors into a 2D space. Products are colored by different product categories. We sampled products from *food* and the light orange color represents the sub-category *pantry*. As shown in Figure 5.4, products with the same category labels are well clustered, and different clusters are well separated with a large margin.

We also plot the model predicted similarity score distribution on our human evaluation dataset in Figure 5.5. We color samples based on the human rated relevance levels. As can be seen, higher relevance groups are distributed to the right side, which means higher relevance score predictions. The non-relevant group is located at the most left area, with a much lower relevance score, separated from the relevant groups. This distribution shows the proposed method can help predict the semantic relevance very well.

Learned Semantic Signals Outperforms Historical Engagement Signals: Further more, we show a top query case to show how our model out perform previous model. We first show the learned representation performs well with an example query: "Milk". Users usually send this query for products under our "Plain Milk" category. We compare the scores under this category with another category "Milk Chocolate", which also contains the word "Milk" and might cause failure in keyword based systems. We collect the click through rates (CTRs) and the embedding

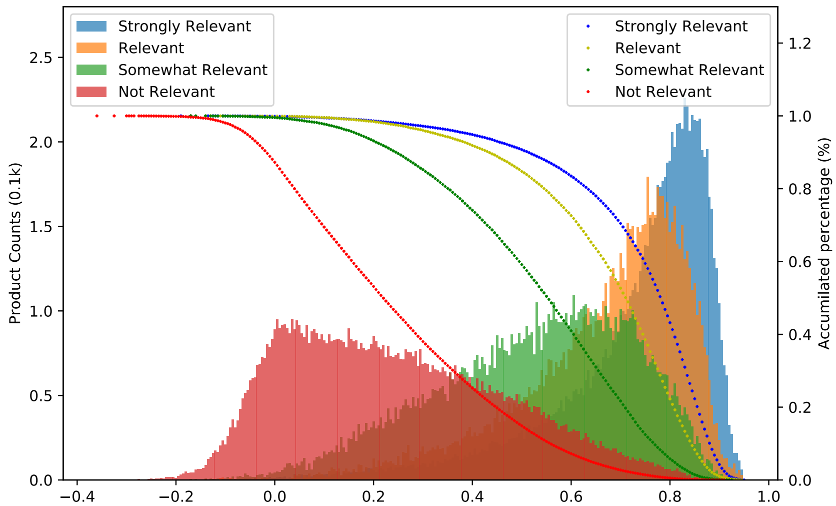
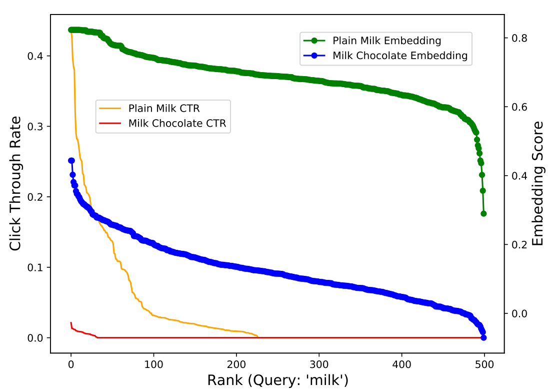
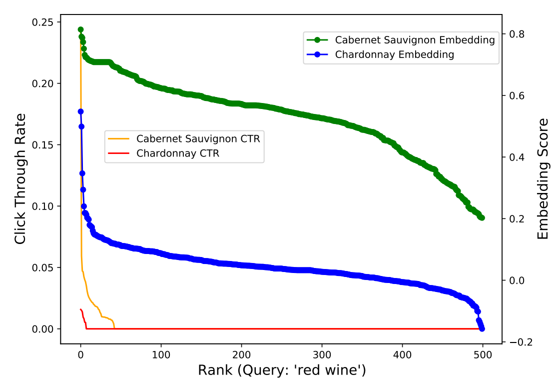


Figure 5.5: Score distribution over the human evaluation dataset.



(a) Query “milk”



(b) Query “red wine”.

Figure 5.6: Score distribution, comparing embedding scores with Click Through Rate(CTR) for different queries.

scores for randomly selected products under both categories given "milk" as a query. Figure 5.6(a) shows CTRs and embedding scores in descending order. From the figure, we observe that almost all the products in the "Plain Milk" category have higher embedding scores than "Milk Chocolate" products. That means embedding score can be used as a good indicator for differentiating between "Plain Milk" products and "Milk Chocolate" products. Half of the "Plain Milk" products have zero CTRs and some "Milk Chocolate" products have non-zero CTRs. If we rank items with CTRs, half of the relevant milk products get ranked lower than several milk chocolate products. That shows CTR is not a good search relevance indicator.

We then show how the proposed methods perform on the cold-start products with the example query "red wine". There are hundreds of different red wines in our product set, and very few of them are purchased through search. We compare similarity scores for products under "Cabernet Sauvignon" (one of red wine categories) and "Chardonnay" (one of white wine categories). We collect the CTRs and the embedding scores for randomly selected products under the two categories given "red wine" as a query. There are only about 10 % products under the "Cabernet Sauvignon" with positive CTRs, meaning all the other products are cold but relevant products. Figure 5.6(b) shows that embedding scores can be used for separating products in "Cabernet Sauvignon" from the products in "Chardonnay" while CTRs suffer from the cold start problem.

5.1.4 Deployment

We use the trained embedding model for retrieval and ranking. We utilize an approximate nearest neighbor search (FAISS [89]) for retrieval. We use similarity scores of the query and product embeddings as one of the features for our ranking model. Figure 5.7 demonstrates the overall system architecture.

Daily Offline Computation Pipeline: Once we train the embedding model, we pre-compute product embeddings, train the ANN indices offline daily. At Instacart, we have thousands of retailers and they don't share the same product set, and users first choose a retailer they want to shop at. We build an individual index for each retailer. We can avoid retrieving products from irrelevant retailers, but the overall ANN model size will increase due to repeated products. The total size of all the indices is less than 4x of the combined index enough to be fitted in a single server instance, so we choose to build per-retailer ANN indices. The pre-computed product embeddings are also indexed into the catalog store daily, so they can be used during online serving.

Online Serving: During serving time, we compute the query embedding with the query encoder and use the ANN service to retrieve top k nearest neighbors in the product embeddings set. Embedding based retrieval (EBR) is efficient especially for non-keyword matching cases which

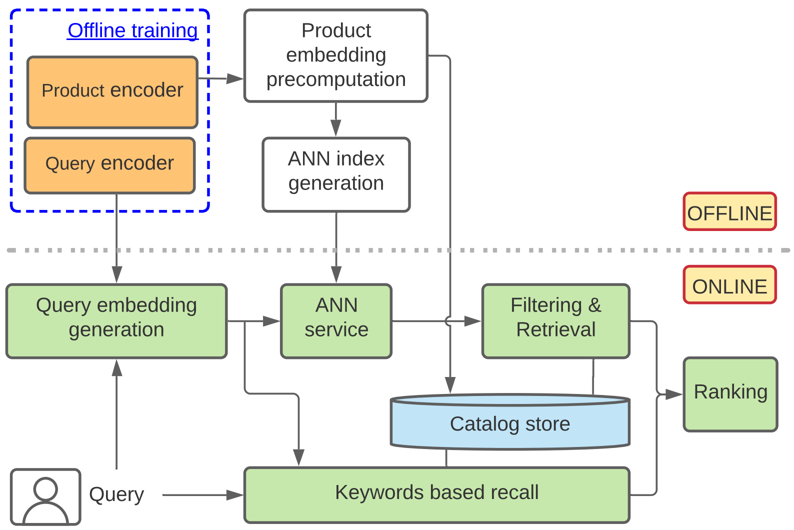


Figure 5.7: Overall system architecture that utilizes the embedding model for retrieval and ranking.

will be discussed in later However, EBR can include irrelevant products occasionally if we use the top k method. To address this challenge, we apply the following rules.

- First we retrieve top k products from the ANN service.
- We drop products if the similarity scores are below a certain threshold regardless of the ranking.
- We construct a white list categories based on the most top several products, and drop the products falling outside the white list categories.
- We apply availability filtering based on warehouse data.
- We finally merge the retrieved products with the keyword-based retrieval results.

5.2 Retrieval-Based Math Question Answering

In section, we further extend our exploration to the information access system with math formulae. The primary task in MathIR is to retrieve relevant information from documents that contain math formulas. However, the heterogeneous nature of math content, which includes rich-structured formulas and their textual context, requires special treatment to create an effective search engine. This is because math languages have special semantic properties such as expression commutativity and symbol substitution equivalence that require consideration different from traditional retrieval models. With the surge of deep neural retrievers, we witness the capacity of deep models being able to boost in-domain effectiveness to a new level. Compared to the previous work [307], we further introduce a domain-adapted retriever backbone named Coco-MAE [309]. We demonstrate improved effectiveness by incorporating structure search and a ColBERT model built on top of this backbone. We participated in the ARQMath 3 Lab, where the math ODQA task was introduced. We demonstrate the system we constructed in this competition, which achieve the best results among all the participants. For math ODQA, retrieval is still the most challenging part although the advanced deep learning methods. We start with an overview of the math IR task.

5.2.1 Related work

Early work in the field of MIR simply applies specialized tokenizers to handle math formulas [167]. Later, different intermediate tree representations are utilized to extract unsupervised features for capturing structure similarities.

Operator Tree Representation: The OPT (Operator Tree) representation is first utilized by extracting features from leaf-root paths [74, 75]. Following work in this direction [290, 70] expand the leaf-root path set by extracting prefixes and suffixes and use them for additionally retrieving sub-expressions in formulas. However, without whole structure comparison, a naive leaf-root path matching offers high recall but lacks good precision in retrieval. More recently, Approach0 [304, 305] approximately matches structure holistically without overlapping features by grouping the root-end nodes of leaf-root paths on the fly.

Symbol Layout Tree: On the other hand, the Symbol Layout Tree (SLT) [293, 214] represents a lower-level structure semantics for math formulas. Similar to the LaTeX representation, it only captures the layout or the topology of a formula. This creates an advantage of little ambiguity in parsing. SLT is adopted as the main representation by a line of MIR works, e.g., the Tangent and Tangent-S systems [185, 294, 296], and the Tangent-L or the MathDowsers system [54, 39, 171, 172, 170]. Local features such as symbols on adjacent nodes or nodes within a distance window and their spatial relations are together tokenized into *math tuples* and used for retrieval.

Data-driven Methods for MIR: Different from structure matching, data-driven methods for MIR discover semantic matches without resorting to formula structure match constraints. Methods in this direction have been explored initially using linear neighbor tokens with limited success [57]. However, models such as Tangent-CFT [157], NTFEM [34], and FORTE [263] first construct structure representation(s) and learn word embeddings or tree embeddings from structure features have later shown effective. Recent advances in natural language processing have led to the use of the Transformer model [245] for the MIR domain. The MathBERT model [187] introduces structure mask in addition to BERT objectives and has been evaluated on the NTCIR-12 collection [295]. SentenceBERT [202] has been domain-adapted for MIR by regressing QA pair scores based on user-generated data in the original Math StackExchange thread [179, 180]. The DPRL QASim method [159] uses two Transformers as similarity assessors, one question-question SentenceBERT [202] assessor pretrained on the Quora website and fine-tuned using related/duplicate links on the MSE website, as well as a question-answer TinyBERT [88] assessor pretrained on the MS-MARCO dataset [173] and fine-tuned on the ARQMath training data. The TU_DBS system utilized an ALBERT-based cross encoder as a primary model [203, 204] which was further pre-trained on the ARQMath corpus directly with a maximum token input of 512.

The ARQMath Lab Datasets

The ARQMath Tasks [158, 161, 162] have been one of the few tasks trying to address the problem of retrieving math questions containing structured formulas. The ARQMath Lab includes multiple tasks:

- **Task 1, Community Question and Answer (CQA).** Retrieval of relevant answer posts from a limited set of Math StackExchange (MSE).
- **Task 2, Retrieving relevant formulas.** The formulas are retrieved together with their context, given a specified query formula from Task 1.
- **Task 3, Open Domain QA task.** ARQMath-3 [162] introduced this new task, which challenges participants to return a single answer for each Task 1 topic.

In this thesis, we mainly cover Task 3, the ODQA part, demonstrate the design choice of building such math retrieval based question answering system. The math retrieval part, we mainly follow Zhong et al. [307, 309], please refer to our original paper [307] for the other details.

5.2.2 Method

Structure Search using Approach0: The main idea of parsing and retrieval on math formula is to matching the the structure according to heuristics. The Approach0 system [304, 305] provide a tool to construct an OPT representation, designed to maximize retrieval recall. Approach0 tokenizes all nodes along the extracted leaf-root paths (and their prefixes). These paths are used similarly as text keywords in regular IR settings, To accelerate query processing, a dynamic pruning strategy [305] has been applied to help the retriever skip evaluating some documents.

Retrieval of Text Keywords: To handle text keywords, we adopt the BM25+ scoring schema [148]. The overall score in Approach Zero is a weighted sum of all partial scores obtained by BM25+ in normal text keywords and those by formula similarity scoring in math formula keywords. We constructed our version of ColBERT [307, 309] to catch the text semantic. We gather the further per-taining data of 1.69 million documents, crawled from the MSE and the Art of Problem-Solving community (AoPS) website.

We merge the top results from both structural and dense retrieval with weighted sum. Studies have demonstrated that this fusion scheme is more empirically robust than other alternatives for retrieval purposes [308, 17].

We report our results on the ARQMath-3 collection [162]. The evaluation of our runs in ARQMath adheres to the official protocols, with the top 1000 hits being considered for each run. For Task 3, we use our Task 1 retriever for producing a set of candidate answers.

QA Model Selection

When tackling the ODQA problem, there are various approaches to consider. One option is to use parametric generative models, which are trained with questions as input and output question-answer pairs without relying on external knowledge. Alternatively, non-parametric models are commonly employed, following a retrieve-and-read framework as we introduced in Chapter 2. We adopt non-parametric models in this task, which offer advantages such as smaller model sizes and the ability to work with limited training data.

In non-parametric models, there are generative or extractive readers. Generally speaking, generative readers either need much training data to transfer themselves to the target domain or at least need a few examples to familiarize themselves with the answer formatting. Unfortunately, we do not have a good amount of training data for MIR. Furthermore, generative models tend to recall non-overlapping spans in the training data, which leads to non-logical answers that are not desired for math question answering. Therefore, we adopt extractive readers in this task.

Generating Candidate Answers

We use our Task 1 retriever for producing a set of candidate answers. As a principle, we choose to base our answer on a single post because different answer posts together will introduce different notations and dialects which create difficulties to align them in the final answer.

We design three strategies for selecting the top 1 candidates:

- **Original**: A straightforward way to narrow down answer posts is to take the top-1 result from Task 1, we will use this as our first strategy.
- **Re-rank**: The next strategy is simply considering a larger set of top results – here we use the top-20 results from Task 1 – and select one of them using the methods in the next subsection.
- **Re-map**: Another strategy is to trust the community: we first try to retrieve the most similar corpus question post given a topic question, then we pick the accepted answer post, if non-exists, the top voted answer post as our candidate answer post.

Snippet Selection

A candidate answer post might be either inappropriately long or exceeding the length limit imposed by the Task 3 requirements, therefore we add a snippet selection step based on the candidate answer posts. We split the answer posts into sentences with the PyA0 sentence splitting utility which takes care of the punctuations and avoids cutting them in the middle of a LaTeX string. Then a window of varied sizes (from a minimum of 5 up to 10 sentences) will go through the beginning of a post to its end to select a combination of sentence spans to form candidate snippets. The beginning of a post usually contains some conclusive answers, or it serves as a good start for smooth reasoning, therefore we also add a selection strategy that always starts the window from the beginning of candidate answer posts.

Finally, we use the same ColBERT model to score the snippets and pick the top 1 snippet as the final answer.

5.2.3 Experiments

We show the results for the Open QA task in Table 5.4. The ARQMath contest hosts hired assessors for result evaluation. They provide mainly two metrics: Average Relevance (AR) score and Precision@1 (P@1). AR is equivalent to the unnormalized Discounted Cumulative Gain at position 1 (DCG@1). Please refer to ARQMath-3 Overview work report[162]. The *Type* column lists the type of the QA model: 'E' for extractive and 'G' for generative. In Task3, *text-davinci-002*, GPT-3 [16] is used as the baseline system. Another generative run *amps3_se1_hints* by the TU_DBS team [205] uses the GPT-2 model [198] but is further fine-tuned on the AMPS dataset [72]. On the other hand, the DPRL run, SBERT-SVMRank [160], uses an extractive approach based on SVM and Sentence-BERT models. The Task 3 effectiveness of our model is

Runs / Description	**Type**	**AR'**	**P'@1**
Others (team / run)			
GPT-3 (baseline)	G	**1.346**	**0.500**
DPRL / SBERT-SVMRank	E	0.462	0.154
TU_DBS / amps3_se1_hints	G	0.325	0.078
Ours			
Re-map ColBERT	E	0.949	0.282
Re-rank Struct. + ColBERT	E	1.179	0.372
Original Struct. + ColBERT	E	1.231	0.397
Re-rank Struct. (Porter) + ColBERT	E	**1.282**	**0.436**

Table 5.4: Effectiveness evaluation for ARQMath-3 Labs Task 3.

largely attributed to the ability for our hybrid search using structure search and data-driven model to generate high top precision in Task 1. However, it is intriguing to note that the text-davinci-002 model, which is a variant of the powerful GPT-3 model, has exhibited a superior ability to answer math questions compared to our extractive approach that relies on a highly effective retriever (while it remains uncertain whether the GPT-3 model merely recalls certain answers from its training data).

Excitingly, large language models such as GPT-3 and others, hold the potential to enable the active selection and complete automation of query processes in our model. It remains to be seen whether a generative approach will eventually become the dominant method for directly and comprehensively handling queries in this domain.

Chapter 6

Conclusions

6.1 Summary of Findings

In this thesis, we discussed several aspects of designing a neural model for domain-specific retrieval-based information access.

In Chapter 2, we provided an introduction to RBQA systems. We defined the problem statement and discussed the dataset used for training and evaluation purposes. Additionally, we explained the evaluation metrics employed to assess the performance of the system.

In Chapter 3, we delve into the intricacies of each component within the RBQA system. We introduced BERTserini, an end-to-end open-domain question answering system that seamlessly integrates BERT and the Anserini IR toolkit. Leveraging a straightforward two-stage pipeline architecture, our system achieved substantial advancements compared to previous approaches. A key challenge we addressed in this chapter was enabling cross-paragraph comparison in the reader model within the RBQA framework. Furthermore, we examined the trade-off space within the retriever component, focusing on the dimensions of index-based and ANN-based candidate generation in the context of RBQA. We discovered that the computational overhead associated with BERT-style reader models significantly restricts the flexibility of trade-offs within the retriever. Additionally, we delved into strategies for training an improved reader model through data augmentation in the RBQA setup. Our findings indicated that fine-tuning the reader model using a distant-to-close dataset schedule facilitates successful knowledge transfer to new distributions. Moreover, we concluded that randomly selecting positive examples based on distant supervision aids the model in learning more diverse distributions. We showcased the state-of-the-art results achieved by our system in retrieval-based question answering tasks across two English and two Chinese datasets sourced from Wikipedia. These exceptional outcomes were obtained by leveraging effective data augmentation techniques.

In Chapter 4, we focus on the study of adapting retrieval-based systems to specific domains without relying on domain-specific training data. The chapter begins with an introduction to the concept of domains and methods for distinguishing them. We then establish baselines for domain adaptation, including directly applying retrieval-based question answering systems trained on general domain data and fine-tuning models with limited in-domain training data. Next, we explore several zero-shot adaptation methods, including question generation as data augmentation, in-domain further pre-training, and progressive domain-clustered training. Through our experiments, we determine that the most effective method among these is further pre-training in the specific domain. Additionally, we propose the GPT-based Compressor as a re-ranker for the retrieval-based system. This approach does not require additional labeled data, prompt tuning, or domain-specific knowledge. Despite these simplifications, the GPT-based Compressor achieves comparable results to the aforementioned adaptation methods.

In Chapter 5, we expand our investigation to incorporate non-textual materials into the retrieval-based systems. Specifically, we explore two main directions: retrieval from e-commerce databases and question answering involving mathematical formulae. For the e-commerce database retrieval system, we propose cross architecture cascade training as a method to effectively train the model on noisy data and address the challenge of cold start. We also introduce self-adversarial negative sampling and re-weighting techniques to improve the model's performance. These approaches help to tackle the inherent difficulties in retrieving information from e-commerce databases. In the domain of math question answering, we combine the ColBERT model with the Approach0 framework to enhance the retrieval of mathematical formulae and other math-related content. We propose the use of answer sentence selection with a ColBERT-based re-ranker to improve the selection of the correct answer. Our approach yields the best results among all the participants in the task.

6.2 Discussion

I start writing this thesis in the winter of 2022, and the thesis defence was in August of 2023. Throughout this time span, the field of Natural Language Processing (NLP) experienced significant developments. Within this discourse, we delve into the two primary streams of research that emerged.

Close-Book QA with ChatGPT

A particularly impactful avenue of exploration is centered around ChatGPT [182] and its subsequent iterations. ChatGPT stands as a remarkable artificial intelligence language model within the GPT series. Tailored for conversational contexts, it underwent extensive training on an expansive corpus of internet text encompassing books, articles, websites, and more. The model's pre-training phase involved assimilating language patterns and structures from diverse datasets, honing its ability to predict forthcoming words within sentences. Fine-tuning followed, incorporating human reviewer feedback to align the model's responses with human values and context, thus ensuring interactions that are both safe and pertinent.

ChatGPT excels across tasks demanding few-shot generalization and language comprehension, propelling a diverse spectrum of NLP challenges to new pinnacles. Its proficiency in generating coherent, precise text brings forth a multitude of advantages.

We tested ChatGPT with the exact same questions as in Figure 3.3, the outcomes of which are delineated in Table 6.1. The results show ChatGPT's capability to furnish highly reasonable answers. However, in comparison to our system, it tends to generate lengthy responses. Furthermore, it adopts a cautious stance when confronted with vague questions. These aspects, amenable to prompt tuning, can be addressed to yield a robust baseline system for factoid question answering.

Retrieval-based Language Models

Large generative models such as GPT, relying solely on pre-trained language model parameters to encode a wealth of world knowledge, require a prohibitively large number of parameters, and often struggle to learn long-tail knowledge. Moreover, these parametric language models are incapable of adapting over time, often hallucinate, and may leak private data from the training corpus.

To surmount these challenges, an increasing interest has emerged in the realm of retrieval-based language models [4]. These models integrate non-parametric datastores, such as text fragments from external corpuses, with their parametric counterparts, akin to the approach laid out by REALM [69]. This thesis aligns itself within this category, contributing to the progression of this direction.

Question	Answer
Where did Teska work in Budapest?	I’m sorry, but as of my last knowledge update in September 2021, I don’t have information about an individual named “Teska” or their place of work in Budapest. It’s possible that this information might have become available after my last update or that it’s not widely known. If Teska is a relatively recent or obscure figure, you might want to try searching online or referring to more current sources for accurate information.
What was the very first television station to broadcast in Fresno?	The very first television station to broadcast in Fresno, California, was KFRE-TV (now KSEE-TV). It began broadcasting on June 1, 1953. KFRE-TV was an NBC affiliate and played a significant role in bringing television programming to the Fresno area.
Why did Mark Twain call the late 19th century the glided age?	Mark Twain, along with his co-author Charles Dudley Warner, referred to the late 19th century as the ”Gilded Age” in their novel of the same name, published in 1873. The term ”Gilded Age” was used ironically to describe the era’s superficial appearance of prosperity and progress, which masked underlying social, economic, and political problems. The word ”gilded” implies a thin layer of gold covering something of lesser value underneath. Twain and Warner used this term to critique the era’s excessive wealth, materialism, and ostentatious displays of affluence, often seen among the upper class and industrialists. They saw this superficial veneer as masking issues like corruption, social inequality, and the exploitation of labor. In essence, Mark Twain used the term ”Gilded Age” to convey the idea that while there might have been a shiny surface of wealth and progress, beneath that surface lay a reality that was not as perfect or prosperous for everyone as it appeared.

continued . . .

... continued

Question	Answer
Why do geese honk while flying?	Geese honk while flying for several reasons, primarily related to communication and coordination within their flock during migration. Here are some of the main reasons why geese honk while in flight: 1. **Communication**: Honking is a way for geese to communicate with each other while flying. It helps them stay in contact, share information about their flight path, and maintain their formation. The honking serves as a form of vocal bonding that keeps the flock together. 2. **Coordination**: Geese fly in V-shaped formations during migration, which offers several aerodynamic benefits. The honking helps them coordinate their movements within the formation. The lead goose sets the pace, and the other geese follow, adjusting their positions to reduce wind resistance. The honking signals help maintain the formation and synchronize their flight. 3. **Motivation**: The honking of geese can also serve as a way to encourage each other to keep up the pace during long flights. It provides motivation and support to the entire flock. 4. **Navigation**: Geese migrate over long distances, and their honking can help them navigate and stay on course. The vocal signals provide information about the direction and location of the flock. 5. **Safety and Alertness**: Honking can serve as an alarm call to warn the flock about potential dangers, such as predators or obstacles. Geese are known to be protective of each other and use their vocalizations to alert others to possible threats. 6. **Group Identity**: Honking helps establish and reinforce the social bonds within the flock. It creates a sense of identity and belonging among the geese, which is important for their collective survival. Overall, the honking behavior of geese while flying is a complex and evolved communication strategy that helps them navigate, cooperate, and ensure the safety and cohesion of their flock during migration.

continued ...

...continued

Question	Answer
What's the weather in Canada?	I don't have access to real-time data as my browsing capability is disabled and my information only goes up until September 2021. Weather conditions can vary widely across Canada due to its vast size and diverse geography. It could be warm and sunny in some areas while cold and snowy in others, depending on the time of year and the specific region you're asking about. If you're looking for current weather information in Canada, I recommend using a reliable weather website or app, or you can use a search engine to find up-to-date weather forecasts for the specific city or region you're interested in.

Table 6.1: ChatGPT answers for questions tested with BERTserini in Figure 3.3.

6.3 Future Directions

Reasoning and Explanation: Enhancing the interpretability and explainability of deep learning models in domain-specific applications is crucial, especially in domains where trust and transparency are critical. Future research can focus on developing techniques for interpreting and explaining the model's decisions and generating human-understandable justifications for the answers provided, enabling users to trust and understand the model's responses.

Integration of Domain-specific Knowledge: Incorporating external knowledge sources specific to the domain can significantly enhance the performance of pre-trained models. Techniques such as knowledge graph integration, ontology-based reasoning can be leveraged to improve the model's ability to access and utilize relevant domain knowledge during the question answering process. Despite our initial exploration in e-commerce database search, numerous unexplored research directions remain for further investigation.

Real-time Information Access: Improving the responsiveness and interactivity of ODQA systems is another area of interest. Future work could focus on developing real-time systems that provide instant responses to user queries. For industry applications, resource constraints often necessitate the use of lightweight models to maintain high query per second (QPS) rates. Cache mechanisms and distilled models are commonly employed, with even linear regression models finding widespread use in the industry. Consequently, the task of harnessing the capabilities of large-scale models for real-time inference poses intriguing and challenging opportunities.

Multi-modal, Multi-lingual: The evolution of retrieval-based question answering systems is poised to embrace a new era of sophistication through the integration of multi-modal and multi-lingual capabilities. These systems will adeptly navigate the intricacies of human communication by seamlessly fusing text, images, audio, and video data. This convergence will enable users to not only ask questions in their preferred language but also receive responses that incorporate information from diverse linguistic sources. Furthermore, these advanced systems will discern context not just from text, but also from visual and auditory cues, leading to a deeper understanding of user intent. To achieve such systems, we need to overcome challenges in integration of low-resource language and more diverse sources.

References

[1] Dogu Araci and Zulkuf Genc. Finbert: Finacial sentiment analysis with pre-trained language models, 2020.

[2] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 1998.

[3] N. Asadi and J. Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *SIGIR*, 2013.

[4] Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. Acl 2023 tutorial: Retrieval-based language models and applications. *ACL 2023*, 2023.

[5] A. Atreya and C. Elkan. Latent semantic indexing (LSI) fails for TREC collections. *SIGKDD Explorations*, 2010.

[6] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. Sparterm: Learning term-based sparse representation for fast text retrieval, 2020.

[7] Fabrice Bellard. Lossless data compression with transformer, 2021.

[8] Eyal Ben-David, Nadav Oved, and Roi Reichart. Pada: Example-based prompt learning for on-the-fly adaptation to unseen domains. *Transactions of the Association for Computational Linguistics*, 2022.

[9] Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. Information distance. *IEEE Transactions on Information Theory*, 1998.

[10] Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*, 2009.

[11] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly, 2009.

[12] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 2017.

[13] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv:1506.02075*, 2015.

[14] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *2015 EMNLP*, 2015.

[15] L. Boytsov, D. Novak, Y. Malkov, and E. Nyberg. Off the beaten path: Let's replace term-based retrieval with k-NN search. In *CIKM*, 2016.

[16] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *NeruIPS*, 2020.

[17] Sebastian Bruch, Siyu Gai, and Amir Ingber. An analysis of fusion functions for hybrid retrieval. *ACM Trans. Inf. Syst.*, 2023.

[18] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. *ICML*, 2007.

[19] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, Guajardo-Cespedes M, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil. Universal sentence encoder for English. In *EMNLP: System Demonstrations*, 2018.

[20] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017.

[21] Danqi Chen. *Neural Reading Comprehension and Beyond*. PhD thesis, Stanford University, 2018.

[22] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *55th ACL*, 2017.

[23] Xin Chen, Brent Francia, Ming Li, Brian Mckinnon, and Amit Seker. Shared information and program plagiarism detection. *IEEE Transactions on Information Theory*, 2004.

[24] Z. Chen, H. Zhang, X. Zhang, and L. Zhao. Quora question pairs, 2018.

[25] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, 2016.

[26] M. Chidambaram, Y. Yang, D. Cer, S. Yuan, Y. Sung, B. Strope, and R. Kurzweil. Learning cross-lingual sentence representations via a multi-task dual-encoder model. In *RepL4NLP-2019*, 2019.

[27] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *CoRR*, 2003.

[28] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. In *56th ACL*, 2018.

[29] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *8th ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[30] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, 2017.

[31] Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. A span-extraction dataset for Chinese machine reading comprehension. In *2019 EMNLP-IJCNLP*, 2019.

[32] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *NeruIPS*, 2015.

[33] Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. Using similarity measures to select pretraining data for NER. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics , Volume 1 (Long and Short Papers)*, 2019.

[34] Yifan Dai, Liangyu Chen, and Zihan Zhang. An n-ary tree-based model for similarity evaluation on mathematical formulae. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020.

[35] Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Web Conference 2020*, 2020.

[36] Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*, WWW '20. Association for Computing Machinery, 2020.

[37] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, 2018.

[38] Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. In *The Eleventh ICLR*, 2023.

[39] Fraser Dallas. Math information retrieval using a text search engine. Master's thesis, University of Waterloo, 2018.

[40] L. Peter Deutsch. Gzip file format specification version 4.3, 1996.

[41] Josh Devins, Julie Tibshirani, and Jimmy Lin. Aligning the research and practice of building search applications: Elasticsearch and pyserini. In *Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, 2022.

[42] J. Devlin, M. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[43] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*, 2005.

[44] Matthijs Douze, Arthur Szlam, Bharath Hariharan, and Hervé Jégou. Low-shot learning with large-scale diffusion. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[45] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 2011.

[46] Harrison Edwards and Amos Storkey. Towards a neural statistician. *ICLR (ICLR)*, 2016.

[47] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 1990.

[48] Yair Feldman and Ran El-Yaniv. Multi-hop paragraph retrieval for open-domain question answering. In *57th ACL*, 2019.

[49] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 2017.

[50] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *2nd Workshop on Machine Reading for Question Answering*, 2019.

[51] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. Splade: Sparse lexical and expansion model for first stage ranking. In *44th SIGIR*, 2021.

[52] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. From distillation to hard negative sampling: Making sparse neural ir models more effective. In *45th SIGIR*, SIGIR '22, 2022.

[53] C. C. Foster. Information retrieval: Information storage and retrieval using avl trees. In *1965 20th National Conference*, ACM '65, 1965.

[54] Dallas Fraser, Andrew Kane, and Frank Wm. Tompa. Choosing math features for bm25 ranking with tangent-l. In *ACM Symposium on Document Engineering 2018*, DocEng '18, 2018.

[55] C. Fu, C. Wang, and D. Cai. Satellite system graph: Towards the efficiency up-boundary of graph-based approximate nearest neighbor search. *arXiv*, 2019.

[56] Hang Gao, Zheng Shou, Alireza Zareian, Hanwang Zhang, and Shih-Fu Chang. Low-shot learning via covariance-preserving adversarial augmentation networks. In *NeruIPS (NeurIPS)*, 2018.

[57] Liangcai Gao, Zhuoren Jiang, Yue Yin, Ke Yuan, Zuoyu Yan, and Zhi Tang. Preliminary exploration of formula embedding for mathematical information retrieval: can mathematical formulae be embedded like a natural language? *CoRR*, 2017.

[58] Luyu Gao and Jamie Callan. Condenser: a pre-training architecture for dense retrieval. In *2021 EMNLP*, 2021.

[59] Luyu Gao, Zhuyun Dai, and Jamie Callan. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *2021 NAACL*, 2021.

[60] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels, 2022.

[61] Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. Tevatron: An efficient and flexible toolkit for dense retrieval. *SIGIR 2023*, 2022.

[62] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *2021 EMNLP*, 2021.

[63] Martin Geletka, Vojtěch Kalivoda, Michal Štefánik, Marek Toma, and Petr Sojka. Diverse semantics representation is king: Mirmu and msm at arqmath 2022. In *CLEF*, 2022.

[64] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[65] Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Deepzip: Lossless data compression using recurrent neural networks. In *2019 Data Compression Conference (DCC)*, 2019.

[66] Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Dzip: improved general-purpose loss less compression based on novel neural network modeling. In *2021 Data Compression Conference (DCC)*, 2021.

[67] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *25th CIKM 2016*, 2016.

[68] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *58th ACL*, 2020.

[69] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[70] Hiroya Hagino and Hiroaki Saito. Partial-match retrieval with structure-reflected indices at the ntcir-10 math task. In *NTCIR*, 2013.

[71] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply, 2017.

[72] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[73] Michael Herrera and Kasey Luo. Lossless neural text compression, 2021.

[74] Yoshinori Hijikata, Hideki Hashimoto, and Shogo Nishida. An investigation of index formats for the search of mathml objects. In *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, 2007.

[75] Yoshinori Hijikata, Hideki Hashimoto, and Shogo Nishida. Search mathematical formulas by mathematical formulas. In Michael J. Smith and Gavriel Salvendy, editors, *Human Interface and the Management of Information. Designing Information Environments, Symposium on Human Interface 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009, Procceedings, Part I*, Lecture Notes in Computer Science, 2009.

[76] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[77] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling. *CoRR*, 2021.

[78] Emiel Hoogeboom, Jorn Peters, Rianne Van Den Berg, and Max Welling. Integer discrete flows and lossless compression. *NeruIPS*, 2019.

[79] Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng li. Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension. In *ACL 2019*, 2019.

[80] Cynthia Huang, Yuqing Xie, Zhiying Jiang, Jimmy Lin, and Ming Li. Approximating human-like few-shot learning with gpt-based compression, 2023.

[81] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. Embedding-based retrieval in facebook search. In *26th ACM SIGKDD*, 2020.

[82] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *8th ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[83] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021.

[84] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*, 2011.

[85] Zhiying Jiang, Yiqin Dai, Ji Xin, Ming Li, and Jimmy Lin. Few-shot non-parametric learning with deep latent variable model. In *NeruIPS*, 2022.

[86] Zhiying Jiang, Rui Wang, Dongbo Bu, and Ming Li. A theory of human-like few-shot learning, 2023.

[87] Zhiying Jiang, Matthew Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, and Jimmy Lin. "low-resource" text classification: A parameter-free classification method with compressors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6810–6828, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.426. URL https://aclanthology.org/2023.findings-acl.426.

[88] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling BERT for natural language understanding. In Trevor Cohn, Yulan He, and Yang Liu, editors, *EMNLP 2020, Online Event, 16-20 November 2020*, Findings of ACL, 2020.

[89] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 2019.

[90] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *55th ACL*, 2017.

[91] Ehsan Kamalloo, Nandan Thakur, Carlos Lassance, Xueguang Ma, Jheng-Hong Yang, and Jimmy Lin. Resources for brewing beir: Reproducible reference models and an official leaderboard, 2023.

[92] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *2020 EMNLP*, 2020.

[93] Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. Towards parameter-free data mining. In ., KDD '04, 2004.

[94] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, 2020.

[95] Omar Khattab, Christopher Potts, and Matei Zaharia. Relevance-guided supervision for OpenQA with ColBERT. *Transactions of the Association for Computational Linguistics*, 2021.

[96] Omar Khattab, Christopher Potts, and Matei Zaharia. Baleen: Robust multi-hop reasoning at scale via condensed retrieval. In *Advances in Neural Information Processing Systems*, 2021.

[97] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd ICLR 2015*, 2015.

[98] Friso H. Kingma, P. Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, 2019.

[99] Ralf Klinkenberg. Meta-learning, model selection, and example selection in machine learning domains with concept drift, 2005.

[100] A.N. Kolmogorov. Three approaches to the quantitative definition of information. *PROBLEMS OF INFORMATION TRANSMISSION*, 1965.

[101] Bernhard Kratzwald and Stefan Feuerriegel. Adaptive document retrieval for deep question answering. In *2018 EMNLP*, 2018.

[102] Bernhard Kratzwald, Anna Eigenmann, and Stefan Feuerriegel. Rankqa: Neural question answering with answer re-ranking. In *ACL*, 2019.

[103] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[104] Roland Kwitt, Sebastian Hegenbart, and Marc Niethammer. One-shot learning of scene locations via feature trajectory transfer. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[105] Yanyan Lan, Yadong Zhu, Jiafeng Guo, Shuzi Niu, and Xueqi Cheng. Position-aware listmle: A sequential learning process for ranking. In *Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, 2014.

[106] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite BERT for self-supervised learning of language representations. In *8th ICLR*, 2020.

[107] Walter I. Landauer. The balanced tree and its utilization in information retrieval. *IEEE Transactions on Electronic Computers*, 1963.

[108] Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. Ranking paragraphs for improving answer recall in open-domain question answering. In *2018 EMNLP*, 2018.

[109] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 2019.

[110] Jinhyuk Lee, Minjoon Seo, Hannaneh Hajishirzi, and Jaewoo Kang. Contextualized sparse representations for real-time open-domain question answering. In *58th ACL*, 2020.

[111] Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. Learning dense representations of phrases at scale. In *59th ACL and the 11th International Joint Conference on Natural Language Processing*, 2021.

[112] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *57th ACL*, 2019.

[113] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *58th ACL*, 2020.

[114] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *34th NeurIPS*, 2020.

[115] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeruIPS 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[116] Hang Li, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and Guido Zuccon. To interpolate or not to interpolate: Prf, dense and sparse retrievers. In *45th SIGIR*, SIGIR '22, 2022.

[117] Ming Li. Information distance and its applications. In *11th International Conference on Implementation and Application of Automata, CIAA*, 2006.

[118] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 2008.

[119] Ming Li, Jonathan H. Badger, Xin Chen, Sam Kwong, Paul Kearney, and Haoyong Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 2001.

[120] Ming Li, Xin Chen, Xin Li, Bin Ma, and P.M.B. Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 2004.

[121] Minghan Li, Ming Li, Kun Xiong, and Jimmy Lin. Multi-task dense retrieval via model uncertainty fusion for open-domain question answering. In *Findings: EMNLP 2021*, 2021.

[122] Minghan Li, Xueguang Ma, and Jimmy Lin. An encoder attribution analysis for dense passage retriever in open-domain question answering. In *2nd Workshop on Trustworthy Natural Language Processing (TrustNLP 2022)*, 2022.

[123] Minghan Li, Xinyu Zhang, Ji Xin, Hongyang Zhang, and Jimmy Lin. Certified error control of candidate set pruning for two-stage relevance ranking. In *2022 EMNLP*, 2022.

[124] Minghan Li, Sheng-Chieh Lin, Xueguang Ma, and Jimmy Lin. Slim: Sparsified late interaction for multi-vector retrieval with inverted indexes. *ArXiv*, 2023.

[125] Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. Embedding-based product retrieval in taobao search. In *27th ACM SIGKDD*, KDD '21, 2021.

[126] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021.

[127] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

[128] Jimmy Lin. The neural hype and comparisons against weak baselines. *SIGIR Forum*, 2018.

[129] Jimmy Lin. A proposed conceptual framework for a representational approach to information retrieval, 2021.

[130] Jimmy Lin. A proposed conceptual framework for a representational approach to information retrieval. *SIGIR Forum*, 2022.

[131] Jimmy Lin and Xueguang Ma. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. *CoRR*, 2021.

[132] Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and David R. Karger. The role of context in question answering systems. In *Extended abstracts of the 2003 Conference on Human Factors in Computing Systems (CHI 2003)*, 2003.

[133] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *44th SIGIR*, 2021.

[134] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *44th SIGIR 202)*, 2021.

[135] Jimmy J. Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig Macdonald, and Sebastiano Vigna. Toward reproducible baselines: The open-source ir reproducibility challenge. In *European Conference on Information Retrieval*, 2016.

[136] Sheng-Chieh Lin and Jimmy Lin. A dense representation framework for lexical and semantic matching. *ACM Trans. Inf. Syst.*, 2023.

[137] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, 2021.

[138] Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen tau Yih, and Xilun Chen. How to train your dragon: Diverse augmentation towards generalizable dense retrieval, 2023.

[139] Sheng-Chieh Lin, Minghan Li, and Jimmy Lin. Aggretriever: A Simple Approach to Aggregate Textual Representations for Robust Dense Passage Retrieval. *Transactions of the Association for Computational Linguistics*, 2023.

[140] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. Denoising distantly supervised open-domain question answering. In *56th ACL (Volume 1: Long Papers)*, 2018.

[141] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. Denoising distantly supervised open-domain question answering. In *56th ACL*, 2018.

[142] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *NeruIPS*, 2022.

[143] Linqing Liu, Patrick Lewis, Sebastian Riedel, and Pontus Stenetorp. Challenges in generalization in open domain question answering. In *NAACL 2022*, 2022.

[144] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 2023.

[145] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2020.

[146] Yiqun Liu, Kaushik Rangadurai, Yunzhong He, Siddarth Malreddy, Xunlong Gui, Xiaoyi Liu, and Fedor Borisyuk. Que2search: Fast and accurate query and document understanding for search at facebook. In *27th ACM SIGKDD*, KDD '21, 2021.

[147] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[148] Yuanhua Lv and ChengXiang Zhai. Lower-bounding term frequency normalization. In *CIKM*, 2011.

[149] Ji Ma, Ivan Korotkov, Yinfei Yang, Keith B. Hall, and Ryan T. McDonald. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *16th Conference of the European Chapter of the*

Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021, 2021.

[150] Xueguang Ma, Minghan Li, Kai Sun, Ji Xin, and Jimmy Lin. Simple and effective unsupervised redundancy elimination to compress dense vectors for passage retrieval. In *2021 EMNLP*, 2021.

[151] Xueguang Ma, Kai Sun, Ronak Pradeep, Minghan Li, and Jimmy Lin. Another look at dpr: Reproduction of training and replication of retrieval. In *Advances in Information Retrieval*, 2022.

[152] Xueguang Ma, Tommaso Teofili, and Jimmy Lin. Anserini gets dense retrieval: Integration of lucene's hnsw indexes, 2023.

[153] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. Zero-shot listwise document reranking with a large language model, 2023.

[154] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge: Financial opinion mining and question answering. In *Companion The Web Conference 2018*, WWW '18, 2018.

[155] Y. Malko and D. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *PAMI*, 2016.

[156] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. Learning passage impacts for inverted indexes. In *44th SIGIR*, SIGIR '21, 2021.

[157] Behrooz Mansouri, Shaurya Rohatgi, Douglas W. Oard, Jian Wu, C Lee Giles, and Richard Zanibbi. Tangent-cft: An embedding model for mathematical formulas. In *SIGIR*, 2019.

[158] Behrooz Mansouri, Anurag Agarwal, Douglas Oard, and Richard Zanibbi. Finding old answers to new math questions: The ARQMath Lab at CLEF 2020. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval*, 2020.

[159] Behrooz Mansouri, Douglas W. Oard, and Richard Zanibbi. Dprl systems in the CLEF 2021 ARQMath lab: Sentence-BERT for answer retrieval, learning-to-rank for formula retrieval. In *CLEF*, 2021.

[160] Behrooz Mansouri, Douglas W Oard, and Richard Zanibbi. Dprl systems in the clef 2022 arqmath lab: Introducing mathamr for math-aware search. *CLEF*, 2021.

[161] Behrooz Mansouri, Richard Zanibbi, Douglas W. Oard, and Anurag Agarwal. Overview of ARQMath-2 (2021): Second CLEF lab on answer retrieval for questions on math (working notes version). In *Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum*, 2021.

[162] Behrooz Mansouri, Vít Novotný, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. Overview of ARQMath-3 (2022): Third CLEF lab on Answer Retrieval for Questions on Math (Working Notes Version). In *Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum*, 2022.

[163] Yu Mao, Yufei Cui, Tei-Wei Kuo, and Chun Jason Xue. Trace: A fast transformer-based general-purpose lossless compressor. In *ACM Web Conference 2022*, WWW '22, 2022.

[164] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th ICLR 2017, Toulon, France, April 24-26, 2017,*, 2017.

[165] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, 2010.

[166] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[167] Bruce R. Miller and Abdou Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 2003.

[168] Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. Efficient and robust question answering from minimal context over documents. In *56th ACL*, 2018.

[169] Sewon Min, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, and Hannaneh Hajishirzi. Joint passage ranking for diverse multi-answer retrieval. In *2021 EMNLP*, 2021.

[170] Yin Ki Ng. Dowsing for math answers: Exploring MathCQA with a math-aware search engine. Master's thesis, University of Waterloo, 2021.

[171] Yin Ki Ng, Dallas J. Fraser, Besat Kassaie, George Labahn, Mirette S Marzouk, Frank Tompa, and Kevin Wang. Dowsing for math answers with Tangent-L. In *CLEF*, 2020.

[172] Yin Ki Ng, Dallas Fraser, Besat Kassaie, and Frank Tompa. Dowsing for answers to math questions: Ongoing viability of traditional MathIR. In *CLEF*, 2021.

[173] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset, 2016.

[174] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Allen Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. Semantic product search. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *25th ACM SIGKDD, Anchorage, AK, USA, August 4-8, 2019*, 2019.

[175] R. Nogueira, W. Yang, J. Lin, and K. Cho. Document expansion by query prediction. *arXiv*, 2019.

[176] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *CoRR*, 2019.

[177] Rodrigo Nogueira and Jimmy Lin. From doc2query to docttttquery, 2019.

[178] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *EMNLP 2020*, 2020.

[179] Vít Novotnỳ, Petr Sojka, Michal Stefánik, and Dávid Lupták. Three is better than one: Ensembling math information retrieval systems. In *CLEF*, 2020.

[180] Vít Novotnỳ, Michal Štefánik, Dávid Lupták, Martin Geletka, Petr Zelina, and Petr Sojka. Ensembling ten math information retrieval systems. In *CLEF*, 2021.

[181] OpenAI. Gpt-4 technical report, 2023.

[182] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeruIPS*, 2022.

[183] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *40th Annual Meeting on Association for Computational Linguistics*, ACL '02, 2002.

[184] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeruIPS 32*, 2019.

[185] Nidhin Pattaniyil and Richard Zanibbi. Combining tf-idf text retrieval with an inverted index over symbol pairs in math expressions: The tangent math search engine at ntcir 2014. In *NTCIR*, 2014.

[186] Jan Pedersen. Query understanding at bing. In *SIGIR*, 2010.

[187] Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. Mathbert: A pre-trained model for mathematical formula understanding. *CoRR*, 2021.

[188] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *2014 EMNLP*, 2014.

[189] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. *Advances in neural information processing systems*, 34:11054–11070, 2021.

[190] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019.

[191] Tomas Pfister, James Charles, and Andrew Zisserman. Domain-adaptive discriminative one-shot learning of gestures. In *European Conference on Computer Vision (ECCV)*, 2014.

[192] Barbara Plank and Gertjan van Noord. Effective measures of domain similarity for parsing. In *49th ACL*, 2011.

[193] Nick Pogrebnyakov and Shohreh Shaghaghian. Predicting the success of domain adaptation in text similarity. In *6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, 2021.

[194] Ronak Pradeep, Haonan Chen, Lingwei Gu, Manveer Singh Tamber, and Jimmy Lin. Pygaggle: A gaggle of resources for open-domain question answering. In *Advances in Information Retrieval*, 2023.

[195] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 1999.

[196] Xiao Qin, Nasrullah Sheikh, Berthold Reinwald, and Lingfei Wu. Relation-aware graph attention model with adaptive self-adversarial training. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

[197] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[198] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[199] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

[200] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *2016 EMNLP, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, 2016.

[201] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In Iryna Gurevych and Yusuke Miyao, editors, *56th ACL, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, 2018.

[202] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *2019 EMNLP and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[203] Anja Reusch, Maik Thiele, and Wolfgang Lehner. An albert-based similarity measure for mathematical answer retrieval. In *44th SIGIR*, SIGIR '21, 2021.

[204] Anja Reusch, Maik Thiele, and Wolfgang Lehner. Tu_dbs in the ARQMath lab 2021, CLEF. In *CLEF*, 2021.

[205] Anja Reusch, Maik Thiele, and Wolfgang Lehner. Transformer-encoder and decoder models for questions on math. *CLEF*, 2022.

[206] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, 2015.

[207] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Thirteenth National Conference on Artificial Intelligence*, 1996.

[208] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 2009.

[209] Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, 1995.

[210] Sebastian Ruder. *Neural Transfer Learning for Natural Language Processing*. PhD thesis, National University of Ireland, Galway, 2019.

[211] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS 2019*, 2019.

[212] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. Plaid: An efficient engine for late interaction retrieval. In *31st CIKM*, CIKM '22, 2022.

[213] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *2022 NAACL*, 2022.

[214] Thomas Schellenberg, Bo Yuan, and Richard Zanibbi. Layout-based substitution tree indexing and retrieval for mathematical expressions. In *DRR*, 2012.

[215] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 1997.

[216] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *54th ACL*, 2016.

[217] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *ICLR*, 2017.

[218] Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Real-time open-domain question answering with dense-sparse phrase index. In *57th ACL*, 2019.

[219] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 1948.

[220] Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. Drcd: a Chinese machine reading comprehension dataset. *arXiv:1806.00920*, 2018.

[221] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, 2018.

[222] Harshdeep Singh, Robert West, and Giovanni Colavizza. Wikipedia citations: A comprehensive data set of citations with identifiers extracted from english wikipedia. *Quantitative Science Studies*, 2020.

[223] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *NeruIPS (NeurIPS)*, 2017.

[224] Ian Soboroff, Shudong Huang, and Donna Harman. Trec 2019 news track overview. In *TREC*, 2019.

[225] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast WordPiece tokenization. In *2021 EMNLP*, 2021.

[226] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In Maosong Sun, Xuanjing Huang, Heng Ji, Zhiyuan Liu, and Yang Liu, editors, *CCL*, 2019.

[227] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. *AAAI Conference on Artificial Intelligence*, 2020.

[228] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

[229] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[230] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeruIPS (NeurIPS)*, 2014.

[231] Mike Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: Optimising non-smooth rank metrics. In *WSDM 2008*, 2008.

[232] Stefanie Tellex, Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes. Quantitative evaluation of passage retrieval algorithms for question answering. In *26th Annual SIGIR (SIGIR 2003)*, 2003.

[233] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[234] Nandan Thakur, Nils Reimers, and Jimmy Lin. Domain adaptation for memory-efficient dense retrieval, 2022.

[235] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[236] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[237] James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *ICLR*, 2019.

[238] Andrew Trotman, Joel Mackenzie, Pradeesh Parameswaran, and Jimmy Lin. A common framework for exploring document-at-a-time and score-at-a-time retrieval methods. In *45th SIGIR*, SIGIR '22, 2022.

[239] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 2015.

[240] Zhengkai Tu, Wei Yang, Zihang Fu, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. Approximate nearest neighbor search and lightweight dense vector reranking in multi-stage retrieval architectures. In *2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, ICTIR '20, 2020.

[241] Mozes van de Kar, Mengzhou Xia, Danqi Chen, and Mikel Artetxe. Don't prompt, search! mining-based zero-shot learning with language models. In *2022 EMNLP*, 2022.

[242] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008.

[243] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 2020.

[244] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 1971.

[245] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeruIPS 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017.

[246] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *NeruIPS (NeurIPS)*, 2016.

[247] Paul M. B. Vitányi. Conditional kolmogorov complexity and universal probability. *CoRR*, 2012.

[248] Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. Trec-covid: Constructing a pandemic information retrieval test collection. *SIGIR Forum*, 2021.

[249] Ellen M. Voorhees and Dawn M. Tice. The TREC-8 Question Answering Track evaluation. In *Eighth Text REtrieval Conference (TREC-8)*, 1999.

[250] Henning Wachsmuth, Shahbaz Syed, and Benno Stein. Retrieval of the best counterargument without prior topic knowledge. In *56th ACL*, 2018.

[251] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *2020 EMNLP*, 2020.

[252] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *2020 EMNLP*, 2020.

[253] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[254] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. In *5th ICLR 2017, Toulon, France, April 24-26, 2017,*, 2017.

[255] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. R3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18, 2018.

[256] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. Evidence aggregation for answer re-ranking in open-domain question answering. In *ICLR*, 2018.

[257] Siyu Wang, Jianfei Chen, Chongxuan Li, Jun Zhu, and Bo Zhang. Fast lossless neural compression with integer-only discrete flows. In *International Conference on Machine Learning*, 2022.

[258] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. In *ICLR*, 2020.

[259] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *55th ACL*, 2017.

[260] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeruIPS 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[261] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *2019 EMNLP and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[262] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *2019 EMNLP and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[263] Zichao Wang, Andrew S Lan, and Richard G Baraniuk. Mathematical formula representation via tree embeddings. In *iTextbooks@ AIED*, 2021.

[264] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. In Yoshua Bengio and Yann LeCun, editors, *4th ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016,*, 2016.

[265] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *2018 NAACL*, 2018.

[266] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 1987.

[267] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *2020 EMNLP: System Demonstrations*, 2020.

[268] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam

Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *EMNLP*, 2020.

[269] Tao Wu, Ellie Ka-In Chio, Heng-Tze Cheng, Yu Du, Steffen Rendle, Dima Kuzmin, Ritesh Agarwal, Li Zhang, John Anderson, Sarvjeet Singh, Tushar Chandra, Ed H. Chi, Wen Li, Ankit Kumar, Xiang Ma, Alex Soares, Nitin Jindal, and Pei Cao. Zero-shot heterogeneous transfer learning from recommender systems to cold-start search retrieval. In *29th ACM International Conference on Information & Knowledge Management*, CIKM '20, 2020.

[270] Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering, 2023.

[271] Yuqing Xie, Wei Yang, Luchen Tan, Kun Xiong, Nicholas Jing Yuan, Baoxing Huai, Ming Li, and Jimmy Lin. Distant supervision for multi-stage fine-tuning in retrieval-based question answering. In *Web Conference 2020*, 2020.

[272] Yuqing Xie, Taesik Na, Xiao Xiao, Saurav Manchanda, Young Rao, Zhihong Xu, Guanghua Shu, Esther Vasiete, Tejaswi Tenneti, and Haixun Wang. An embedding-based grocery search model at instacart. *SIGIR eCom'22*, 2022.

[273] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *40th SIGIR*, SIGIR '17, 2017.

[274] Jun Xu and Hang Li. Adarank: A boosting algorithm for information retrieval. In *30th SIGIR*, SIGIR '07, 2007.

[275] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, 2020.

[276] Jheng-Hong Yang, Xueguang Ma, and Jimmy Lin. Sparsifying sparse representations for passage retrieval by top-k masking, 2021.

[277] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond, 2023.

[278] Matthew Y. R. Yang, Siwen Yang, and Jimmy Lin. Integration of text and geospatial search for hydrographic datasets using the lucene search library. In *22nd ACM/IEEE Joint Conference on Digital Libraries*, JCDL '22, 2022.

[279] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In *40th SIGIR*, SIGIR '17, 2017.

[280] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, 2018.

[281] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with BERTserini. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 2019.

[282] Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. Data augmentation for BERT fine-tuning in open-domain question answering. *CoRR*, 2019.

[283] Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *2015 EMNLP*, 2015.

[284] Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. Multilingual universal sentence encoder for semantic retrieval. In *58th ACL: System Demonstrations*, 2020.

[285] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeruIPS 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, 2019.

[286] Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *2013 NAACL*, 2013.

[287] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd ACL*, 1995.

[288] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. Pretrained transformers for text ranking: BERT and beyond. In *NAACL Tutorials*, 2021.

[289] Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *15th Conference on Computational Natural Language Learning*, 2011.

[290] Keisuke Yokoi and Akiko Aizawa. An approach to similarity search for mathematical expressions using MathML. In *DML (Digital Mathematics Library)*, 2009.

[291] Yaodong Yu, Heinrich Jiang, Dara Bahri, Hossein Mobahi, Seungyeon Kim, Ankit Singh Rawat, Andreas Veit, and Yi Ma. An empirical study of pre-trained vision models on out-of-distribution generalization. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[292] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Association for Computing Machinery*, CIKM '18, 2018.

[293] Richard Zanibbi and Dorothea Blostein. Recognition and retrieval of mathematical expressions. In *IJDAR*, 2012.

[294] Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Tompa. The Tangent search engine: Improved similarity metrics and scalability for math formula search. *arXiv:1507.06235*, 2015.

[295] Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. Ntcir-12 MathIR task overview. In *NTCIR*, 2016.

[296] Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Tompa. Multi-stage math formula search: Using appearance-based similarity metrics at scale. In *SIGIR*, 2016.

[297] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *ArXiv*, 2012.

[298] Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wenyun Yang. Towards personalized and semantic retrieval: An end-to-end solution for e-commerce search via embedding learning. In Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *43rd International ACM SIGIR, China, July 25-30, 2020*, 2020.

[299] Xinyu Zhang, Minghan Li, and Jimmy Lin. Improving out-of-distribution generalization of neural rerankers with contextualized late interaction, 2023.

[300] Yue Zhang, ChengCheng Hu, Yuqi Liu, Hui Fang, and Jimmy Lin. Learning to rank in the age of Muppets: Effectiveness–efficiency tradeoffs in multi-stage ranking. In *Second Workshop on Simple and Efficient Natural Language Processing*, 2021.

[301] Zhibo Zhang. Gpt2-ml: Gpt-2 for multiple languages, 2019.

[302] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, 2021.

[303] Wei Zhong and Jimmy Lin. Pya0: A python toolkit for accessible math-aware search. In *44th SIGIR*, 2021.

[304] Wei Zhong and Richard Zanibbi. Structural similarity search for formulas using leaf-root paths in operator subtrees. In Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra, editors, *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I*, Lecture Notes in Computer Science, 2019.

[305] Wei Zhong, Shaurya Rohatgi, Jian Wu, Lee Giles, and Richard Zanibbi. Accelerating substructure similarity search for formula retrieval. In *ECIR*, 2020.

[306] Wei Zhong, Xinyu Zhang, Ji Xin, Jimmy Lin, and Richard Zanibbi. Approach zero and Anserini at the CLEF-2021 ARQMath track: Applying substructure search and BM25 on operator tree path tokens. In *CLEF*, 2021.

[307] Wei Zhong, Yuqing Xie, and Jimmy Lin. Applying structural and dense semantic matching for the arqmath lab 2022, CLEF. In Guglielmo Faggioli, Nicola Ferro, Allan Hanbury, and Martin Potthast, editors, *Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, Bologna, Italy, September 5th - to - 8th, 2022*, CEUR Workshop Proceedings, 2022.

[308] Wei Zhong, Jheng-Hong Yang, Yuqing Xie, and Jimmy Lin. Evaluating token-level and passage-level dense retrieval models for math information retrieval. In *EMNLP 2022*, 2022.

[309] Wei Zhong, Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. One blade for one purpose: Advancing math information retrieval using hybrid search. In *46th SIGIR*, 2023.

[310] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, 2015.

[311] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015.

[312] Nivio Ziviani, Edleno Silva de Moura, Gonzalo Navarro, and Ricardo Baeza-Yates. Compression: A key for next-generation text retrieval systems. *Computer*, 2000.

[313] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 2006.

[314] A Zvonkin and L Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 2007.