Towards the Advancement of Open-Domain Textual Question Answering Methods

TOWARDS THE ADVANCEMENT OF OPEN-DOMAIN TEXTUAL QUESTION ANSWERING METHODS

by

Fan Luo

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF COMPUTER SCIENCE

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2022

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

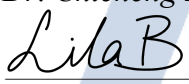
As members of the Dissertation Committee, we certify that we have read the dissertation prepared by: Fan Luo, titled: Towards the Advancement of Open-Domain Textual Question Answering Methods

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Dr. Mihai Surdeanu Date: Nov 30, 2022

Dr. Joshua Levine Date: Nov 29, 2022

Dr. Chicheng Zhang Date: Nov 29, 2022



Dr. Lila Bozgeyikli Date: Nov 29, 2022

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Date: Nov 30, 2022

Dr. Mihai Surdeanu
Dissertation Committee Chair
Computer Science Deparment

ACKNOWLEDGEMENTS

With what I learned from classes, professors, colleagues, and many previous research works, I stood on the shoulders of giants to get this work done.

Thank you to my supervisor, Mihai, for his guidance, patience and assistance all along the way. I want to thank my Committee members, Dr. Mihai Surdeanu, Dr. Joshua A Levine, Dr. Chicheng Zhang, and Dr. Lila Bozgeyikli for bearing with me and encouraging me along the way. I also would like to thank colleagues I worked with and learned from in the CluLab, including Rebecca Sharp, Marco A Valenzuela-Escárcega, Ajay Nagesh, Mithun Paul, Gus Hahn-Powell, Zhengzhong Liang, and Zheng Tang. I especially want to thank my parents for their encouragement and support throughout my studies. Finally, I am very grateful to the University of Arizona and the computer science department for providing me with the educational opportunity to gain knowledge and skills that prepare me for my future career goals.

DEDICATION

To my grandmother. Thanks for encouraging me to see this adventure through to the end.

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF TABLES

LIST OF TABLES – *Continued*

ACRONYMS

- Active Learning (AL)
- Approximated Nearest Neighbor (ANN)
- Average Precision (AP)
- Average Ranking (AR)
- Conditional Random Fields (CRF)
- Convolution Neural Networks (CNN)
- Cross-Encoder (CE)
- Decision Trees (DT)
- Deep Learning (DL)
- Deep Neural Network (DNN)
- Dense Passage Retrieval (DPR)
- Entailment-Aware Re-ranking (EAR)
- Gated Recurrent Units (GRU)
- Graph Attention Network (GAN)
- Graph Convolutional Network (GCN)
- Graph Neural Network (GNN)
- Hidden Markov Model (HMM)
- Inference Similarity (IS)
- Information Retrieval (IR)
- Knowledge Kase (KB)

- Language Model (LM)
- Long Short-Term Memory (LSTM)
- Machine Learning (ML)
- Masked Language Modeling (MLM)
- Mean Average Precision (MAP)
- Multi-Hop Question Answering (MHQA)
- National Institute of Standards and Technology (NIST)
- Natural Language Processing (NLP)
- Next Sentence Prediction (NSP)
- Open Domain Textual Question Answering (ODQA)
- Perturbation-based Active Learning (PAL)
- Pre-trained Language Model (PLM)
- Query Expansion (QE)
- Question Answering (QA)
- Question answering over Knowledge Bases (KBQA)
- Question Evidence Relevance (QER)
- Reading Comprehension (RC)
- Recurrent Neural Networks (RNN)
- Semantic Textual Similarity (STS)
- Semi-Supervised Learning (SSL)
- Semi-Supervised SVM (S3VM)
- Similarity Combination (SimCom)
- STEP: Steiner Tree-based Expansion Phrase identification
- Support Vector Machines (SVM)
- Term Frequency (TF) - Inverse Document Frequency (IDF)
- Text REtrieval Conferences (TREC)
- Visual Question Answering (VQA)

ABSTRACT

Open-Domain Textual Question Answering (ODQA) aims to answer a question in the form of natural language based on large-scale unstructured documents. While recent neural reading comprehension has advanced performance to new heights, there are several open research questions in this research area, including answering complex questions, providing human interpretable explanations, and minimizing labeling costs. In particular, we investigate Multi-Hop QA (MHQA), a well-known complex QA task, which requires combining multiple supporting context pieces scattered in documents to infer the correct answer. In this dissertation, we investigate the architecture of a typical ODQA system as well as the techniques adopted in each of the components, and present solutions that improve the systems to deal with the challenges aforementioned. First, we presented an unsupervised graph-based method called STEP [112] for identifying bridge phrases that connect the supporting evidence, which remains one of the challenging problems for MHQA. The identified bridge phrases are then used to expand the query, helping in increasing the relevance of evidence that has little lexical overlap or semantic relation with the question. In our second work, we presented a hybrid solution to improve the effectiveness of evidence re-ranking for MHQA, which scores and ranks the retrieved contexts to push the most useful information to the top before feeding to the answer prediction module. We applied off-the-shelf statistical and neural models to capture different dimensions of relevance, including exact matching, semantic textual similarity, and textual entailment, and effectively combined them to jointly rank candidate evidences. Lastly, to advance MHQA or ODQA in general, the underlying reader model needs to be improved. Due to the 'data-hungry' of the state-of-art reader model and the high annotation cost in practice, we applied a deep active learning technique and presented a perturbation-based acquisition function to select unlabeled training instances that are most informative to annotate in an effective way to reduce the annotation effort. All the approaches presented in the dissertation easily fit into a typical modern ODQA architecture.

CHAPTER 1

INTRODUCTION

My research focuses on the Open Domain Textual Question Answering (ODQA) task. Due to the increasing amount of information on the web, users expect direct responses to their complex queries. QA research attempts to enable machines to meet the growing information needs and combat information overflows, and provides a quantifiable way to test the reasoning ability of intelligent systems as it has long been the holy grail of machine intelligence [70, 142, 178, 141]; It is one of the most important but challenging tasks on the road towards natural language understanding [52]. QA systems put together Information Retrieval (IR) and Natural Language Processing (NLP) techniques to serve for information production. IR and NLP are two closely related fields that both seek to process and understand text data. IR systems often employ highly scalable statistics-based techniques to index and search large volumes of text efficiently, while NLP systems are used to analyze and interpret text data and to convert text to information that could be effectively harnessed by computer systems, raising information access to higher level to better serve information needs. IR and NLP have a long history of rivalry, but they are now beginning to work together more closely because they both rely heavily on embeddings produced by neural networks with the recent deep learning revolution. And QA systems utilize the advancement of both techniques to process requests formulated in natural language questions. Well-known applications of QA systems include virtual assistants (e.g.: chatbots, companion learning systems, custom service) to better serve information needs, and enhance the traditional search engines help users combat information overload and find information more efficiently.

There are different classifications of QA systems. For example, QA systems have been designed to address a diverse variety of question types, such as factoid, definition, how, why, list, multi-choice, and true-false questions. With respect to whether the system is supposed to deal with questions about a specific domain or not, QA systems can be classified as open-domain (ODQA) systems and closed-domain QA systems. ODQA deals with questions about nearly anything, over a large collection of unstructured documents from the web. From the perspective of the information source that is used to answer questions, QA systems can be classified as textual QA and knowledge-based QA.

Question answering over knowledge bases (KBQA) requires machines to answer natural language questions based on large-scale knowledge bases (KB). It usually relies on a large-scale structured knowledge base, datasets consist of a set of facts in the form of triples, such as Freebase [19], DBpedia [10], and Wikidata [186]. A structured knowledge base generally consists of many knowledge triples in the form of ⟨subject, relation, object⟩. For example, ⟨Ada Lovelace, birth-year, 1815⟩This triple can be used to answer text questions like "When was Ada Lovelace born?" or "Who was born in 1815?". In a knowledge triple, the subject is usually an entity and the object can be an entity or an attribute value. The relation describes the relationship between the subject and the object or indicates what attribute the object measures. In some literature, the relation is also referred to as the predicate. KBQA models should exploit the KB triples related to the questions and precisely identify the answers by reasoning through various KB relations. KBQA provides a direction for accessing and utilizing structured information on the Internet and thus has wide applications in the industry.

Besides structured knowledge, a tremendous amount of information is encoded in the text on the web. Textual QA systems are developed to answer questions asked in natural language (For example: *Which MSNBC commentator has been very critical of "race-norming?"*) with the precise answer (*George Will*). Given a user question, an information retrieval (IR) system then returns an ordered set of relevant documents[1] from a corpus. The IR systems (such as TF-IDF, dense retrieval model) usually map queries and documents to vectors and use the cosine similarity between the vectors to rank candidate documents.

[1] A document refers to whatever unit of text the system indexes and retrieves (web pages, scientific papers, news articles, or even shorter passages like collection paragraphs).

Then neural reading comprehension algorithms read these retrieved passages and draw an answer from spans of text. For each token p_i in the passage, compute two probabilities: $p_{start(}i)$ that p_i is the start of the answer span, and $p_{end(}i)$ that p_i is the end of the answer span. The answer is chosen as the span with the highest probability.

ODQA aims to answer questions from large data sources of diversified topics like Wikipedia or the web effectively harnesses a vast amount of textual articles available online to better serve information needs. A practical textual ODQA system is supposed to deal with questions about nearly everything with unstructured documents from the whole web. The problem of ODQA studied in this dissertation can be described as follows: Given a factoid question, such as "Who first voiced Meg on Family Guy?" or "Where was the 8th Dalai Lama born?", a system is required to answer it using a large corpus of diversified topics. More specifically, we assume the extractive QA setting, in which the answer is restricted to a span appearing in one or more passages in the corpus [86].

While being a longstanding problem in NLP and has been studied for decades [185], research on ODQA gained wide attention in recent years. Simmons et al. [170] developed the first QA system by matching dependency relations of the question and candidate answer statements. Text REtrieval Conferences (TREC) sponsored by the National Institute of Standards and Technology (NIST) initiated the Question-Answering Track in 1999, and provided benchmark datasets for fact questions over a large collection of documents, fostering extensive research on QA systems. QA systems obtained a lot of public attention when IBM Watson DeepQA [54] defeated two human champions at the popular live American television quiz show Jeopardy! (Figure 1.1). The DeepQA system was an ensemble of many methods and components. It took three years of intense research and development by a core team of about 20 researchers. With the rapid development of neural reading comprehension (RC), ODQA has been extensively studied recently [74, 33, 39, 117]. However, there is not yet a complete solution for ODQA. In this dissertation, we discuss the remaining open research questions (Section 1.2) and introduce three methods that advance the individual components of the typical architecture of an ODQA system (Section 1.1).

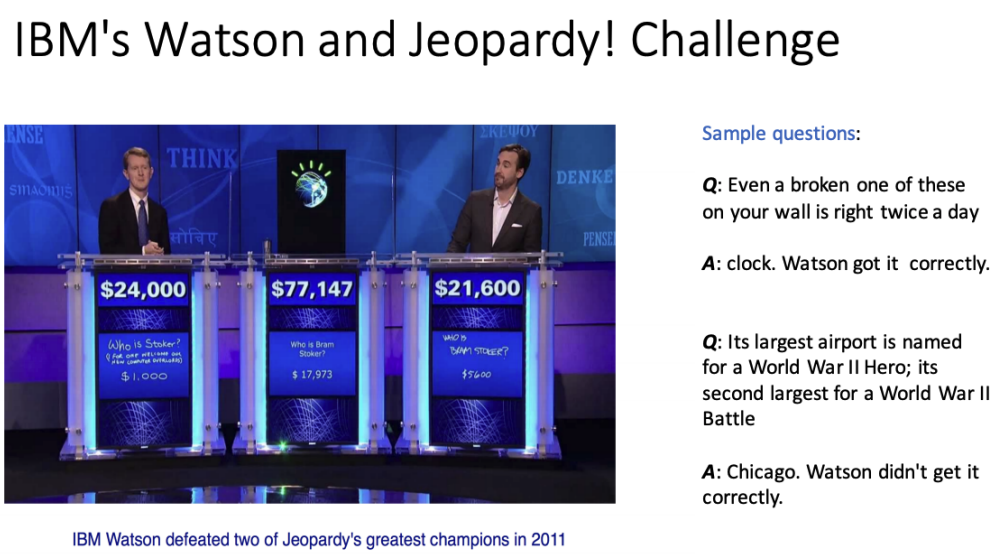


Figure 1.1: QA system obtained a lot of public attention when IBM Watson DeepQA [54] defeated two human champions at the popular American television quiz show Jeopardy!.

1.1. Typical Architecture of a Question-Answering System

For the architecture of textual QA systems, there are several versions and details vary from system to system. Jurafsky and Martin [83] described a three major phases system, namely question processing, document and passage retrieval, and answer extraction, as shown in Figure 1.2. The Question processing phase includes two major steps: query formulation and answer type detection. A typical QA system can be considered as consisting of (a) **Question analysis and reformulation** focus on the analysis of the question and reformulating questions before presenting it to the QA framework for processing and thereafter boosts the possibility of finding the appropriate response, (b) **Retrieval and re-ranking** retrieve potentially relevant documents and identify the supporting sentences within the restricted candidate documents, and c) **Answer extraction** to locate the text span that answers the question. Chen et al. [33] presented a two-stage retriever-reader framework. It contains a retriever component that narrows search space by retrieving a question-relevant set of documents from a large collection of documents, followed by a reader component extracting answers from the retrieved single document or a small collection of documents that are likely to be relevant.

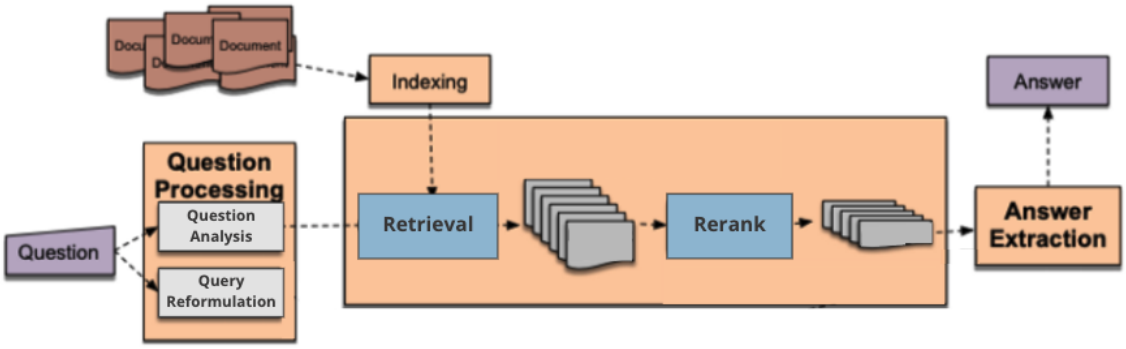


Figure 1.2: A typical architecture of Textual QA systems

1.2. Open Research Questions

To advance the ODQA systems, there are several open research questions in the field of to be tackled:

1.2.1. Answering Complex Question

Researchers have made substantial headway in answering simple questions (i.e., answering a question given a single document or paragraph) [141, 111, 192, 162, 3, 23]. Recent focus has been shifted to answering questions in more complex settings, i.e, complex questions. Complex questions cannot be answered by a single document, and require searching and integrating information from multiple pieces of evidence. Figure 1.3 shows an example of a complex question, "Which novel by the author of 'Armada' will be adapted as a feature film by Steven Spielberg?". It is necessary to first find the author of 'Armada', that is 'Ernest Cline' from one document. And then find out the novel by 'Ernest Cline' that will be adapted as a feature film by Steven Spielberg. Complex questions are difficult to answer since they require multi-hop reasoning and integration of information across multiple sources. However, even retrieving the documents that might contain the relevant information can be difficult. In general, questions that require integrating information from multiple sources are difficult for reading comprehension models to answer. This is because reading comprehension models are mostly trained to match questions to local contexts, and the assumption that all the information relevant to the answer is available is not always

realistic, especially for complex questions. Answering complex questions is still an ongoing research challenge. The ability to answer complex questions has various practical applications, and can significantly improve the utility of QA systems, such as improving the satisfaction of users when using web search systems via enhancing results that are not reachable in a simple setting, and helping chatbot assistants have more informative and smoother conversations between human users and agents.

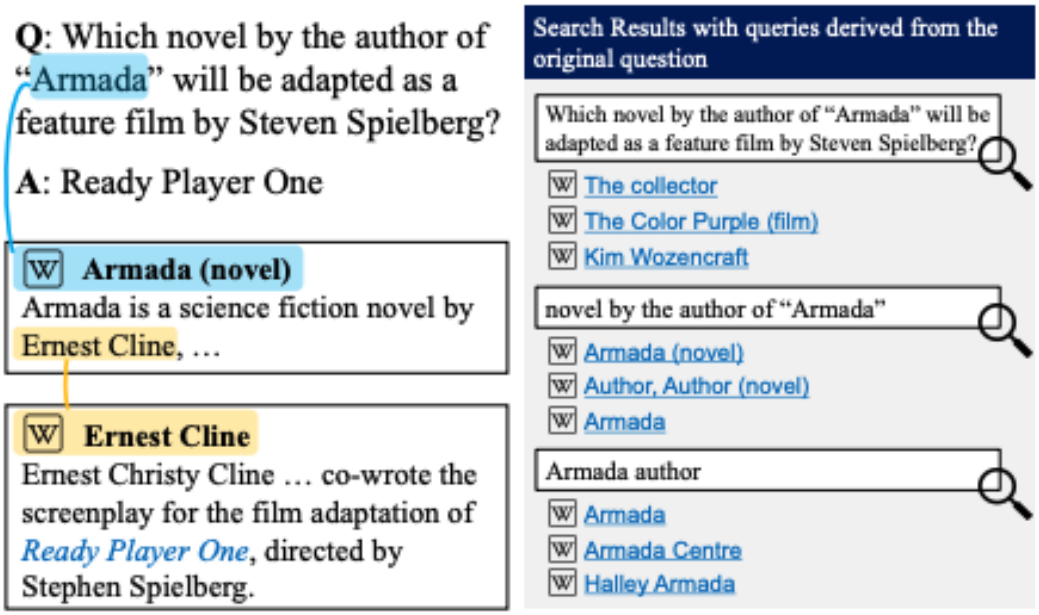


Figure 1.3: A example of Complex QA (Figure taken from [140])

Multi-Hop QA (MHQA) is one of the most researched tasks over recent years, which serves as an appropriate benchmark task for evaluating QA systems' ability to answer complex questions. The notion of 'multi-hops' refers to multiple pieces of information relevant to tackle the multi-hop question. A single independent piece of information, a sentence, a document, a field in a table, or an entity in a knowledge graph, depending on the specific dataset, is the 'hop'. MHQA is significantly more challenging than its single-hop counterpart and current models have limitations. To deal with MHQA, a QA model can either retrieve the information pieces from different sources and feed into a reading comprehension model, or train a fat reader model to perform reason over an aggregated long piece of text. The task of retrieving context from multiple hops away from the question

is challenging due to less lexical overlap and little semantic relation with the question and suffers from semantic drift. Besides, current models have difficulty combining multiple contexts for reasoning since automatic aggregation combines much semantically unrelated information. Above all, it is fair to say that the task of MHQA is still a long way from being solved.

1.2.2. Explainability

The task of explainable multi-hop QA requires the system to return the answer with evidence sentences by reasoning and gathering disjoint pieces of the reference texts [126]. Figure 1.4 illustrates the concept of explainable multi-hop QA with an example question. Despite rapid progress in language models for tasks like question answering and more, state-of-the-art systems still struggle to explain their decisions in natural language. Being able to provide evidence to explain the prediction the QA system made is vital for the wide adoption of a system, enabling human users to understand and verify by going through the step-by-step solution and verifying the correctness. This increases the model's accountability and interpretability to the end user. This is crucial for most high-stake applications such as legal, finance, and healthcare domains [59, 8, 157, 18, 5]. Different formats of explanation are possible, including sentences, phrases, reasoning chains or graphs, heatmaps, and attention scores. Techniques of explanation generation and identification [116, 76, 88] are proposed to help improve the explainability, such as pinpointing the supporting facts [203] from scattering information in context for sentence-level explainability, or providing an ordered entity chain for entity-level explainability. Nishida et al. [126] formally defined the explainable multi-hop QA task as below:

Def. 1. *Explainable Multi-hop QA*

Input: *Context C (multiple texts), Query Q (text)*

Output: *Answer String A (text), Evidence E (multiple texts)*

*The **Context** C is regarded as one connected text in the model. If the connected C is too long (e.g. over 2000 words), it is truncated. The **Query** Q is the query. The model answers Q with an answer string A. The **Answer String** A is a short span in C. **Evidence** E consists of the sentences in C and is required to answer Q.*

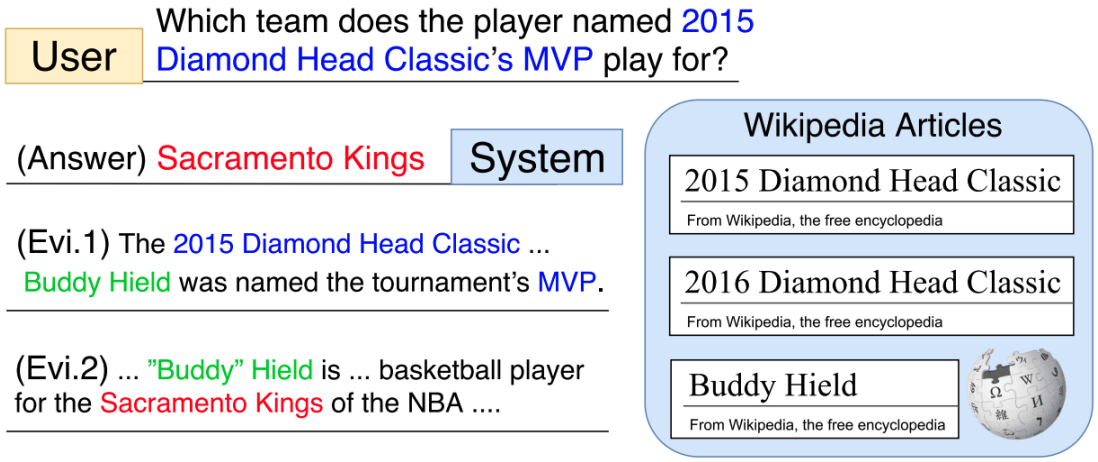


Figure 1.4: Concept of explainable multi-hop QA. Given a question and multiple textual sources, the system extracts evidence sentences from the sources and returns the answer and the evidence (Figure taken from [126]).

1.2.3. Lack of Annotation

Today's state-of-the-art QA models are deep neural networks with transformer-based architectures. Deep models are data-hungry, training such models requires a large amount of high-quality annotated training data to achieve a reasonably good performance. Lacking enough annotation is one of the central bottlenecks of training a capable QA model. A vast amount of labeled data is often not available in practice, in terms of both answers and explanations.

Recently, crowdsourcing is utilized for obtaining a large number of annotations [82, 142, 93, 40, 70] from the crowdworkers, the paid labor to perform annotation tasks. Amazon Mechanical Turk (MTurk) is such a crowdsourcing platform where requesters can post tasks, known as "Human Intelligence Tasks (HITs)", to coordinate the use of human intelligence from crowdworkers, known as "Turkers". Mechanical Turk is applied to a diverse set of natural language processing tasks. Kaisser and Lowe [85] used MTurk to create question-answer sentence pairs, Heilman and Smith [68] used MTurk to rank computer generated questions about provided texts, Gordon et al. [61] used MTurk to evaluate the quality and accuracy of automatically extracted common sense knowledge from news

and Wikipedia articles. For creating the widely used reading comprehension data SQuAD, Rajpurkar et al. [142] employed crowdworkers located in the United States or Canada to spend 4 minutes on every paragraph and answer up to 5 questions on the content of that paragraph, and paid them $9 per hour.

Even if crowdsourcing enables large-scale annotation possible and is a cheaper option than hiring full-time employees, it is not trivial. The instruction for the task needs to be carefully designed and clear enough to make sure the task is doable for the general crowdworkers. Although hired annotators are reasonably well educated, they are often non-experts and lack domain-specific knowledge, such as terminologies and technical terms. Additional demos or training sessions might also need to be provided for clarification. Besides, [29, 2] point out several causes of poor quality data from crowdsourcing, including 1) The task may be too complex or the instructions might not be clear enough to follow. 2) The financial incentives may be too low for Turkers to act conscientiously, and 3) Certain designs may allow them to simply randomly click instead of thinking about the task. 4) Inattentive or dishonest workers contribute statistical noise. 5) "bots," which are programs that mimic workers [25]. Thus, quality control checks are necessary. Besides the commonly used quality control measures that address different threats to validity, the task needs to be designed in a way that will detect cheating or make cheating obvious. Since the quality of annotations from different crowdworkers varies, whether and how to weight their annotations and combine their judgments also needs to be considered. To increase the quality of the data, it is common to hire domain experts to check the results of the crowdworkers and do the post-processing. And even with the efforts to clean up the annotations, we can expect many noisy examples in the dataset.

In all, manual annotations for large-scale QA datasets take a lot of human effort and are expensive in general. Crowdsourcing has been used for the creation of large-scale annotated datasets to support product development. However, in practice, the limited available labeling budgets often put a barrier to annotating large-scale QA datasets for many teams. Thus, it is vital to building a high-performance QA model with less number of annotated examples for practical application.

This dissertation presents potential solutions to advance components of the ODQA system to deal with the challenges aforementioned, which will be discussed in each chapter.

1.3. Terminology and Definitions

In this chapter, I explain the definitions of the terms used throughout the dissertation and the task. The terminology used throughout this dissertation generally follows the Natural Language Processing (NLP) literature.

1.3.1. Natural Language Processing

NLP is a research area that began in the 1950s as the intersection of computer science, artificial intelligence, and linguistics. The main goal of the NLP area is to enable communication between humans and computers using natural language, by building computational models that can analyze and understand natural languages, and learn the representation of human languages, so that computers can understand human language and respond in a way that is natural for humans. The automatic analysis of text requires a deep understanding of natural language by machines. Several NLP tasks break down the text to help the machines automatically analyze and understand human language. These tasks include tokenization, part-of-speech tagging, named entity recognition, parsing, and coreference resolution.

- **Part-of-speech tagging** Given a sentence, determine POS tags for each word (e.g., NOV, VERB, ADV, ADJ) More advanced forms of POS tagging include Hidden Markov Models (HMMs) and Recurrent Neural Networks (RNNs).
- **Named Entity Recognition** A named entity is a real-world object, such as persons, locations, organizations, products, etc., that can be denoted with a proper name.
- **Tokenization** is the process of breaking up text documents into individual words called tokens.
- **Relation Extraction** is a task of information extraction that seeks to identify and classify the relationships between entities in a text.

NLP has many real-world applications, such as spam detection, sentiment analysis, machine translation, automatic summarization, and question answering. NLP is also used in consumer applications like digital assistants and translation software, as well as in enterprise solutions that help streamline business operations and increase productivity .

1.3.2. Information Retrieval

IR is the process of finding documents or other information units that are relevant to a particular information need. To serve the ever-increasing information needs, it finds useful information from a large volume of information. It helps combat information overload that exceeds the human capacity to aggregate and interpret the fragments of knowledge published in millions of articles per year in research publications repositories, digital libraries, and electronic editions of newspapers, journals, and magazines. Modern information retrieval systems no longer simply return links to documents that hold the desired information. Instead, they often return sets of salient facts and locate the actual position of the required information extracted from the documents that users are likely to find helpful. For example, the information needed for the QA task is in the form of a question for which one is interested in finding an answer so that the IR system returns a set of candidate evidence sentences containing relevant information for a model to interpret the answer. This is done by automatically indexing the documents using keywords, and then searching for documents that contain those keywords. For example, one of the most popular methods is TF-IDF, which stands for ”term frequency-inverse document frequency.” It assumes that each document is described by a set of keywords, called index terms. An index term is a word from a document that is supposed to represent the document's semantics. Some index terms describe a given document better than others, so an index term is assigned a numerical weight to capture this effect. The index terms and their weights are used to determine which documents are retrieved for a given query [37]. However, this process can be difficult if the keywords are not chosen properly, or if the context of the keywords is not taken into account.

1.3.3. Machine Learning

Machine learning (ML) is a field of computer science that focuses on the creation of algorithms that can automatically recognize patterns in high-dimensional data and make predictions accordingly. ML methods can be broadly categorized as supervised, semi-supervised, and unsupervised. Supervised learning is the dominant technique, including

hidden Markov model (HMM), conditional random fields (CRF), deep learning (DL), support vector machines (SVM), decision trees (DT), and Naive Bayes, which train the model on data with labels. Semi-supervised learning, such as bootstrapping, uses a small amount of labeled data, and unsupervised learning does not use labeled data. Compared to systems based on handwritten rules, systems that are based on ML algorithms are more accurate and easier to maintain. In general, ML can automatically recognize patterns, focus on the most common cases and produce models that apply to unfamiliar input, while handwritten rules are difficult to write and are often not effective in handling unfamiliar or incorrect input. Additionally, ML systems can be improved simply by supplying more input data, while systems based on handwritten rules can only be made more accurate by increasing the complexity of the rules, which is much more difficult.

1.3.4. Deep Learning and Deep Neural Networks

DL is a type of machine learning algorithm that allows learning algorithms to learn from their mistakes and get better over time based on continuous optimization. It relies on a neural network abstraction that represents a computational graph of dependencies between the input and output, a loss function to learn a task, an optimization algorithm to find the right parameters, and numerical methods to compute derivatives. As the architecture is shown in Figure 1.5, Deep Neural Networks (DNNs) are computational models that have multiple layers of processing, allowing them to create multiple levels of abstraction to represent data. Thus, DNN is a very effective type of model for analyzing complex data types like images, videos, audio, and text. The term “Deep” in the deep learning methodology refers to the concept of multiple levels or stages through which data is processed for building a data-driven model [158]. A typical deep neural network contains multiple hidden layers including input and output layers. Examples of deep learning algorithms are convolution neural networks (CNN) and recurrent neural networks (RNN). While most traditional machine learning methods rely on hand-crafted feature extraction, in contrast, DL methods automatically extract features from data and generate the near-universal embedding representation. Also, traditional machine learning methods often have limited learning capabilities, and the increase in the amount of data cannot continue to increase the total amount of knowledge learned, DNNs can improve performance by accessing more data for “more

experience". When combined with a mass of training data and the large-scale computing power of computers, they adjust internal parameters and approximate the problem goal as much as possible. DL techniques have achieved state-of-the-art results in many complex tasks such as NLP, image recognition and classification, face recognition, time series prediction, and cybersecurity [66, 95, 143]. The deep learning revolution also has a major impact on NLP, and DNNs have become increasingly popular in natural language processing since they often produce better results than traditional ML methods.

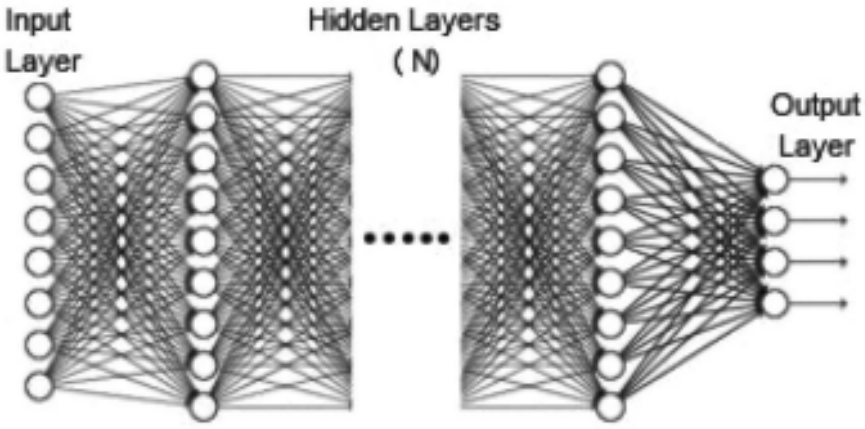


Figure 1.5: A general architecture of deep neural networks with multiple hidden layers (Figure taken from [158])

1.3.5. Transformer Model

The Transformer model is a neural network model presented in the paper *Attention Is All You Need* [184]. Figure 1.6 shows its architecture. The main innovations behind Transformers are positional encoding and self-attention. Transformer neural network architecture uses explicit positional encoding to learn word order, rather than relying on the sequential structure of traditional recurrent neural networks. While the "vanilla" attention mechanism allows the model to capture relations between words from two sequences, self-attention can be used to help the model understand a word in the context of the words around it. The transformer also performs multi-head attention, which allows the model to jointly capture different attentions from different subspaces, e.g., jointly attend to information that

might indicate both coreference and syntactic relations [172]. Compared with the RNN (or LSTM, GRU, etc.) network structure whose calculation is limited to sequential, Transformer is not limited by the order of the sequence so that it can be calculated in parallel, which makes it more efficient. This makes them ideal for training on large datasets.

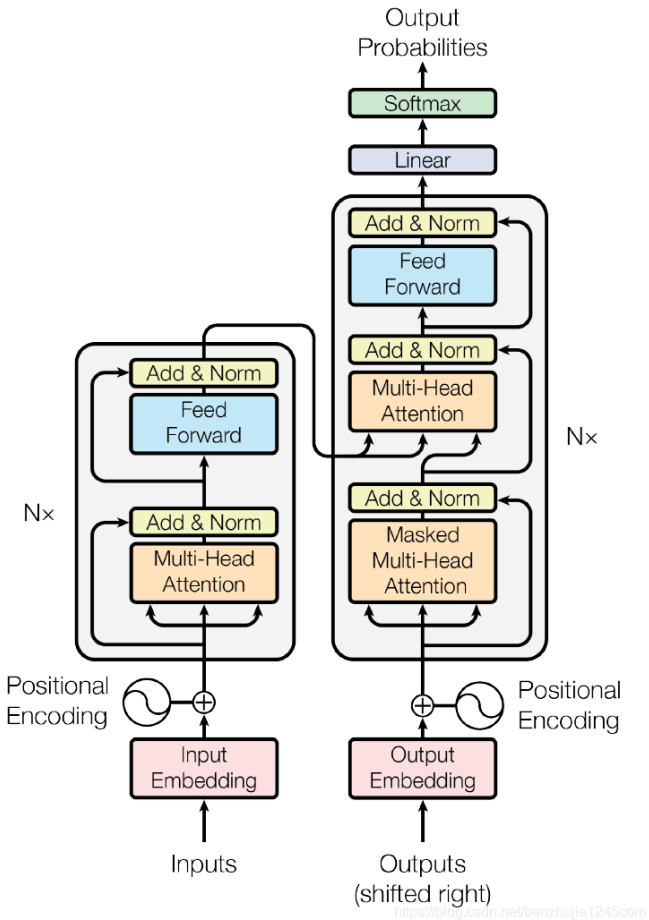


Figure 1.6: Transformer model architecture (Figure taken from [184])

1.3.6. Pre-trained Language Model

Language models are used to predict the probability of the occurrence of a word in a text sequence by analyzing text data. Traditionally, RNNs were used to train such models due to the sequential structure of language, but they are less effective due to slow training speed

and difficulty in convergence. With self-attention mechanisms and different pre-training objectives, variants of Transformers learn useful representations of language from unlabeled text corpora. The resulting Transformer-based pre-trained language model (PLM) understands natural language well and can be fine-tuned for a downstream task of primary interest, often leading to better generalization than training from scratch [63]. BERT (Bidirectional Encoder Representations from Transformers) [46] is one such model. BERT has been trained with Masked Language Modelling (MLM) and the Next Sentence Prediction (NSP) pre-training objective. Training these models can be computationally costly, but once these models are pre-trained, they can then be further fine-tuned for various downstream tasks to leverage the linguistic knowledge [134, 60, 179] encoded in a general language representation called embedding. Embedding is vector representations of text, projecting words or sequences of words as points in a vector space. The proximity in the embedding space indicates the similarity of meaning, based on words' co-occurrence with other words in training texts. Recent progress in PLM on large textual corpora led to a surge of improvements for downstream NLP tasks [135].

1.4. What makes answering complex questions difficult

The task of answering complex natural language questions requires aggregation of information from multiple supporting context pieces scattered in documents to infer the answer. Recent success on simple QA tasks has shifted the focus to more complex settings. Among these, MHQA is one of the most researched tasks over recent years [115]. Unlike single-hop QA, simply matching the semantics of the question and context is not sufficient to find the answer to a multi-hop question. Multi-hop QA faces two challenges. The first is it requires the QA system to be able to find the disjoint pieces of information as evidence and synthesize them to infer the correct answer. The retrieval of relevant information from multiple sources is challenging and vital, otherwise automatic aggregation would fail because it combines semantically unrelated facts leading to bad inferences. The subsequent supporting context beyond the first hop often consists of little lexical overlap or semantic relationship with the question, and thus fails to be retrieved. The second challenge is

explainability. It is even more challenging to generate an explanation for the answer prediction for multi-hop QA. The evidence used to reason is not necessarily located close to the answer, so it is difficult for users to verify the answer [94]. Above all, it is fair to say that despite the inspiring progress made so far, the task of MHQA is still a long way from being solved [115].

CHAPTER 2

Question Analysis and Reformulation

2.1. Chapter Overview

The first key component in QA systems is question processing, which consists of two main procedures: question analysis and reformulation. It is important to improve the efficiency of the following steps. Traditional methods of question analysis include morphological, syntactical, and semantic analysis, as well as focus recognition. The focus recognition method locates the word or sequence of words that pertain to the information requested [27]. Question reformulation transforms the question to improve its search relevance for effectual information retrieval, which narrows down the search space from millions of web pages to a small set of relevant documents or paragraphs. Query reformulation can be categorized into three main types: query reduction (removing tokens to loosen the search constraint), query expansion (adding tokens for more specificity), and query rewrite (abbreviation expansion, synonym expansion). Query expansion is a technique used to augment a query with additional terms that might be relevant to increase the likelihood of matching a relevant document from the corpus that uses terms not present in the original query. The added terms are typically synonyms and morphological variations of the terms in the query. For example, a query with the keyword “help” can be expanded to match documents containing the keywords “assist” and “support”. Information retrieval techniques mainly rely on lexical overlap or semantic matches between the questions with their candidate documents or paragraphs. Multi-hop QA usually requires finding more than one evidence document. The subsequent supporting context beyond the first hop could consist of little

lexical overlap or semantic relationship with the question, and thus fail to be retrieved. This poses a challenge to collect all relevant information for answering a multi-hop question.

This chapter summarizes work that appear in [112], in which we present a solution to bridge the information gap inherent between multi-hop questions and their answers. Our solution is to augment (i.e., expand) the original query with additional terms that bridge the supporting context pieces, so that the reformulated question increases the likelihood of matching the secondary-hop document from the corpus that uses the intermediate terms that not present in the original query. Considering the example question in Figure 2.1, it is necessary to first find the film that *has a score composed by Jerry Goldsmith*. And then find out the *name of the executive producer* of the film. Phrases such as *'Alien'* that connect the supporting evidences in multi-hop QA are called **bridge phrases**.

Question: What is the name of the executive producer of the film that has a score composed by Jerry Goldsmith?
Supporting Document1: Alien (soundtrack) The iconic, avant-garde score to the film "Alien" was composed by Jerry Goldsmith and is considered by some to be one of his best, most visceral scores. . . **Supporting Document2: Alien (film)** Alien is a 1979 science-fiction horror film ... Dan O'Bannon, drawing upon previous works of science fiction and horror, wrote the screenplay from a story he co-authored with Ronald Shusett ... Shusett was executive producer. . . .
Bridge Phrases: Alien
Answer: Ronald Shusett

Figure 2.1: An example multi-hop question from HotpotQA [203]. Information pieces from two supporting documents need to be connected to infer the answer. However, there is no lexical overlap between the question and the sentence containing the answer, which can be used by a general-purpose retriever to locate all the supporting information pieces, especially the answer *'Ronald Shusett'*. To answer this question, an interpretable question answering approach needs first identify the bridge phrase *'Alien'*, which is *'the film that has a score composed by Jerry Goldsmith',* and then answers the simpler question *"What is the name of the executive producer of the film Alien?"*

Identifying bridge phrases remains one of the challenging problems for multi-hop QA, especially when explainability is a concern. In this work, we presented an unsupervised method for the identification of bridge phrases in MHQA. Our method constructs a graph of noun phrases from the question and the available context and applies the Steiner tree algorithm to identify the minimal subgraph that connects all question phrases. Nodes in the subgraph that bridge loosely connected or disjoint subsets of question phrases due to low-strength semantic relations are extracted as bridge phrases. The identified bridge phrases are then used to expand the query based on the initial question, helping in increasing the relevance of evidence that has little lexical overlap or semantic relation with the question, helping in increasing the relevance of evidence that has little lexical overlap or semantic relation with the question. We call our method **STEP** (**S**teiner **T**ree-based **E**xpansion **P**hrase identification). Through an evaluation on HotpotQA [203], a popular dataset for multi-hop QA, we show that our method yields: (a) improved evidence retrieval, (b) improved QA performance when using the retrieved sentences; and (c) effective and faithful explanations when answers are provided.

2.2. Related Work

2.2.1. Question Decomposition

In general, two research directions have been explored to solve the multi-hop QA task. The first direction is to decompose the multi-hop questions into single-hop questions and directly apply the previous neural networks that are successful in single-hop QA tasks to multi-hop QA tasks.

[118, 79, 55, 133, 177] decompose multi-hop questions into multiple single-hop sub-questions and use a single-hop QA model to answer each of the questions.

Min et al. [118] presented DecompRC, a system that learns to break multi-hop questions into simpler, single-hop sub-questions using spans from the original question, assuming each sub-question can be formed from a multi-hop question by copying and lightly editing a key span from it. DecompRC segments the question text into several spans and then modifies the spans slightly depending on the reasoning type. Any single-hop QA model can be used for answering each single-hop question. The reasoning type is decided as the

one resulting in the maximum score. The method is tested with span-based questions and human-written questions and is found to be effective for both.

Jiang and Bansal [79] designed a self-assembling controller-based neural modular network made up of four modules (Find, Relocate, Compare, NoOp) to perform different types of reasoning. The NoOp module is also used to keep the network in its current state when no further action is needed. A controller RNN decomposes the multi-hop question into multiple single-hop sub-questions, dynamically infers which modules to use for each question, and outputs the sub-question.

Talmor and Berant [177] decomposed a complex question into a sequence of simple questions, each of which was submitted to a search engine. Once all answers are gathered, a final answer is then computed using symbolic operations such as union and intersection. The model uses a sequence-to-sequence architecture to map utterances to programs that indicate how to decompose the question and compose the retrieved answers. To obtain supervision for our model, they perform a noisy alignment from machine-generated questions to natural language questions and automatically generate noisy supervision for training.

Cao and Liu [30] presented two strategies for question answering, a coarse-grained decomposition (CGDe) strategy to divide a complex query into multiple single-hop simple queries and a fine-grained interaction (FGIn) strategy to represent each word in the document and help the model find the sentences needed to answer the question.

Most of these works involve training a decomposer model based on human labels in order to decompose the multi-hop questions into single-hop sub-questions.

2.2.2. Graph-based Multi-Hop QA

The other direction for solving MHQA is to model the task as performing inference over static or dynamic graphs. These graph-based methods mostly construct a graph with one or more types of nodes and edges added based on some lexical heuristics and applied Graph Neural Networks (GNNs), a Graph Attention Network (GAN), or a Graph Convolutional Network (GCN), to learn contextual encoding of the nodes. The representation of each node gets updated using the representation of each of its neighboring nodes so that nodes share information with each other.

Fang et al. [53] presented a model using a hierarchical graph network. They created a hierarchical graph with nodes on different levels of granularity (questions, paragraphs, sentences, entities), and bidirectional edges between the first-hop source sentences and the second-hop target paragraphs, as well as edges between paragraphs and sentences, sentences, and entities, question and entities, and all paragraphs. Using the RoBERTa encoding and a bi-attention and Bi-LSTM layer, the contextualized representations of the nodes are obtained. Then GAN is applied over the hierarchical graph, and the updated representations are merged with the original contextual representations.

Ding et al. [49] presented CogQA, which iteratively extracted entities and answer candidate spans using the BERT model for each hop and organized them as a cognitive graph, and then conducted the reasoning procedure over the graph with GNN, and collects clues (i.e., sentences containing the extracted entities) to better extract next-hop entities.

The "select, answer, and explain" (SAE) model proposed by Tu et al. [182] first filters out answer-unrelated documents, then selects answer-related documents and jointly predicts the answer and supporting sentences. The model is optimized with a multi-task learning objective on both the token level for answer prediction and sentence level for supporting sentence prediction, together with an attention-based interaction between these two tasks.

Qiu et al. [141] presented a Dynamically Fused Graph Network (DFGN): constructing a dynamic entity graph based on entity mentions in the query and documents, and iteratively generate and reason on a dynamic graph. The fusion process is then performed at each hop through the document tokens and entities, and the final resulting answer is obtained from document tokens.

Cao et al. [31] presented a Bi-directional Attention Entity Graph Convolutional Network (BAG) to learn query-aware node representations for entity graphs. Documents are transformed into a graph in which nodes are entities and edges are relationships between them. The graph is imported into GCNs to learn a relation-aware representation of nodes. Then bidirectional attention is applied to graphs and queries to generate a query-aware node representation, which will be used for the final prediction.

Coref-GRN from Dhingra et al. [47] applied a recurrent layer biased towards coreferent dependencies. The layer uses coreference annotations extracted from an external system to

connect entity mentions belonging to the same cluster. Specifically, given an input sequence and coreference clusters extracted from an external system, a term that depends on the hidden state of the coreferent antecedent of the current token (if it exists) is introduced in the update equations for Gated Recurrent Units (GRU). This way hidden states are propagated along coreference chains and the original sequence in parallel.

De Cao et al. [44] presented Entity-GCN, which constructed an entity graph, with three types of edges (1 DOC-BASED edges based on co-occurrence within the same document, 2) MATCH edges - if the pair of named entity mentions is identical, and 3) COREF edges between co-reference mentions of the same entity. A document encoder used to obtain representations of mentions in context, and a relational graph convolutional network that propagates information through the entity graph.

Song et al. [171] introduced a complex graph by considering three types of edges. The first type of edges connects the mentions of the same entity appearing across passages or further apart in the same passage. The second type of edges connect two mentions of different entities within a context window. The third type of edges are coreference-typed edges. Then they investigated GCN and GRN on this graph to perform evidence integration.

Unlike most multi-hop QA methods presented in the above-cited works, our method is unsupervised. Besides, instead of training a complex end-to-end black-box model, our method focuses on identifying the bridge phrases used by query expansion to boost the performance of evidence retrieval. We formulate our task as a subgraph identification problem, which can be solved with unsupervised methods. Moreover, our simple method enables easier interpretability via phrases over subgraph. Importantly, even though our method operates over a noun-phrase graph to identify bridge phrases, it does not require or expect the answer to be a noun phrase as many other graph-based methods do.

2.2.3. Query Expansion Techniques

Query expansion is a commonly used technique in the area of information retrieval as a solution for the term mismatch problem. It augments (i.e., expands) the original query with additional terms that might be relevant in terms to capture the actual intent before indexing so that the augmented query increases the likelihood of matching a relevant document from the corpus that uses terms not present in the original query. Existing works

mainly adopted query expansion (QE) as a solution to the "lexical chasm" [16] caused by the mismatch between the expression in questions and their answers. Most QE approaches expand the original query with terms obtained based on (a) a thesaurus, including synonyms, hypernyms, and acronyms [51, 120], Riezler et al. [148] expand the original query using Statistical Machine Translation to bridge the linguistic difference. (b) an ontology, a knowledge base describing the concepts, properties, instances, and structure of knowledge of a domain [62, 188, 4], or (c) additional features, such as query logs, web search information, or user intent mined from a community QA archive [11, 22, 196]. Several research works [119, 130, 17, 108] also utilize community-based features, search information, and social network to improve the retrieval performance in QA. Wu et al. [196] augmented queries with user intent mined from community QA archives, query logs, and web searches. Shekarpour et al. [166] expanded the original query on linked data using linguistic and semantic features extracted from WordNet and Linked Open Data.

In this work, we identify the bridge phrases that do not present in the question but are vital to connecting the question phrases to bridge the multi-hop information gap between the question and its answer.

2.3. The Algorithm

In this section, we provide an overview of STEP. Figure 2.2 overviews our approach; Table 2.1 shows a walk-through example.

2.3.1. Noun Phrase Extraction

We extract noun phrases from the input question and N candidate paragraphs (documents)[1]. These paragraphs may be provided with the dataset or retrieved by any off-the-shelf information retrieval component. To extract the noun phrases from the input question, we apply: (1) quotation extraction, i.e., extracting noun phrases between a pair of single or double quotes; (2) noun phrase grounding, using the titles from all the paragraphs as a simple encyclopedic resource and fuzzy match to tolerate typo and variations; For example, extract

[1]Paragraphs or documents depending on the input dataset. For HotpotQA, we use paragraph(s) and document(s) interchangeably, because HotpotQA only extracts the introductory paragraphs of documents from Wikipedia.

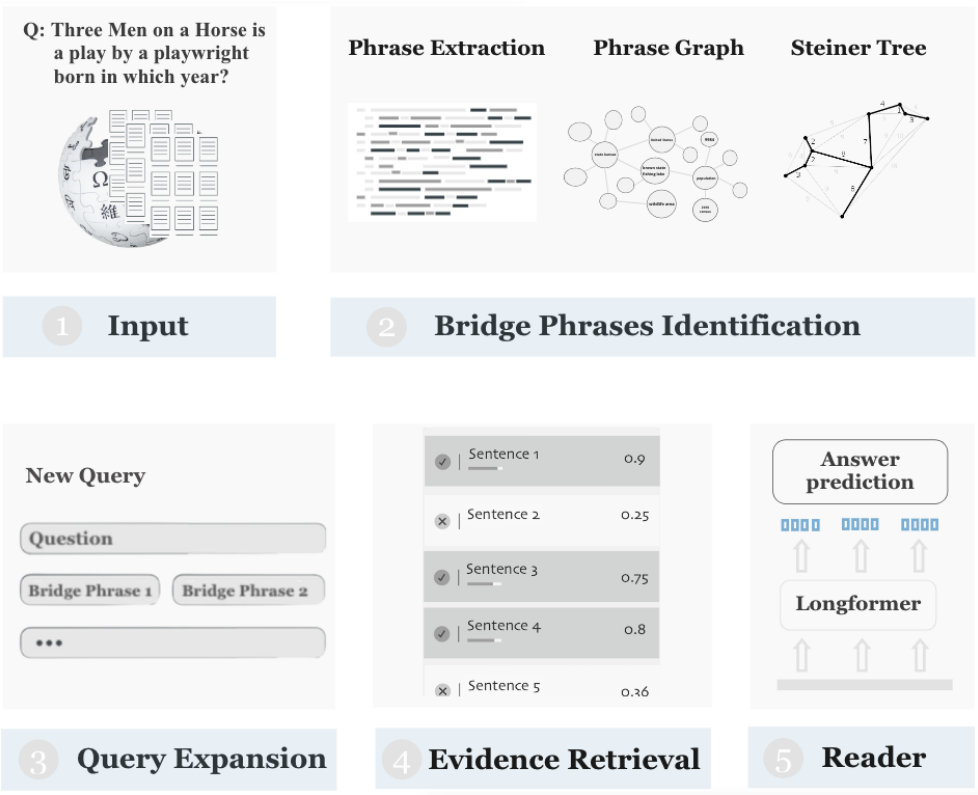


Figure 2.2: Overview of our approach. Given the input question and a corpus of candidate documents, we first extract noun phrases and construct the noun phrase graph. We then use the Steiner tree algorithm to identify the minimal subgraph that connects all question phrases. In this minimal subgraph, the Steiner points in the graph bridge subsets of question phrase nodes. Thus, the Steiner points are extracted as bridge phrases. These phrases are then appended to the original question to generate an augmented query. An off-shelf retrieval model uses the augmented query to rank the context sentences, and the top ones are then used as input to a Longformer reader for answer extraction.

the phrase "Three men on a horse" from the context by linking it to one of the title phrases for the question: "Three Men on a Horse is a play by a playwright born in which year?". (3) basic normalization, i.e., removing special punctuation and wh-words like *'what' and 'who'*; (4) named entity recognition, identifying named entities such as 'George Fancis Abbott' with an entity type of Person, and *'New York'* with a named entity type of *GPE*; and (5) noun chunks extraction, i.e., extracting a chunk of text containing a single noun word (*'songwriter'*) or a noun plus the words describing the noun (*'an organic compound'*). All

these noun phrases are normalized by: converting them to lowercase, removing articles, lemmatization, and discarding transparent question words such as *'time', 'place'*, etc.

We use the same process to extract noun phrases from every paragraph, with the exception that we apply coreference resolution before all the steps listed above, and use both document titles and extracted question phrases as the encyclopedias for noun phrase grounding.

In addition, we add the prefix *[P_i]*[2] to 'polyseme' named entities[3] (such as '*[P_8] January*'), and to noun chunks containing no named entity (such as '*[P_5]* video game') to distinguish the noun phrases that appear the same in the text but might refer to different things. For example, '*[P_1] January*' extracted from paragraph 1 is *2011 January*, while '*[P_3] January*' from paragraph 3 refers to *2017 January*. For all the components mentioned above, we used either SpaCy [72] or in-house components.

2.3.2. Noun-phrase Graph Construction

After all noun phrase mentions in the paragraphs have been identified, we create the initial noun phrase graph G: unique noun phrases are nodes in this graph; edges encode the co-occurrence or coreference relation between noun phrases. We model three types of co-occurrence: (a) SENT-SENT edges, which capture co-occurrences between noun phrases mentioned in the same sentence. For example, 'George Fancis Abbott' and 'playwriter' are both extracted from sentences of the second document in Table 2.1, so we add the SENT-SENT type edge between them, i.e., *'George Francis Abbott'* - *'[P_2]* playwriter'; (b) TITLE-SENT edges, which connect noun phrases occurring in the title of the document and its most similar noun phrase from each sentence or any single-word noun chunk in this document; For example, in Table 2.1, 'three men on a horse' is the title of the first document, and 'play' is a phrase from a sentence in the first document, so they form a TITLE-SENT edge. Connecting the single-word noun chunk to the title phrases directly indicates that it is mentioned in the paragraph, also to avoid too many direct connections to other nodes from the document via SENT-SENT edge; and (c) TITLE-TITLE edges, which connect noun phrases extracted from the same title. For instance, *'Tomb Raider'* -

[2]The prefix *[P_i]* indicates the phrase is extracted from the i_{th} paragraph.

[3]We consider a named entity with an entity type that is one of 'DATE', 'LANGUAGE', 'NORP', 'GPE', 'CARDINAL', 'PERCENT', 'LOC', 'QUANTITY','ORDINAL' as a 'polyseme' named entity.

'2013 video game', as 'Tomb Raider' and '2013 video game' are both from the title *'Tomb Raider (2013 video game)'*. Lastly, to better capture coreference, even though we already applied coreference resolution when extracting noun phrases, we also add: (d) coreference edges between inclusive phrases from the same paragraph (as *'Ronald Shusett' - 'Shusett'* in Figure 2.1 as they refer to the same person).

Note that the initial noun phrase graph G can be disjoint since some paragraphs do not share noun phrases with others. We then prune G by discarding disconnected graph components that do not contain a node that matches any of the question phrases.[4] If a question phrase is not in G, but a node in G fuzzily matches or partially matches the question noun phrase or a node matches with the question noun phrase when removing the prefix *[P_i]*, we add the question phrase as a new node and add an edge between the new question phrase node and the node matches with it. For example, we add edge 'Blake Shelton song' - 'Blake Shelton' for the question phrase 'Blake Shelton', and add edge '*[P_1] 'play' - 'play'* for the question phrase 'play' as in Table 2.1.

Instead of using entity linking to identify which phrases are similar to question phrases, which requires an external knowledge base (KB) for the mapping, we run fuzzy matching to find out the nodes that represent context noun phrases that are similar to the question phrases. We add edges between '[P1] play' to 'play' because it matches with the question noun phrase 'play' without the prefix [P1]. We then prune the graph by discarding disconnected graph components that do not contain a node matching with any of the question phrases. For example, the graph components grayed out on the left and right sides are removed. If there is more than one disjoint graph component containing at least one node that matches with a question phrase, we then add an additional edge between nodes that are the same without the prefix, such as '*[P_2] 'voice' - '[P_5] voice'*. We call the resulting graph the Relevant Graph (RG).

[4]Note this is different from simply discarding paragraphs not containing any question phrases, as one paragraphs can still be in a graph component that contains question phrase(s) even if itself does not.

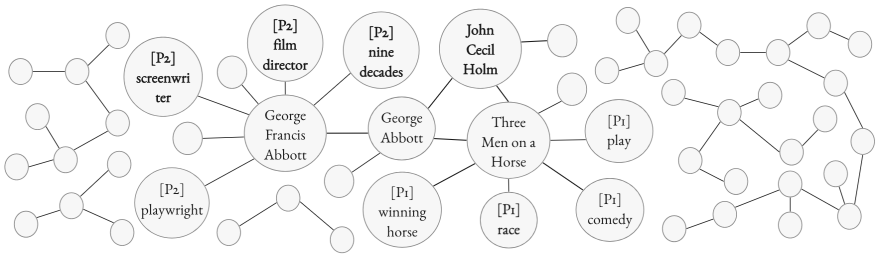
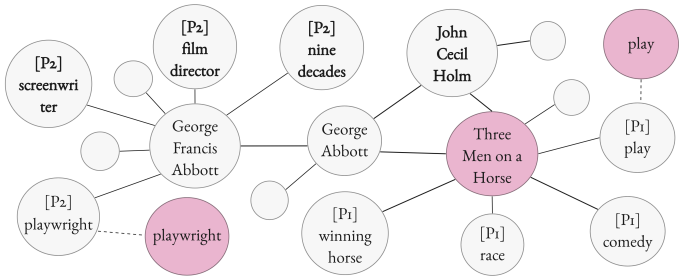
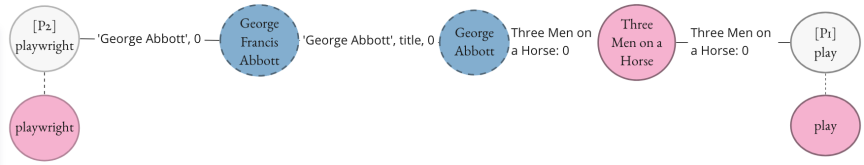
Input	**Question**: Three Men on a Horse is a *play* by a *playwright* born in which year? **Supporting Document 1: Three Men on a Horse** Three Men on a Horse is a *play* by George Abbott and John Cecil Holm. . . . **Supporting Document 2: George Abbott** George Francis Abbott (June 25, 1887 – January 31, 1995) was an American theater producer and director, *playwright*, screenwriter, and film director and producer whose career spanned nine decades. . . .
STEP$_1$: Noun phrase Extraction	**Question**: [Three Men on a Horse]$_G$ is a [play]$_C$ by a [playwright]$_C$ born in which year? ***Supporting Document 1:*** [***Three Men on a Horse***]$_T$ [Three Men on a Horse]$_G$ is a [play]$_C$ by [George Abbott]$_G$ and [John Cecil Holm]$_E$. . . . ***Supporting Document 2:*** [***George Abbott***]$_T$ [George Francis Abbott]$_E$ ([June 25, 1887 – January 31, 1995]$_E$) was an [American theater producer]$_C$ and [director]$_C$, [playwright]$_C$, [screenwriter]$_C$, and [film director]$_C$ and [producer]$_C$ whose [career]$_C$ spanned [nine decades]$_E$. . . .
STEP$_2$: Phrase graph construction	
STEP$_3$: Graph pruning & Question phrase identification	
STEP$_4$: Steiner tree computation	

Table 2.1: A walk-through example of our method. First, STEP extracts noun phrases[5]. Next, construct graph G using co-occurrence and coreference relations between noun phrases. Then, STEP prunes graph G by only keeping graph components that exactly or fuzzily match with at least one question phrase. The resulting graph is called RG. Question phrases in RG are the purple nodes. Lastly, STEP runs the Steiner tree algorithm over RG to identify the Steiner point *'George Abbott'* and *'George Francis Abbott'*, the nodes highlighted in blue, which are the correct bridge phrases. Due to limited space, we only show the source of each edge in the computed Steiner tree listed in the bottom row.

2.3.3. Steiner Tree Computation

We frame the identification of bridge phrases as a *minimum Steiner tree* problem [65][6], i.e., we use the Steiner algorithm [176] to compute the minimum spanning tree of the subgraph that covers all question phrases. Our implementation runs NetworkX's approximation minimum Steiner tree algorithm[7] over RG to identify the Steiner points (i.e., bridge phrases) that do not exist in the question but are needed to connect all question phrases in the graph. In the example question shown in Table 2.1, besides the two gray nodes which match with the question phrase nodes, the two nodes in blue 'George Abbott' and 'George Francis Abbott', are the Steiner points that connect the question phrase nodes. Thus, our algorithm identifies *'George Abbott'* and *'George Francis Abbott'* as the bridge phrases. We verified the two Steiner points are the correct bridge phrases for this question.

2.3.4. Query Expansion and Retrieval

To bridge the information gap inherent between multi-hop questions and their answers, we adopt the query expansion technique to augment the question with the bridge phrase(s) proposed by STEP. As an example, for the question in Table 2.3, we expand the question *"Three Men on a Horse is a play by a playwright born in which year?"* by appending the two bridge phrases 'George Abbott' and 'George Francis Abbott' to the end of the question, resulting in a new query *"Three Men on a Horse is a play by a playwright born in which year, George Abbott, George Francis Abbott"*. When using only the original question as the query, the second-hop evidence was less likely to be ranked at the top by either BM25 or cross-encoder. The expanded query assists a retrieval model to bring more and the most relevant sentences to the top. In particular, there is a higher chance for the second hop evidence *"George Francis Abbott (June 25, 1887 – January 31, 1995) was an American theater producer and director, playwright, screenwriter. . . "* to be retrieved. The top-ranked sentences are then fed into the downstream reader model for answer prediction.

Our method is agnostic to the retrieval algorithm used. , i.e., it can be coupled with both traditional retrieval methods such as BM25, and neural ones such as cross encoders.

[6] https://en.wikipedia.org/wiki/Steiner_tree_problem

[7] https://networkx.org/documentation/stable/_modules/networkx/algorithms/approximation/steinertree.html

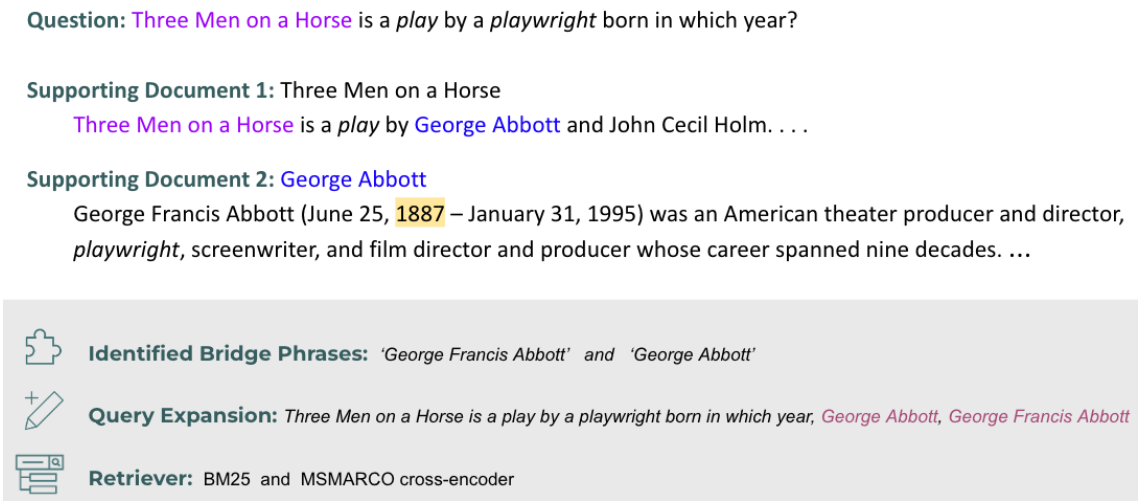


Figure 2.3: Query Expansion: In this example, we expand the question 'Three Men on a Horse is a play by a playwright born in which year?' by appending the two bridge phrases 'George Abbott', 'George Francis Abbott' to the end of the question, so that there is higher chance for the second hop evidence to be retrieved.

Theoretically, the retriever can be any method that ranks candidate sentences by comparing the semantic similarity against the input query. To evaluate the effectiveness of utilizing STEP as a query expansion method, we choose two off-the-self retrieval models. One is a widely used traditional IR model - BM25 [150]; the second is a transformer-based neural dense retrieval model - cross-encoder [145]. The cross-encoder pass text pair directly into the Transformer network and perform full attention over all tokens of the input pair to predict the similarity score that indicates the similarity of the input sentence pair. Specifically, we use a cross-encoder model pre-trained with passages from the MS MARCO dataset [124], a different dataset from HotpotQA. MS MARCO is a large-scale IR corpus, which has become a common starting point for building transformer-based ranking models before further fine-tuning on in-domain and task-specific data [204].

2.3.5. Answer Prediction

Identifying high-quality bridge phrases that connect the information gap between the questions and their answers is not only the key to the query expansion technique for retrieving more relevant evidences, which should result in improved performance of question

answering. To further evaluate the impact of STEP on retrieving relevant evidences, we feed the top-ranked set of sentences into a reader model for answer extraction. We use the Longformer model as the reader, since it is one of the open-sourced models at the top of the HotpotQA leaderboard. Following Beltagy et al. [15], we fine-tune the Longformer model in a multi-task way to predict evidences, answer spans, and question types (yes/no/span) jointly. For the prediction of relevant paragraphs and supporting fact sentences, a 2 layer feed-forward network is applied on top of the encoding of the sentence and paragraph title start tokens. The input to the Longformer with HotpotQA's training data by concatenating the input query and context sentences in the following format:

$$[CLS][Q]\ \texttt{Query}\ [/Q][SEP]\ [T]\texttt{title}_1\ [/T]\ sent_{11}\ [/S]\ sent_{12}[/S]\ldots[SEP]\ [T]\texttt{title}_2\ [/T]sent_{21}[/S]sent_{22}[/S]\ldots$$

where $[Q]$ and $[/Q]$ mark the start and end of the query, $[T]$ and $[/T]$ mark the start and end of the title of the current document, and $[/S]$ marks the end of a sentence. For answer prediction, a classification layer is applied over the $[CLS]$ token for question type classification, and a linear transformation is applied to each token for the prediction of the start and end of the answer span.

2.3.6. Post-hoc Reasoning

Neural models such as Longformer performing strongly on QA tasks are "black box" models. There is no easy way to interpret the answer provided. Even with the high accuracy of the predicted answer, when the predictions are wrong, there is no easy way to find out. Thus, there is a high demand for building tools and techniques that could provide human-readable interpretation of the predictions of these black boxes.

STEP can also serve as a post-hoc explanation module, for answers provided by other QA components. That is, the high-quality bridge phrases proposed by STEP can be used in conjunction with top-ranked evidences as focus points to help human users pinpoint the relevant information pieces more efficiently, instead of going over the ranked evidence candidate top-down. Post-hoc bridge phrase identification follows the same processes described above, the only difference being that we handle the provided answer similarly to the question. That is, we also ground to answer text during noun phrase identification, we maintain graph components containing nodes matching with at least a question phrase or

the answer, and, lastly, we also add the answer node to RG if it does not already exist. Then the Steiner tree algorithm is applied to find the minimal subgraph that connects all question phrase nodes and the answer node, in which the post-hoc bridge phrases are identified. After bridge phrases are identified, With the identified bridge phrases, we generate the top-ranked evidences by cross-encoder as the sentence-level explanation, using a similar query expansion technique. The differences are: first, we replace the *wh*-words in the question with the known answer; otherwise, we append the answer to the question. Next, we append the post-hoc bridge phrases to the query. In addition to enhance the evidences retrieved using the bridge phrases expanded query, the post-hoc bridge phrases also serve as focal points to help pinpoint the relevant information pieces more efficiently.

2.4. Experiments and Results

2.4.1. Dataset

To validate the proposed method, we ran a series of experiments using the HotpotQA dataset [203]. HotpotQA contains multi-hop questions created by human annotators using documents from Wikipedia as the information sources, instead of knowledge bases. Thus, the questions are not restricted by a fixed KB schema and cover more diverse topics. Importantly, the questions are designed to only be answerable by combining information from two documents and require to bridge documents via a concept or entity mentioned in both documents.

HotpotQA contains two question categories: *bridge-type questions*, in which an intermediate entity (i.e., the bridge phrase) is needed to be retrieved before inferring the answer; and *comparison-type questions*, where two entities are mentioned simultaneously and demand fact-checking or comparing the entities' properties from two different documents. Table **??** shows examples of each of the two question types, with the bridge phrases colored in blue (where applicable).

Given the focus of this work, we use solely the bridge questions in our evaluation.[8] We

[8]On average, comparison questions are easier to answer because the necessary information (i.e., the two entities to be compared) are present in the question.

Bridge Questions
Question: Bordan Tkachuk was the *CEO* of a company that provides what sort of products? **Answer**: IT products and services
Evidences: 1. ***Bordan Tkachuk***: Bordan Tkachuk is a British business executive, the former *CEO* of Viglen, also known from his appearances on the BBC-produced British version of "The Apprentice," interviewing for his boss Lord Sugar. 2.***Viglen***: Viglen Ltd provides IT products and services, including storage systems, servers, workstations and data/voice communications equipment and services.
Question: Three Men on a Horse is a play by a *playwright* born in which year ? **Answer**: 1887 **Evidences:** 1.***Three Men on a Horse***: Three Men on a Horse is a play by George Abbott and John Cecil Holm. 2.***George Abbott***: George Francis Abbott (June 25, 1887 – January 31, 1995) was an American theater producer and director, *playwright*, screenwriter, and film director and producer whose career spanned nine decades.
Comparison Questions
Question: Which American singer and songwriter has *a mezzo-soprano vocal range*, Tim Armstrong or Tori Amos? **Answer**: Tori Amos
Evidences: 1.***Tim Armstrong***: He is best known as the singer / guitarist for the punk rock band Rancid and hip hop/punk rock supergroup the Transplants. 2.***Tori Amos***: Tori Amos (born Myra Ellen Amos, August 22, 1963) is an American singer-songwriter, pianist and composer. She is a classically trained musician with *a mezzo-soprano vocal range*.
Question: What genre of music did Benjamin Burnley and Yannis Philippakis have in common? **Answer**: rock **Evidences:** 1.***Benjamin Burnley***: Benjamin Jackson "Ben" Burnley IV (born March 10, 1978) is an American musician, composer, and record producer, best known as the founder and frontman of the American rock band Breaking Benjamin. 2.***Yannis Philippakis***: Yannis Philippakis (born 23 April 1986) is the lead singer and guitarist of the British indie rock band Foals.

Table 2.2: Examples of bridge and comparison questions from HotpotQA. Bridge phrases are colored in blue (for the bridge question). The bridge phrases help retrieve the second-hop evidence that has low lexical similarity with the question. In contrast, there is no need of a bridge phrase for the comparison questions. The evidence documents contain one of the entities mentioned in the question, so that they can be easily retrieved with simple lexical match.

conduct the evaluation of our unsupervised method STEP on the 5918 bridge-type questions out of the 7,405 examples from the development partition of HotpotQA dataset in the distractor setting. Each question in HotpotQA is supported by two documents, and provided with ground-truth supporting sentences, which enables us to evaluate our approach for both evidence retrieval and the actual QA task.

2.4.2. Experiments

We demonstrate that STEP identifies bridge phrases that help catch more relevant information for answering multi-hop questions and providing post-hoc reasoning explanations with the following experiments as shown in Figure 2.4:

Figure 2.4: To evaluate the effectiveness of our method, we run four experiments on around 6,000 bridge-type questions from the HopotQA dataset. The first experiment compares the evidence retrieval performance with and without using our method for query expansion. The second experiment compares the answer prediction performance using the retrieved context from the first experiment. In the third experiment, we manually evaluate the identified bridge phrases on a sample of questions. The last experiment evaluates the capacity of our method to provide post-hoc explanations for the situations when an answer exists or is provided by other methods.

Experiment 1 (evidence retrieval): In this experiment, we evaluate our method as query expansion for evidence retrieval, i.e., we expand the original question with the bridge phrase(s) identified by STEP. We couple our query expansion strategy with both traditional and neural information retrieval algorithms for evidence retrieval.

Experiment 2 (question answering): We use the outputs of the above evidence retrieval

components as context for QA and evaluate the impact of this improved context on answer extraction.

Experiment 3 (bridge phrases): To account for the possibility that our bridge phrases yield better evidence sentences and/or answers by mistake, we manually evaluate the bridge phrases generated by our method on a sample of the questions.

Experiment 4 (explanations): We evaluate the capacity of our method to provide post-hoc explanations for the situations when an answer exists or is provided by another method. In this experiment, we manually evaluate the quality of the explanations provided by our method for a sample of questions using the gold answers from the dataset.

2.4.3. Results

Experiment 1: Evidence Retrieval

Note that STEP is agnostic to the downstream evidence retrieval component. STEP serves as a query expansion component, where the original HotpotQA question is expanded with the bridge phrases proposed by STEP. We test this query expansion with both a traditional information retrieval model (BM25[9] [181]) and a transformer-based neural retrieval model (cross-encoder[10] [145]).

	P@2	P@3	Avg Prec	R@2	R@3	R@5	R@10	R@20
Cross Encoder	0.59	0.47	0.64	0.50	0.59	0.69	0.81	0.92
Cross Encoder w/ STEP	**0.64**	**0.51**	**0.69**	**0.54**	**0.65**	**0.75**	**0.85**	**0.94**
BM25	0.55	0.44	0.60	0.46	0.55	0.65	0.78	0.90
BM25 w/ STEP	0.60	0.49	0.66	0.51	0.62	0.72	0.84	0.93

Table 2.3: Evidence retrieval performance (precision@k and recall@k) for all bridge questions in the development partition of HotpotQA, when STEP is coupled with a traditional IR component (BM25), or with a neural one (cross-encoder).

[9] https://pypi.org/project/rank-bm25/

[10] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2

Table 2.3 lists evidence retrieval performance for both strategies when retrieving $k \epsilon \{2, 3, 5, 10, 20\}$ sentences on the bridge questions from the development partition of the HotpotQA dataset. The same results are summarized in Figure 2.5.

These results highlight several observations. First, the neural retriever performs consistently better than BM25. For example, when only using the question as the query, the average precision of the cross-encoder is 0.64 while the average precision of BM25 is 0.6. This is not a surprise: these multi-hop questions exhibit a large "lexical chasm" [16], which is better bridged by neural methods. Second, when our method STEP is coupled with either BM25 or with the cross-encoder, STEP improves evidence retrieval performance in all settings, as shown in Figure 2.5. For example, after using STEP for query expansion, the average precision of the cross-encoder increases from 0.64 to 0.69. Despite neural retrievers improved capacity to bridge the lexical chasm between multi-hop questions and answers. STEP is capable of retrieving additional information and further bridging the information gap that is not modeled by neural retrievers.

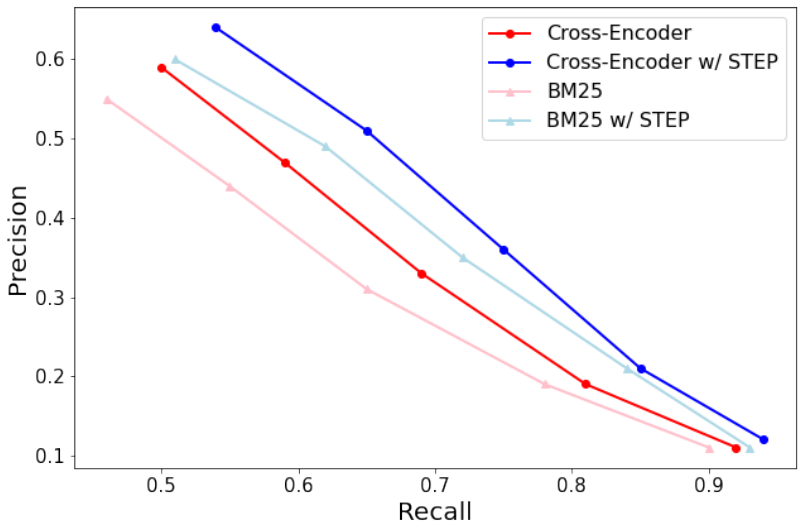


Figure 2.5: Precision-recall curve for evidence retrieval, when STEP is coupled with a traditional information retrieval component (BM25), or with a neural one (cross-encoder).

Experiment 2: Question Answering

To further evaluate the impact of the improved context on question-answering performance, we concatenate the top-ranked sentences from the cross-encoder in the previous experiment with the question as the input to the Longformer reader model [15] for answer extraction. That is, in this experiment, we provide the question along with the retrieved $k\epsilon\{5, 10, 20\}$ sentences by the cross encoder with STEP from the previous step as context to the Longformer model as input. With metrics exact match, F1 score, precision, and recall, we compare the question answering performance with using context sentences retrieved by cross-encoder with STEP expanded query against using context obtained from multiple baseline and ceiling strategies.

The two baseline strategies used are:

Random: This baseline uses a set of k sentences randomly selected from the HotpotQA documents associated with the corresponding question.

Question-only: This strong baseline relies on the context consisting of top-ranked evidence sentences from the neural retriever using solely the original HotpotQA question as a query.

The two ceiling strategies used are:

SF only: This strategy uses the ground-truth supporting sentences as the context.

Oracle: This ceiling strategy uses queries expanded with oracle bridge phrases extracted directly from the ground-truth supporting sentences, based on several heuristics that identify bridge phrases as (1) a phrase shared by two supporting sentences; (2) a phrase extracted from a supporting sentence that is also a title phrase of another supporting document; (3) a phrase extracted from a supporting sentence that has an inclusive relationship with a phrase from another supporting sentence.

Table 2.4 lists overall QA results for the bridge-type questions in the development partition of HotpotQA. These results indicate that the Longformer using context sentences retrieved with STEP expanded query consistently obtains better QA performance than the strong baseline that uses sentences retrieved by the cross-encoder retriever using just

the original question (the "Question-only" configurations). Further, our best configuration (cross-encoder with STEP) outperforms the Random baseline considerably, and approaches the performance of the "Oracle" setting, i.e., when bridge phrases are extracted directly from the correct supporting sentences. This further suggests that STEP identifies new and useful information that is missed by transformer networks.

	Exact Match	F1	Precision	Recall
Random 5	0.11	0.18	0.19	0.19
Random 10	0.19	0.29	0.3	0.29
Random 20	0.3	0.44	0.46	0.45
Question-only Top 5	0.33	0.48	0.5	0.49
Question-only Top 10	0.41	0.57	0.6	0.58
Question-only Top 20	0.49	0.67	0.7	0.69
SF only	**0.56**	**0.77**	**0.79**	**0.79**
Oracle Top 5	0.49	0.67	0.7	0.68
Oracle Top 10	0.52	0.71	0.74	0.72
Oracle Top 20	0.54	0.73	0.76	0.75
STEP Top 5	0.4	0.55	0.58	0.57
STEP Top 10	0.45	0.62	0.65	0.64
STEP Top 20	**0.51**	**0.69**	**0.72**	**0.71**

Table 2.4: Exact match, F1, precision, and recall scores for QA performance using Longformer for answer extraction, over contexts retrieved with various strategies.

On HotpotQA, we demonstrate an improved question answering accuracy achieved by adopting STEP as the query expansion technique. The results reinforce our assumption that STEP is capable of identifying the bridge phrases that enrich the original query representation, and thus identify the sequence of documents that provide relevant information to answer the multi-hop question more accurately. As aforementioned, STEP is agnostic to downstream models and does not depend on a corpus-dependent graph structure, which opens the possibility of applying STEP to any downstream model easily and broadly crosses different domains and settings without a change to the model itself. Note that Longformer already has a global attention mechanism for handling long text, which potentially mitigates the benefits of STEP.

Experiment 3: Bridge Phrases

The above experiments demonstrated that STEP identifies expansion terms that augment the question to help in increasing the relevance of evidences that have low lexical or semantic overlap with the initial question in general. Given the opacity of neural methods, it is possible that STEP improves evidence retrieval and answer extraction by chance. That is, STEP may generate bridge phrases that are incorrect, but they yield sentences that are useful for the downstream components. To take a closer look at the quality of the bridge phrases proposed by STEP, and to investigate whether they do assist to connect the information gap between the questions and their answers, we randomly selected 100 questions from the dataset, and asked two human annotators to annotate the extracted bridge phrases as: correct (if they bridge the necessary connection between question and answer), incorrect (if they do not), and partially correct (if only some of correct bridge phrases are identified).

According to the human annotators, the average accuracy of the bridge phrases generated by STEP is 76.3%. The Kappa inter-annotator agreement was 46%, which is ranked as moderate agreement. We consider this agreement respectable given the complexity and the ambiguity of the task (i.e., there may be multiple ways to answer a given question). Table 2.5 lists a few examples of bridge phrases extracted by STEP.

Experiment 4: Post-hoc Explanations

Lastly, we evaluate STEP's capacity to provide auxiliary *post-hoc* explanations to interpret answers provided by an external component (be it human or machine). Table 2.6 shows an example of a post-hoc explanation generated by coupling STEP with the cross-encoder. Given the answer *'Murray Hill'*, one will fast locate the candidate evidence #1, and find the underlined bridge phrase *'Bell Labs'*, and gap the reasoning by locating another evidence that would confirm *'Bell Labs'* is *"the American research and scientific development company where Ravi Sethi worked as computer scientist"*. This is done efficiently by looking for the candidate evidence that connects *'Bell Labs'* with one of the rest question phrases *'the American research and scientific development company'*, *'Ravi Sethi'*, *'computer scientist'*, and thus locates candidate evidence #4 and #2 immediately, which closes the reasoning loop. In this example, one would skip the candidate evidence #3 because it does not

	Question	Answer	Bridge Phrases (STEP)	Annotations
(1)	Ralph Hefferline was a psychology professor at a university that is located in what city?	New York City	Columbia University	(Correct, Correct)
(2)	The Vermont Catamounts men's soccer team currently competes in a conference that was formerly known as what from 1988 to 1996?	North Atlantic Conference	America East Conference	(Correct, Correct)
(3)	According to the 2001 census, what was the population of the city in which Kirton End is located?	35,124	Boston	(Correct, Partial)
(4)	When was the Western Germanic language spoken from which the small settlement situated on the river Leda opposite Leer derives its name?	between the 8th and 16th centuries	Old Frisian English language	(Correct, Partial)
(5)	Ellie Goulding worked with what other writers on her third studio album, Delirium?	Max Martin, Savan Kotecha and Ilya Salmanzadeh	–	(Incorrect, Incorrect)
(6)	This Celtic ruler who was born in AD 43 ruled southeastern Britain prior to conquest by which empire?	Roman	Roman conquest of Britain	(Correct, Incorrect)

Table 2.5: Examples of bridge phrases STEP identified, and human evaluation from two annotators. Due to limited space, supporting facts the annotators used to evaluate the bridge phrases are not listed here.

contain any connection between the noun phrases of interest. Note that during this post-hoc reasoning, even though the candidates ranked higher are more likely to be correct evidence after the reranking with the bridge phrase augmented query, one does not have to follow the ranking sequence to check the possible evidence in the top-down order. In this example, one would skip the candidate #3 because it does not contain any connection between the noun phrases of interest.

For the evaluation of STEP's capacity to provide auxiliary explanations to interpret provided answers, we provided 50 random sampled questions as in Table 2.6 with answers and the top 10 evidences ranked by cross-encoder using STEP for query expansion and asked the annotators to evaluate the quality of the generated explanations. The overall

Question: In which city are the headquarters of the American research and scientific development company where Ravi Sethi worked as computer scientist located?
Answer: Murray Hill
Top ranked evidence candidates:
1. ***Bell Labs***: Its headquarters are located in Murray Hill, New Jersey, in addition to other laboratories around the rest of the United States and in other countries.
2.***Ravi Sethi***: Ravi Sethi (born 1947) is an Indian computer scientist retired from Bell Labs and president of Avaya Labs Research.
3. ***Ravi Sethi***: He also serves as a member of the National Science Foundation's Computer and Information Science and Engineering (CISE) Advisory Committee.
4. ***Bell Labs***: Nokia Bell Labs (formerly named AT&T Bell Laboratories, Bell Telephone Laboratories and Bell Labs) is an American research and scientific development company, owned by Finnish company Nokia.
5. ***Ravi Sethi***: He is best known as one of three authors of the classic computer science textbook "", also known as the "Dragon Book".

Table 2.6: Examples of post-hoc explanation. Phrases in the Steiner tree computed by STEP are underlined, from which bridge phrases that do not appear in question nor answer are marked in blue. Candidate evidences ranked at positions 1, 2, and 4 are the correct supporting facts.

results are shown in Figure 2.6. The annotators report 44.5 out of the 50 questions were provided with high-quality explanations, and STEP identified the correct post-hoc bridge phrases for 48 questions. Considering the re-ranker we used is a model trained in a zero-short manner, it is likely that an even higher quality of explanation would be generated when using a more powerful ranker trained in-domain.

2.4.4. Error Analysis

We manually inspected examples where the two annotators disagreed, and found that disagreement happens in the following cases: (a) lexical ambiguity or overlap. As an example,

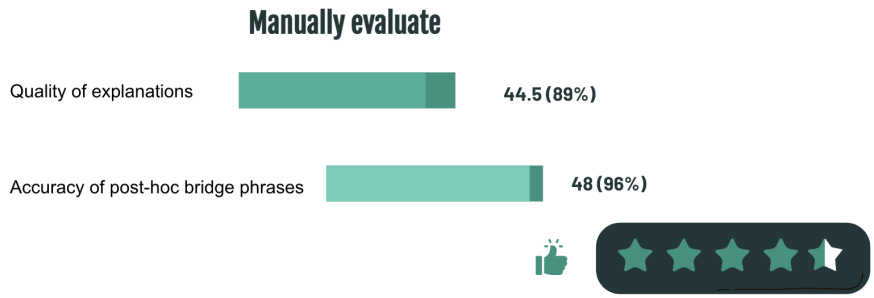


Figure 2.6: Post-hoc explanations manual evaluation results

the correct bridge phrase for the question *"According to the 2001 census, what was the population of the city in which Kirton End is located?"* (i.e., (3) in Table 2.5) is *'Boston Lincolnshire'*, but STEP extracts *'Boston'*, which lexically overlaps with the correct phrase; One annotator marked as correct and another annotator marked as partially correct. (b) there is more than one bridge phrase. For example, question (4) in Table 2.5 has two bridge phrases: *'Old Frisian'* (i.e., *'the Western Germanic language'*) and *'Kloster Muhde'* (i.e., *'the small settlement'*). While *'Old Frisian'*, the major bridge phrase that connects all the question phrases, has been identified correctly, one annotator marked as correct, and the other marked as partially correct for missing *'Kloster Muhde'*.

We also inspected the erroneous cases and found two common causes of error: (a) STEP found the answer instead of the bridge phrase, as in question (6) in Table 2.5, while the correct bridge phrase is *'Catuvellauni'*; and (b) STEP does not identify a bridge phrase because all the mentioned phrases in the question are well-connected. For example, for question (5) in Table 2.5, *'On My Mind'* is the correct bridge phrase (i.e., *the song in Ellie Goulding's third studio album that was written by her and Max Martin, Savan Kotecha and Ilya Salmanzadeh*). However, the question phrases *'third studio album'*, *'writers'*, *'Delirium'* are all connected to *'Ellie Goulding'* by co-occurrence. There is no need of a Steiner point to bridge them. Further, even though we evaluate on only the bridge-type of questions from HotpotQA, some comparison questions are labeled as bridge questions in the dataset and leaked into our evaluation. For example, *"Both Bishop Carroll Catholic High School and Kapaun Mt. Carmel Catholic High School are located in which city in Kansas?"*

Overall, this manual evaluation showed that STEP is capable of identifying high-quality

bridge phrases that connect the information gap between the question and the relevant context for most questions.

2.5. Summary

We presented an unsupervised approach for the identification of bridge phrases in MHQA. Our method constructs a graph of noun phrases from the question and the available context and applies the Steiner tree algorithm to identify the minimal subgraph that connects all question phrases. We extract as bridge phrases nodes in this graph that bridge loosely-connected or disjoint subsets of question phrases. Using the HotpotQA dataset, we demonstrate that our method yields improved results in all these scenarios, for multiple types of downstream components, including (a) improved evidence retrieval, (b) improved QA performance when using the retrieved sentences; and (c) effective and faithful explanations when answers are provided. In all, the takeaway messages are summarized in Figure 2.7.

Figure 2.7: We presented an unsupervised method to find bridge phrases with the Steiner tree algorithm The identified bridge phrases are then used to expand the query. The experimental results show that the STEP approach is effective. And our method generates post-hoc explanations for provided answers.

CHAPTER 3

Hybrid Evidence Retrieval and Reranking

3.1. Chapter Overview

Answering natural language questions by querying an open-domain QA system that searches over information on a large collection of unstructured documents from the whole web is highly demanded by real-life applications. The content on the web is growing at incredible rates. In these billions of pages, the answer to the question is buried. Open-domain QA systems commonly reduce the scope of information with a retrieve-and-re-rank framework as Figure 3.1 to collect the most relevant and necessary information pieces for answering the question as evidences (or supporting facts, or explanations). Then a reader module reads the refined information to infer an answer.

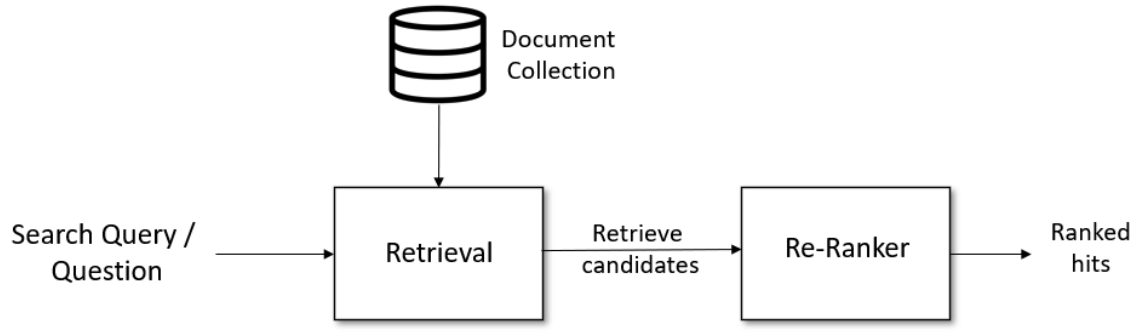


Figure 3.1: Retrieve-and-rerank framework to collect relevant information for a given question (Figure taken from [145])

Given a question, the retrieve-and-re-rank framework [194, 137, 199, 125, 34] first retrieves a large subset of relevant documents which are potentially relevant for the query. The retriever has to be efficient in processing large document collections with millions of entries to decide whether a document is relevant or not. Besides, in this stage, high-recall retrieval strategy is preferred to avoid over-filter useful information that sacrifices the answer's correctness due to missing necessary supporting context. However, over-generate candidates and the high number of candidates (i.e., up to 200) limits the practical usefulness of the systems to an end user [173]. The high-recall retriever often includes many non-relevant content for the reader module to process. Combining semantically unrelated facts leads to bad inferences. The relatively large set of candidates is also not useful in practice as the explanation for a human user to read, since one would have to sort through the explanations to judge the machine's answer. In addition, long text processing is challenging for state-of-art transformer-based reader models. This is because transformers use multi-head self-attention learned in parallel and merged together using concatenation and linear operations. As a result, the memory and computational requirements of self-attention grow quadratically with the input sequence length. In all, a re-ranker is vital to improve the quality of and reduce the amount of candidate context sentences to be processed by the downstream reader module.

Re-ranker uses a more computationally costly function to score the relevance of the smaller set of retrieved candidates for the question to further locate the most relevant evidence sentences from the retrieved documents returned by the retriever by pushing the most useful information to the top of the list. It narrows down the large candidates by truncating the list and returning the top-ranked ones to the reader for downstream answer extraction, and provides a ranked list of supporting facts, which is important for transparency and trustworthiness to users.

The re-ranker scores the relevance of the remaining small set of candidate sentences from the retrieved documents, and returns a ranked list of hits for the given question. The key is the relevance measurement between the question and candidate context, which is often estimated based on 1) sparse representations from traditional sparse vector space models, such as TF-IDF or BM25 [155], which rely on keywords matching with an inverted

index; or 2) the dense, latent semantic encoding of questions and context from neural models [64, 86]. Relevance is measured on the similarity of the vector representations with the inner product of the vectors as Figure 3.2, and then the relevance score is used to rank the relevant queries and documents to the top.

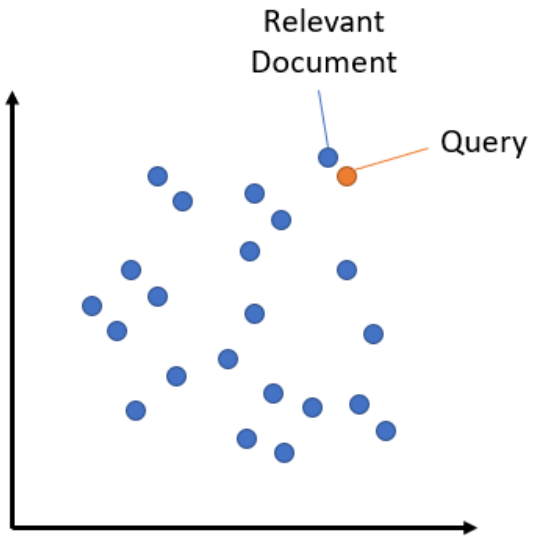


Figure 3.2: Semantic match by embedding question and all entries in the corpus into a vector space, the closest embeddings from the corpus are found (Figure taken from [145])

Most researchers consider the relevance computed using dense representations the same as semantic similarity, which often refers to the notion of semantic equivalence, i.e., "Do these two sentences mean the same thing?", operationally defined by the annotation guidelines of the well-known Semantic Textual Similarity (STS) tasks. For simple question answering, most of the questions can be answered by matching the question with a single sentence in a paragraph, which usually shares the same entity mention(s) with the question. Only measuring the semantic textual similarity between the question and candidates to determine the relevance is enough. However, we demonstrate it is not sufficient for multi-hop questions, whose evidences beyond the first hop might have little lexical overlap with the question and thus fail to be ranked at the top with models that only focused on STS.

According to formal semantic notions, the semantic relationship between two text fragments includes semantic equivalence, referential equality, and textual entailment. While referential equality can be mostly solved by coreference resolution and entity linking,

semantic similarity and textual entailment require deep semantic understanding between question and context [105]. Textual entailment is a framework that captures semantic inference. Textual Entailment (TE) of two text fragments can be defined as the task of deciding whether the meaning of one text fragment can be inferred from another text fragment. That is, a premise T entails a hypothesis H if, typically, a human reading T would infer that H is most likely true. For example, T: Jack sold the house to Peter. H: Peter owns the house. Here H can be inferred from T, so T entails H. While semantic equivalence capture "Do these two texts mean the same thing?", textual entailment is a framework that captures semantic inference, deciding whether the meaning of one text can be inferred from another.

It is common to utilize an inference model as a reader to infer the correct answer from a subset of retrieved context. However, most works of the evidence ranking only measure the semantic textual similarity between the question and candidate corpus to determine the relevance , and ignore the inference signals. We argue that semantic similarity is not the only relevance signal that needs to be considered when ranking evidences for multi-hop questions. As the example in Figure 3.3 shows, relevance is multi-dimensional. The first hop evidence "James Henry Miller (25 January 1915 – 22 October 1989), better known by James Henry Miller stage name Ewan MacColl, was an English folk singer, songwriter, communist, labour activist, actor, poet, playwright and record producer" has lexical overlap with the question and thus get high scores from token overlap match and semantic similarity comparison. However, the subsequent hops in the paragraph 'Peggy Seeger' failed to be retrieved by both lexical match and semantic similarity comparison, as they would not be considered as "mention or mean the same thing" as the question. Instead, they have an entailment relation with the question. "Margaret 'Pegg' Seeger (born June 17, 1935) is an American folksinger." entails a nationality relationship, and similarly 'James Henry Miller's wife' entails a marriage relationship.

In this work, we demonstrate that textual entailment relation is another important relevance dimension that should be considered when ranking evidences for multi-hop questions. Multi-hop questions require synthesis and inference information from multiple documents. Relevant information for answering such a question is from multiple documents or paragraphs that are either semantically equivalent to or entailed by the question. To identify evidences that are either semantically equivalent to or entailed by the question

Question: What nationality was James Henry Miller's wife?
Answer: American

Supporting Evidences

Ewan MacColl
(1) James Henry Miller (25 January 1915 – 22 October 1989), better known by James Henry Miller stage name Ewan MacColl, was an English folk singer, songwriter, communist, labour activist, actor, poet, playwright and record producer .

Peggy Seeger
(2) Margaret "Peggy" Seeger (born June 17 , 1935) is an American folksinger.
(3) She is also well known in Britain, where she has lived for more than 30 years, and was married to the singer and songwriter Ewan MacColl until his death in 1989.

Semantic Equality
James Henry Miller (Q) $\approx$ James Henry Miller (25 January 1915 – 22 October 1989), better known by James Henry Miller stage name Ewan MacColl (1)

Textual Entailment
American (2) $\vdash$ nationality (Q)
was married to (3)$\vdash$ wife (Q)

Figure 3.3: An example from the HotpotQA dataset [203] showing the two different dimensions of relevance between the question and its supporting evidences. Paragraph 2 is unlikely to be retrieved using TF-IDF due to little lexical overlap to the question.

simultaneously, we divide the task of evidence re-ranking for MHQA into two separate ranking tasks, i.e., semantic textual similarity and inference similarity-based ranking. Unlike most research works on evidence retrieval and re-ranking that treat each candidate independently [203, 128, 107, 63, 98, 97, 190, 86], we presented two ensemble models, EAR and EARnest, tackle each of the sub-tasks separately with off-the-shelf models, and jointly rank the candidates with the consideration of the diverse relevance signals. Experimental results on HotpotQA verify that our models not only significantly outperform all the single models it is based on, but are also more effective than two intuitive ensemble baseline models.

3.2. Related Work

Typically, the retriever model builds an index for all the candidate documents for search purposes using either lexical search based on sparse vector representations learned from a term-based method or semantic search based on dense representations in a low-dimensional and continuous space learned from a dense retrieval model Then it can efficiently query the index at run-time using a comparable representation vector of the question, by computing relevance score via a lightweight similarity comparison function. A re-ranker only runs over a potentially relevant candidate set returned by the retriever, which is of a much smaller size. It is designed to have higher performance but is often more computationally costly. Specifically, it outputs a single score between 0 and 1 indicating how relevant a sentence within candidate documents is for the given question, ranks, and returns the high-scored ones accordingly.

3.2.1. Traditional Retrieval Models

The basic IR architecture uses the vector space models (such as TF-IDF and BM25 [151]) to map queries and documents to sparse bag-of-words representations, and use the cosine similarity between the vectors to rank potential documents.

TF-IDF is short for Term Frequency (TF) - Inverse Document Frequency (IDF). As illustrated in Eq. (3.1), TF measures the number of times that a term t occurs in document d proportional to the word number of document d. IDF is a measure to evaluate if a term is common or rare across the document collection. Given a query q, score of a document d using the TF-IDF values is:

$$\begin{aligned} score(q,d) &= \sum_{t \in q} \frac{tf\_idf(t,d)}{|d|} \\ tf\_idf(t,d) &= tf_{t,d} \times log \frac{N}{df_t} \\ tf_{t,d} &= log(count(t,d)+1) \end{aligned} \tag{3.1}$$

Where df_t is the number of documents containing t, and N is the total number of documents.

A slightly more complex variant in the TF-IDF is the Okapi BM25 (Eq. (3.2)). BM25 adds two parameters: k, a knob that adjusts the balance between TF and IDF, and b, which controls the importance of document length normalization. Given a query containing n words $q_1, ...q_n$, BM25 score of document d is :

$$score(d,q) = \sum_{t \in q} log(\frac{N}{df_t}) \cdot \frac{tf_{t,d}}{tf_{t,d} + k \cdot (1 - b + b \cdot \frac{|d|}{|d_{avg}|})} \tag{3.2}$$

where $|d|$ is the length of d, and $|d_{avg}|$ is the average document length in the document collection. A large k results in raw term frequency (plus IDF). b scales document length, ranging from 1 to 0.

Such sparse retrieval models match keywords between the query and documents efficiently with a data structure called an inverted index and return sparse vector representation in high dimension, where the number of dimensions is equal to the vocabulary size, with most of the elements being zero. However, they are mostly limited to exact matches. It fails to recognize synonyms, acronyms, or spelling variations. Useful information might be missed by them due to the "lexical chasm" [16] between the question and the answer.

3.2.2. Learning-to-Rank Methods

A class of supervised machine learning techniques that attempt to solve ranking problems is referred to as learning-to-rank. In general, the goal of learning-to-rank is to learn a ranking function from labeled training data that maps feature vectors to scores. During inference, this scoring function is used to sort and rank items. Specifically, these methods collect various types of features (such as TF-IDF scores, BM25 scores, and Page-rank) as relevance signals, and represent documents as feature vectors, and learn linear or non-linear models to produce the relevance score by optimizing some ranking-based loss functions. According to their loss functions, these methods are categorized into three approaches: point-wise approach, pair-wise approach, and list-wise approach.

Point-wise methods [193, 41, 38, 121, 102, 42, 165, 91] approximate ranking to a classification or regression problem, where a function is learned to predict the relevance score of a document, and attempt to optimize the discrepancy between individual ground-truth label and the relevance score.

Pair-wise methods [26, 80, 132, 90] reduce ranking to pairwise classification, in which a binary classifier is learned to tell which document is better in a given pair of documents. In many cases, the binary classifier is implemented with a scoring function. The goal is to minimize a loss function that may reflect the average number of inversions in ranking.

List-wise methods [175, 129, 152, 28, 136, 197, 32] algorithms try to directly optimize the evaluation measures that average over all instances in the training data. This is difficult because most evaluation measures are not continuous functions and are non-differentiable since they depend on the rank positions, so continuous approximations or bounds on evaluation measures have to be used.

3.2.3. Neural Models

Neural ranking models learn ranking features automatically [46, 201]. They move away from sparse signals to continuous dense representations, which capture semantic meanings and help to address the vocabulary mismatch problem, overcoming the shortcomings of lexical search and can recognize synonyms and acronyms. Dense vector representations have been used for retrieval and ranking for decades since Latent Semantic Analysis [45]. Recently, facilitated by recent developments in DNNs, multiple works [86, 58, 73, 205] trained dense encoders using annotated pairs of queries and candidate documents to generate similar embeddings for semantically similar queries and documents, while minimizing similarity of embeddings for irrelevant queries and documents. They encode the question and candidates into vector space and retrieve the documents that are close in vector space according to the embeddings.

Before the advent of DL, neural ranking models can be divided into three major categories: representation-based, interaction-based, and hybrid [1]. Earlier representation-based neural models encode both the question and the candidate answers as fixed-dimension vectors with neural networks such as RNN or CNN. In contrast, interaction-based models capture the interaction between each term of question and candidate answer sentences. The interaction is often modeled with attention mechanisms or matching strategies. And hybrid models combine both.

With recent developments in pre-trained language models such as BERT, neural ranking models have seen significant improvements in effectiveness [105]. Pre-trained Neural

ranking models can be categorized into two types according to their model architecture, namely *Bi-Encoders* and *Cross-Encoder*, which jointly or separately encode a query q and a passage p, respectively. Figure 3.4 demonstrates the difference between Bi-Encoder and Cross-Encoder.

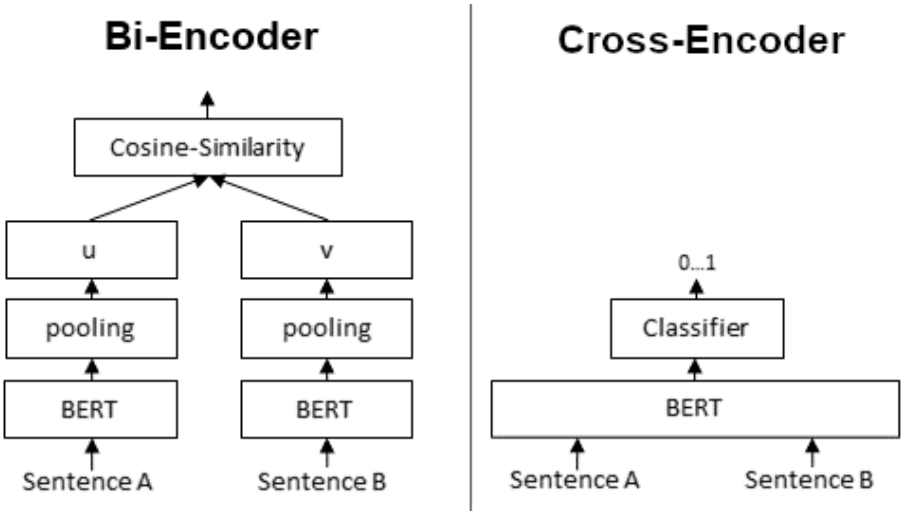


Figure 3.4: Bi-Encoders produce for a given sentence a sentence embedding by passing the individual sentences (such as A and B) to a BERT-like transformer model independently, which results in the sentence embeddings for each sentence. These sentence embeddings can then be compared using cosine similarity. In contrast, a Cross-Encoder produces an output value between 0 and 1 indicating the similarity of the input sentence pair by passing the sentence pair simultaneously to the Transformer network (Figure taken from [145]).

Bi-Encoders encode question and candidate sentences separately and then use a similarity comparison function (such as cosine similarity) to estimate the relevance. Cross-Encoders are trained by taking the concatenated question and candidate sentence as a single input sequence and generating an estimate of relevance score directly. Bi-Encoders are used when we need sentence embeddings for efficient comparison, by making the passage representations indexable and thus enabling efficient retrieval supported by approximated nearest neighbor (ANN) search, e.g., FAISS [81]. Using Cross-Encoders to compute similarity scores for about 50 Million sentence combinations takes about 65 hours. While with a Bi-Encoder, the embeddings for each sentence are computed in only 5 seconds [145]. Cross-Encoders would be the wrong choice for the application to large document collections with millions of entries since the computational overhead is too large. While usually

much faster, Bi-Encoders are less effective than cross-encoder models because the latter can exploit relevance signals derived from attention between the query and candidate sentence at each transformer encoder layer. Since Bi-Encoder loses attention-based interactions between queries and texts from the corpus, its effectiveness degradation is to be expected. It might return irrelevant candidates. Therefore, a re-ranker is important to score the relevancy of the remaining small set of sentence candidates from the retrieved documents for the given search query. Cross-Encoders are often used as the re-ranker to score the much smaller set of sentence pairs from retrieval due to their stronger performance.

3.2.4. Hybrid Methods

A hybrid approach for relevance measurement that combines the advantages of both dense retrieval with sparse retrieval methods together better estimates the relevance between the query and the candidates and thus generally performs better than either individual strategy.

[86, 113] combine results from dense passage retrieval (DPR) and sparse retrieval (BM25) by computing the linear combination of their respective scores to re-rank the union of the two initial retrieved sets.

Gao et al. [57] jointly trained a sparse–dense hybrid model, CLEAR. It uses a bi-encoder to capture semantic matching absent in the lexical model (BM25), instead of having the dense retrieval model "relearn" aspects of lexical matching. The dense retrieval score is computed as the inner product between encoder outputs. As CLEAR is a sparse–dense hybrid, the final relevance score is computed by a linear combination of the lexical retrieval score (produced by BM25) and the dense retrieval score. CLEAR is trained using a pairwise hinge loss to maximize the similarity between a given query and a relevant document while minimizing the similarity between the query and a non-relevant document subject to a minimum margin.

BISON ("BM25-weighted Self-Attention Framework") [164] uses a stack of modified "BISON encoder layers" that are a transformer encoder variant in which self-attention computations are weighted by term importance, calculated using a variant of TF–IDF.

All the dense–sparse hybrids appear to be more effective than either alone. The entailment-aware re-ranking models we introduce in this chapter are also hybrid methods, but we further consider entailment relationships besides semantic equivalence for relevance

estimation, and thus we aggregate multiple dense models with a traditional sparse model to take multiple relevance signals into consideration.

The entailment-aware re-ranking models we present are also hybrid methods that combine sparse and dense methods, but our method is unsupervised. Further, we combine a sparse model with multiple dense models to consider diverse relevance signals, i.e., textual entailment in addition to semantic equivalence.

3.3. Methodology

In this section, we introduce two ensemble models for entailment-aware multi-hop QA evidence re-ranking to model the diverse relevance signals as shown in Figure 3.5. At a high level, we model diverse relevance relationships with three base models to capture the semantic equivalence and entailment relations separately, which produce different and potentially conflicting rankings. In order to generate an aggregated ranking, we combine the relevance signals of the base models captured with ensemble methods to jointly estimate the relevance score for multi-hop questions.

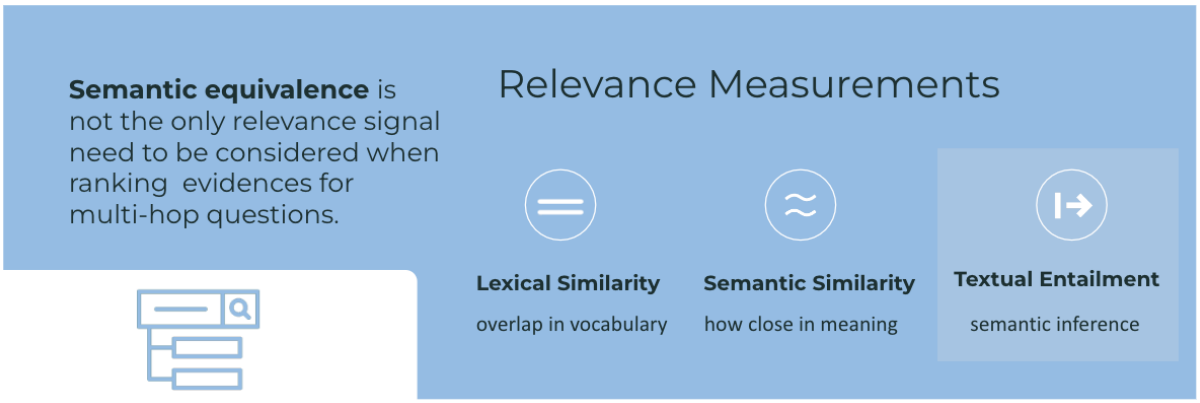


Figure 3.5: Diverse relevance signals. Lexical and semantic similarity together capture the measure of the semantic equivalence between question and candidate evidences. Textual entailment captures the semantic inference relations. While semantic equivalence capture "Do these two texts mean the same thing?" textual entailment is a framework that captures semantic inference, deciding whether the meaning of one text can be inferred from another.

3.3.1. Task

The evidence ranking for the multi-hop QA task broadly involves two different dimensions of relevance to measuring: semantic equivalence and textual entailment. Figure 3.3 shows a multi-hop question example from the HotpotQA dataset. Beside the semantic equivalence relation between the question and its first hop evidence, the textual entailment relation between the question and its secondary hop evidence, such as American and nationality, is another important relevance dimension that should be considered. To identify evidences that are either semantically equivalent to or entailed by the question simultaneously, we divide the task of evidence re-ranking for multi-hop QA into two separate ranking tasks, i.e., semantic textual similarity and inference similarity based ranking. *Semantic textual similarity* is a commonly used relevance measurement for in evidence ranking. *Inference Similarity* (IS) measures the textual entailment between the question and its candidate evidences. For example, 'where' questions would have high IS with evidence about locations or directions. Each of the two sub-tasks aims to rank candidates that score highly on one of the relevance dimensions respectively, and then combine them to output the final ranking with an ensemble model.

Given a multi-hop question Q and a corpus $C = \{P_1, P_2, \ldots, P_m\}$ containing a set of documents or paragraphs, a retrieval system aims to select a small set of relevant supporting facts for the reader module to infer the answer. The corpus size can easily range from millions to billions (e.g., Wikipedia or the web) for open-domain question answering to cover a diverse range of domains. As a result, the retrieval system has to make the trade-off between efficiency and accuracy. The solution is to integrate efficient retrieval methods with an effective re-ranker for the ODQA system to scale across the enormous amount of candidate documents without hurting accuracy. An efficient retriever selects a small subset D of documents that are likely relevant to the question. That is, a retriever $F : (q, C) \rightarrow D$ is a function that takes as input a question q and a corpus C and returns a much smaller filtered set $D \subset C$. The re-ranker first needs to perform sentence segmentation (sentence as the "unit of indexing") over each of the documents in the retrieved subset D to get M total candidate sentences S. Then the re-ranker estimates the relevance of a question and candidate evidence sentence pairs $r(q, s)$, ranks the candidate sentences in S, and outputs

the top-ranked N relevant sentences as supporting facts for answering q. Formally, given the input of a question q and candidate sentences S, re-ranker is a function of $R : q, S \rightarrow S'$, which takes the candidate sentences in S, and returns a subset S' with $r(q, s_h) > r(q, s_l)$ if $h < l$, where $r(q, S) = (s_1, ..., s_k)$. Relevance estimation of each candidate sentence to the question is clearly the key to the task. Two dimensions of relevance (i.e., semantic equivalence and entailment) need to be considered in order to provide a more accurate estimation of the relevance of each candidate sentence for the multi-hop questions.

We divide this multi-criteria task into two separate ranking subtasks: semantic equivalence as well as textual entailment.[1] Both tasks require comparing information between the question and candidates, but the objectives of the comparison are different. In this work, we capture the entailment relations in parallel with the semantic equivalence with separate models, which produce different and potentially conflicting rankings. The goal is to combine them to figure out an aggregated ranking that better estimates overall relevance and promotes gold evidence sentences to the top of the list.

3.3.2. Base Models

In order to build a re-ranking method that does not rely on a large training set with evidence annotations, we chose three off-the-shelf base models (Figure 3.6) to capture diverse relevance patterns. To better estimate semantic equivalence, we use both a sparse model (i.e., BM25) and a dense model to examine exact match and semantic match respectively. BM25 relies on lexical match, which looks for literal matches of the question words in the document collection. In addition, we utilized another dense model pre-trained on QNLI dataset for capturing entailment relations.

While making use of BM25 as a relevance signal relies on the lexical overlap, dense models address a variety of linguistic phenomena beyond exact term matching, including synonyms, paraphrases, term variation, and different expressions of similar intents. For example, a dense model would be able to better match ”password reset” with ”account unlock” and fetch the relevant context, while a term-based system would have difficulty connecting them without exact token matching. Dense models using semantic search overcome the shortcomings of lexical search and can recognize synonyms and acronyms, and

[1]For the focus of this work, we conduct coreference resolution within each paragraph in advance.

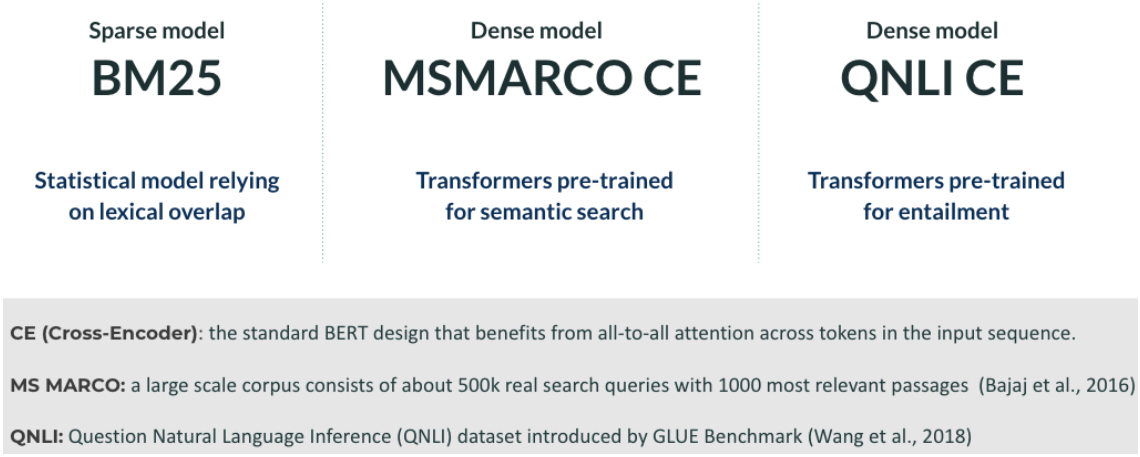


Figure 3.6: Base models. We use off-the-shelf statistical models and pre-trained language models as base models to capture the diverse relevance signals. For estimating semantic equivalence, we use a sparse model BM25 and a transformer-based dense model MS-MARCO Cross-Encoder to capture exact match and semantic match respectively. In addition, we utilized another dense model QNLI cross encoder for capturing entailment relation.

thus are complementary to sparse vector models. Liang et al. [103] introduced a lexical probing task that validates while dense models capture semantic information and have the capacity to capture lexical differences between questions and answers, they miss obvious lexical overlap signals and do not take direct advantage of obvious lexical evidence, due to their reliance on continuous representations. Thus, dense models such as universal sentence encoder (USE-QA) [200] do not always outperform traditional information retrieval methods such as BM25 for evidence retrieval for QA.

For the dense models, we choose two pre-trained cross-Encoders (CE) instead of Bi-Encoders. While by definition, CE supports more extensive query–document interactions than bi-encoders and thus can exploit richer relevance signals and achieve better performances [145]. A re-ranker based on a CE can substantially improve the final results for the user. The query and a possible document are passed simultaneously to the transformer network, which then outputs a single score between 0 and 1 indicating how relevant the document is for the given query. The advantage of CE is the higher performance, as they perform attention across the query and the document.

The two pre-trained CEs we choose are:

MSMARCO Passage Cross-Encoder is trained on the MS Marco Passage Ranking dataset [12] for information retrieval. MS MARCO (Microsoft Machine Reading Comprehension) is a large-scale corpus consisting of about 500k real search queries from the Bing search engine with 1000 most relevant passages. The model is trained to rank the most relevant passage that answers the query labeled by humans as high as possible. It is a strong base model commonly used for semantic search to find relevant text for the given search query.

QNLI Cross-Encoder is a pre-trained model obtained using the Question Natural Language Inference (QNLI) dataset introduced by GLUE Benchmark [187]. Given a passage from Wikipedia, annotators created questions that are answerable by that passage. The positive examples are (question, sentence) pairs that contain the correct answer, and the negative examples are pairs with sentences from the same paragraph that do not contain the answer. QNLI was automatically derived from SQuAD, the Stanford Question Answering Dataset v1.1 with the processing target of question-answer entailment.

BM25	MSMARCO CE	QNLI CE	% Ques (k=3)	% Ques (k=5)
✓	✗	✗	14	10
✗	✗	✓	25	22
✗	✓	✗	20	16
✗	✓		33	30
✗		✓	38	35
	✓	✗	64	62
	✗	✓	35	33
✗	✗	✗	44	29

Table 3.1: Base Models Comparison. Each line shows the percentage of questions that have at least one evidence ranked within the top-k by the model marked with a '✓' but beyond top-k by the model(s) marked with a '✗'. For example, there are 14% of the questions with at least one evidence sentence are ranked within top-3 by BM25[2], but ranked beyond top-3 by MSMARCO CE and QNLI CE; 35% of the questions with at least one evidence sentence ranked within top-3 by QNLI CE but ranked beyond top-3 by MSMARCO CE.

[2] `https://pypi.org/project/rank-bm25`

3.3.3. Ensemble Models

Ensemble modeling is a process where multiple diverse base models are created to predict an outcome, either by using several different modeling algorithms or using different training data sets, the ensemble model then aggregates the prediction of each base model and results in one final prediction. Since the three base models aforementioned independently capture diverse relevance signals and are complementing each other as shown in table 3.1, an ensemble model that seeks the wisdom of crowds to take advantage of all base models should potentially improve the final retrieval performance if an appropriate aggregation strategy is designed to combine them.

In this section, we explore several ensemble techniques to combine the base models. We first introduce two simple and intuitive ranking aggregation strategies as baselines, and then present two ensemble models and explain why they are better than the baselines. We will show the benefits of our ensemble models in the experimental results (Section 3.4).

Ensemble Baselines

Average ranking (AR) is a simple ensemble ranking model which combines the ranking outputs from the multiple base models that ranks all candidate sentences independently. It simply sums up all the ranks from base models for individual candidate evidence, and re-rank all the candidates according to the summation of the ranks. A rank of a candidate sentence obtained by each base model is with respect to the relevance signal the base model targets. Thus, each sentence has M ranks (where M is the number of base models). The final ranking is obtained by sorting the *sum* of all M rankings that each sentence received. Table 3.2 illustrates the AR method with a simple example.

Similarity Combination (SimCom) calculates hybrid retrieval scores through a linear combination of sparse and dense scores. For a given question, each sentence got a sparse score from BM25[3], and dense scores from semantic textual similarity (STS) and inference similarity (IS) from the two cross-encoders. To consider diverse relevance dimensions simultaneously, we add scores produced by different base models together as a weighted normalized sum, called Question Evidence Relevance (QRE):

[3] `https://pypi.org/project/rank-bm25/`

Sents	R_{BM25}	$R_{MSMARCO}$	R_{QNLI}	Sum(R)	AR
S_1	1	1	5	7	1
S_2	4	3	2	9	3
S_3	3	4	6	13	5
S_4	5	6	3	14	6
S_5	6	5	1	12	4
S_6	2	2	4	8	2

Table 3.2: A pseudo-example of an average ranking method. Each of the 6 candidate sentences is ranked by the three base models according to its degree of relevance to the query with respect to the relevance signal each model captures. When aggregating the ranking results of the various models, the average ranking method simply sums all the ranks for each sentence, and re-rank all the sentences according to the summation of the ranks of each sentence.

$$QRE_{q,s_j} = \eta(BM25(q, s_j)) + \alpha \cdot \eta(STS(q, s_j)) + \beta \cdot \eta(IS(q, s_j)) \tag{3.3}$$

which first normalizes the output similarity scores with η[4], and then combines the normalized scores by weights α and β. QER is then used to rank the candidate evidence sentences, so that candidate sentences with high relevance to the question are promoted to the top of the list.

Entailment-Aware Re-ranking

Both the order-based ensemble ranking and the score-based ensemble baselines calculate the final ranking based on low level fusion approaches to combine the outputs of the base models. In this work, we present an entailment-aware re-ranking (EAR) method to jointly consider pairs of candidate sentences that potentially contain complementary relevance signals. We form such sentence pairs using the Cartesian product of two sets of top ranked candidate sentences with respect to semantic equivalence and textual entailment correspondingly. Figure 3.7 illustrate an overview of our EAR method.

While BM25 and MSMARCO Cross-Encoder capture exact and semantic matches, respectively, they both aim for estimating STS. Thus, we take the union of top-ranked sentences by BM25 and MSMARCO Cross-Encoder as a unified set $\mathcal{A} = \{S_{a_1}, S_{a_2}, S_{a_3}, S_{a_4}\}$,

[4] η performs normalization to scale inputs to unit norms with Scikit-learn's normalizer:`https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html`

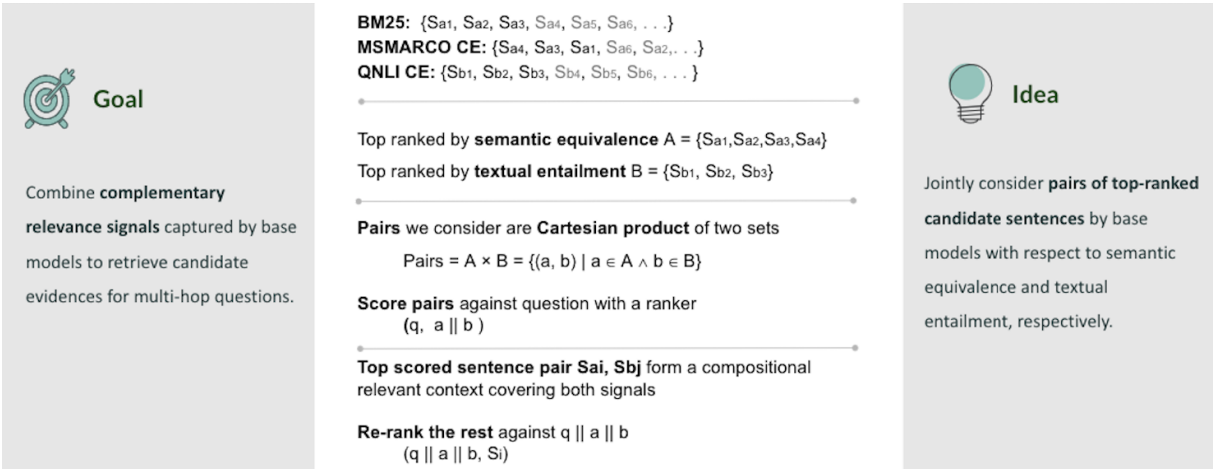


Figure 3.7: An overview of our ensemble method EAR, short for entailment-aware re-ranking. It jointly considers pairs of candidate evidences that potentially contain complementary relevance signals.

and top-ranked sentences by QNLI Cross-Encoder as another set $\mathcal{B} = \{S_{b_1}, S_{b_2}, S_{b_3}\}$. The pairs we consider are $\mathcal{P} = \mathcal{A} \times \mathcal{B} = \{(\mathbf{a}, \mathbf{b}) \,|\, \mathbf{a} \in \mathcal{A} \wedge \mathbf{b} \in \mathcal{B}\}$. We then concatenate the two sentences of each pair as a sequence to score against the question with a re-ranker[5], such that the top-scored sentence pair (S_{a_i}, S_{b_j}) is most likely to form a compositional relevant context covering both semantic equivalence and entailment relevance signals. When S_{a_i} and S_{b_j} are examined individually, there is a high chance that S_{a_i} receives a low IS score from the QNLI cross-encoder and is ranked down to the list, S_{b_j} can be scored and ranked low by BM25 and MSMARCO cross-encoder. Thus, either using individual base models or aggregating ranking or scores with the ensemble baseline models, S_{a_i} and S_{b_j} are unlikely to be both promoted to the top of the ranking list. Finding the best combination from the top-ranked subsets with respect to both semantic equivalence and textual entailment efficiently takes the compositional requirement into consideration. In case there are additional relevant sentences besides the highly relevant sentence pair for a question, we further concatenate the question q with the pair S_{a_i} and S_{b_j} as a new query to re-rank the rest of the candidate sentences.

[5]We use the MSMARCO Cross-Encoder as the re-ranker since it is the best performing base model.

EARnest

Evidences for a multi-hop question should be intuitively related, and often logically connected via a shared named entity that would allow a human reader to connect the information they contain. For example, "Viglen" in table 2.2. The presence of a shared named entity between two candidate sentences often indicates the likelihood that the sentence pairs relate to each other and, thus, they can be connected to form a coherent context for the question. To leverage such connection as an additional cue, we add a *named entity similarity term (NEST)* to the scoring function of the re-ranker in EAR, as shown in Eq. (3.4), to score the top-scored sentence pairs.

$$QER_{_{Earnest}} \ = \ (1 \ + \ NEST) \ * \ Sim(q, \ s_i \parallel s_j) \tag{3.4}$$

where $Sim()$ is the scoring function of the re-ranker, which scores the concatenation of sentence pair S_i and S_j against the question. $NEST$ is a binary switch, that is, if the two sentences share one or more named entities, the promotion mechanism is activated; otherwise, it is deactivated.

Named Entity Similarity Term: Besides using SpaCy [72] to recognize named entities with common entity types (such as names of people, places, and organizations), we also consider titles of documents and phrases between a pair of single or double quotes. When comparing whether two sentences share an entity, we apply basic normalization (i.e., lower case, removing articles and special punctuations) and fuzzy match to tolerate typos, variations, and inclusive match.

3.4. Evaluation and Results

3.4.1. Dataset

We conduct our evaluation using the HotpotQA dataset [203]. HotpotQA contains multi-hop questions created by human annotators using documents from Wikipedia as the information sources, instead of knowledge bases. Thus, the questions are not restricted by a fixed KB schema and cover more diverse topics. We evaluate our methods on the 5918

bridge-type questions out of the 7,405 examples from the development partition of the HotpotQA dataset in the distractor setting. Each question in HotpotQA is supported by two documents and provided with ground-truth evidence sentences, which enables us to evaluate the evidence retrieval performance of the various models.

3.4.2. Metrics

To evaluate and compare the performance of the base models and various ensemble methods, we adopt the standard retrieval performance metrics, including precision at different cut-off points (P@k), mean average precision (MAP), and recall at different cut-off points (R@k). For a given question, P@k is the percentage of relevant sentences in the top-k ranked results of the ranking list. The average precision is defined as the average of the P@k values of all ground-truth evidence sentences for a single question, and MAP is the mean of the average precision scores over all the questions. R@k is the portion of ground-truth evidence sentences ranked among the top-k results. With Eq. (3.5), the Average Precision (AP) for a question is the average value of the precision at every position in the ranked list. MAP simply averages the AP for every question in the dataset, and it is a standard measure for information retrieval. One drawback of P@k and R@k is that they fail to take into account the positions of the relevant sentences among the top k. And the average number of ground-truth evidence sentences is less than 3. For k that is larger than 3, even a perfect system will have a P@k score of less than 1 and decrease with higher k. Thus, MAP is a relatively more important metric to exam.

$$\begin{aligned} AP &= \sum_{k=1}^{K} \frac{p(k))}{k} \\ MAP &= \frac{1}{Q} \sum_{i=1}^{Q} AP_i \text{ , where Q is the number of questions} \end{aligned} \tag{3.5}$$

3.4.3. Results

Table 3.3 reports the evidence ranking performance of all models discussed. All three base models that target either semantic equivalence or inference do not yield optimal performance. As expected, the MSMARCO CE achieves the highest performance among the base models, as it is a strong baseline that is commonly used for retrieval tasks. However, it only considers the semantic matching between question and individual candidate sentences, ignoring the other important relevance matching characteristics such as exact matching signals, textual entailment, and relatedness between candidate evidence sentences.

For the baseline ensemble models, AR performs worse than the MSMARCO CE, while being slightly better than BM25 and the QNLI CE. Its performance is essentially a compromise among the performances of the three base models because it directly averages the individual ranking results. In contrast, SimCom[6] does take advantage of complementary relevance signals from the base models, so as to perform better than any of the individual base models. This suggests that with an appropriate aggregation, the base models cooperate advantageously to produce a better ranking. However, it fails to deliver the best overall performance because it simply combines the final output scores from the base models without exploiting the interactions between the relevance signals behind.

Lastly, our approaches (i.e., EAR and EARnest) not only outperform the base models but also exceed the order-based and score-based ensemble models on all metrics. They both jointly consider diverse relevance signals simultaneously, and therefore achieve greater improvements in the performances. EARnest further considers the relatedness between evidence sentences, becoming our best model. It achieves the highest MAP, and is higher than the MSMARCO CE by 10%.

[6]The result of the SimCom model uses $\alpha = 3$ and $\beta = 1$, which achieves the highest performance according to the grid search results on 10% of the full dataset.

Models	P@3	P@5	MAP	R@3	R@5	R@10
Base models						
BM25	0.43	0.31	0.59	0.54	0.65	0.78
MSmarco	0.47	0.33	0.64	0.59	0.69	0.81
QNLI	0.33	0.25	0.46	0.43	0.52	0.65
Ensemble Baselines						
AR	0.43	0.31	0.61	0.55	0.66	0.83
SimCom	0.5	0.36	0.68	0.63	0.74	0.86
Our Approach						
EAR	0.53	0.36	0.71	0.66	0.76	0.86
EARnest	**0.55**	**0.38**	**0.74**	**0.7**	**0.78**	**0.87**

Table 3.3: Evidence ranking results of base models, baseline ensembles, and our methods on HotpotQA. As can be seen, the performance of our ensemble methods (EAR and EARnest) are effective for improving the performance in terms of all the metrics. Our best model EARnest achieves the highest MAP performance, outperforming all the base models and ensemble baselines.

3.4.4. Discussion

QER of Similarity Combination

$$QER_{q,s_j} = \begin{cases} \dfrac{\eta(BM_{q,s_j}) + \alpha \cdot \eta(STS_{q,s_j}) + \beta \cdot \eta(IS_{q,s_j})}{3} & \text{if } BM_{q,s_j} > 0 \\ \dfrac{\alpha \cdot \eta(STS_{q,s_j}) + \beta \cdot \eta(IS_{q,s_j})}{2} & \text{Otherwise} \end{cases} \tag{3.6}$$

where semantic textual similarity (STS) and inference similarity (IS) are scores from MS-MARCO CE and QNLI CE. It first normalizes the scores with η[7], and then combines the normalized scores using the weights α and β.

[7] η performs normalization to scale inputs to unit norms with Scikit-learn's normalizer:`https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html`

Impact of K

EAR and EARnest both jointly consider pairs of candidate sentences top-ranked by the base models. The cut-off parameter K is used to partition sentences considered as top-ranked by individual base models or not. The larger K is, the more exhaustive combination of candidate sentence pairs would be considered. However, the number of pairs is quadratic in the number of K, so it becomes much more computationally costly when K is too large. Thus, we test on 600 randomly sampled questions (about 10% of the full dataset) to compare the impact when changing the value of K. The resulting performance is the same when changing K from 3 to 5, while the number of pairs compared increases from 12 to 33.5 on average. This is expected because we only consider the top pair to score against the question, and sentences in the pairs are often more likely to be ranked closer to the top of lists by base models respectfully since they contain stronger relevance signals.

Necessity of Inference model

To further demonstrate the benefits brought by the inference model, we conduct an ablation experiment by replacing the QNLI CE in EARnest with MSMARCO CE while keeping everything else the same. We also compare the difference in performance with the randomly sampled 600 questions. The result is shown in Table 3.4.

Models	P@3	P@5	MAP	R@3	R@5	R@10
BM25	0.44	0.32	0.6	0.55	0.66	0.79
MSmarco	0.47	0.34	0.65	0.59	0.71	0.82
QNLI	0.34	0.24	0.45	0.43	0.52	0.64
EARnest	**0.56**	**0.38**	**0.75**	**0.71**	**0.8**	**0.88**
EARnest - QNLI	0.52	0.37	0.7	0.66	0.77	0.86

Table 3.4: With the EARnest ensemble model framework, we replace QNLI CE with Ms Marco CE, and the performance significantly decreased. Compared to the full EARnest model, MAP drops 5% without exploiting the QNLI CE model to capture the textual entailment relevance signal.

Without the QNLI CE capturing the entailment relation to promote evidences that can be inferred by the questions to the top, BM25 and MSMARCO CE might miss them according to lexical and semantic matches. Therefore, the result is significantly lower than the full EARnest model, which confirms that textual entailment is a very important relevance signal to the multi-hop QA evidence re-ranking task and should be considered along with the semantic equivalence.

3.5. Summary

Efforts are dedicated to enabling machines to automatically retrieve information and perform inference over texts, but an effective re-ranker on top of the retriever is vital for the QA system. In this work, we showed that successful relevance matching for evidence re-ranking in multi-hop QA requires considering diverse signals including exact matching, semantic textual similarity, textual entailment between question and candidate sentences, and relatedness between candidate evidence sentences. We applied off-the-shelf statistical models and transformers to capture different dimensions of relevance and effectively combined them to jointly retrieve candidate evidences that cover diverse and most relevant information for the question when concatenated. Experimental results on HotpotQA reveal that our models are effective for improving the performance of multi-hop questions, compared to all the single models they are based on, also the order-based and score-based ensemble baseline models.

CHAPTER 4

Learning Strategies for Question Answering with Fewer Annotations

4.1. Chapter Overview

Contemporary state-of-the-art QA models are commonly based on deep machine comprehension reader models [92, 206, 33, 67, 162], that is, using a neural network to extract the answer to a question from the given context. Although employing deep learning models has led to impressive performance improvements, they often suffer from "data hunger" [20] and low robustness issues. Training and even fine-tuning such models require a large amount of high-quality annotated data. However, annotating QA datasets is very costly and requires intensive manual labor. In practice when labeling budgets are limited, we usually can not afford to annotate a large scaled dataset. Obtaining expert manual annotations is infeasible as it is tedious and time-consuming, and crowdsourcing from paid labor is more practical to support collecting annotations for big datasets. But even with many efforts to clean up the crowd-sourced annotations, noisy examples in data are common [2, 29]. This pushes for more robust learning algorithms that are less sensitive to noise.

Active learning (AL) is an ML method that aims at minimizing labeling costs of training data acquisition without sacrificing accuracy [56]. With a limited annotation budget in practice, AL selects the most informative instances and queries their labels through the interaction with oracles (annotated by experts or by applying crowd-sourcing techniques) to learn. In contrast to regular supervised learning (passive learning), where the labeled

data are taken at random, the AL learner iteratively selects the most informative data that help evolve the model for annotation and updating the model with respect to the new annotations, as illustrated in Figure 4.1. The main purpose of this approach is that if the training algorithm can choose more informative data during the learning process, the model can reach almost the same accuracy as a supervised learning method with a much less amount of data.

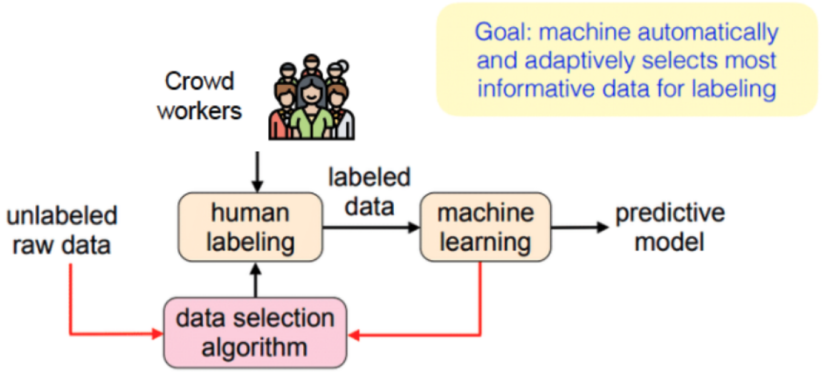


Figure 4.1: Active Learning Process

Although employing DNNs has led to impressive results on QA tasks, there are still several challenges that need to be addressed, such as needing to be trained with a massive amount of labeled examples and low robustness. For QA tasks, it has been demonstrated that an intentional perturbation of a paragraph through including adversarial sentences confuses even the best available QA models, causing a severe reduction in their accuracy [77]. This vulnerability against adversarial examples also makes these models unsuitable for real-world scenarios [21]. In order to build robust, reliable, and consistent DNNs, it is imperative to account for noise tolerance. Failure to do so may lead to unexpected behavior in the DNN, such as memorization or poor generalization.

Common acquisition functions for active learning are based on uncertainty and diversity [163, 50, 160]. The uncertainty-based function selects difficult examples according to probability scores from the model's predictions, and the diversity-based function selects heterogeneous data points in the feature space. The two approaches are orthogonal to each other, since uncertainty sampling is usually based on the model's output, while diversity exploits information from the input feature space [43].

In this work, we introduce a novel acquisition strategy, Perturbation-based Active Learning (PAL), considering the sensitiveness to perturbation as a signal of the informativeness of active learning. It utilizes both the input feature and the model's output predictions to select the next most informative batch of question examples to learn. With the integration of the AL approach into the QA reader model, I experiment with various AL query strategies together with our PAL strategy. For the investigation of the effectiveness of PAL in combination with BERT [46] for QA tasks, we conduct an empirical study to compare it with several commonly used AL query strategies with experiments by fine-tuning the BERT-based model and evaluate on the SQuAD dataset [142].

4.2. Related Work

4.2.1. Reading Comprehension Modeling

The reading comprehension (RC) task is a supervised learning problem that learns a predictor which takes a question and a passage as inputs and outputs the answer. Broadly, RC systems are grouped into three categories, rule-based, ML-based, and DL-based systems.

Rule-based Models

A rule can be a logical combination of any of the language processing tasks such as part-of-speech tagging, semantic class tagging, and entity recognition. Each rule awards some points to all sentences of the input [84]. After all of the rules have been applied, the one that obtains the highest score is returned as the answer. Riloff and Thelen [149] developed a rule-based system, Quarc, that uses heuristic rules that look for lexical and semantic clues in order to identify evidence (a sentence contains the answer to a question). Each type of WH question (WHO, WHAT, WHEN, WHERE, WHY) looks for different types of answers, so Quarc uses a separate set of rules for each question type. Terdalkar and Bhattacharya [180] designed a framework based on rules and heuristics for low-resource languages, saṃskṛta. It is developed using knowledge of grammar of saṃskṛta language and structure of the text. These models are based on hand-crafted rules that require substantial human effort and are incapable of generalization.

Machine Reading with Traditional Machine Learning

Traditional ML models transform RC into a supervised learning task, relying on a set of predefined features. After designing a set of features to capture the information that helps to distinguish answer sentences from non-answer sentences, the learning algorithm generates a classifier from the training examples for answer sentence classification.

Ng et al. [122] developed a ML-based RC system named AQUAREAS (Automated QUestion Answering upon REAding Stories). It introduced some of the features to be extracted from a context sentence like "the number of matching words/verb-types between the question and the sentence", "the number of matching words/verb-types between the question and the previous/next sentence", "co-reference information", and binary features like "sentence-contain-person", "sentence-contain-time", "sentence-contain-location", "sentence-is-title" and so on.

Wang et al. [191] designed an RC system with several modules, including the Name Identification Module, Partial Parser Module, Pronoun Resolution Module, Sentence-to-Question Comparison Module. Each of the modules depends on a set of chosen syntactic or semantic features. For example, Sentence-to-Question Comparison Module used sentence features such as containing a location, containing a time/date, and containing a human, etc. The Comparison Module determines how strongly the phrases of a sentence are related to those of a question, and this information is passed to modules that attempt to learn which features of the comparison are the most important for identifying whether a sentence is a strong answer candidate.

Poon et al. [138] combined bootstrapping, Markov logic, and self-supervised learning for machine reading: bootstrap from the head regime of the power-law distribution of textual knowledge, and conquer the long tail in a self-supervised learning process that raises certainty on sparse extractions by propagating information via joint inference from frequent extractions.

While these methods do not need to design hand-crafted rules, feature engineering is a critical necessity, demanding scientists to provide task-driven and insightful feature vector representations for underlying optimization problems. Besides, most traditional machine learning methods are also 'data hungry', and could potentially benefit from AL.

Neural Reading Comprehension

Recently, the research progress of RC has diverted from pure ML-based models to DL-based models [33, 183, 144, 99]. DL-based RC, also called neural reading comprehension, has gained state-of-the-art results. Most of the recent research falls into this category. Neural RC models automatically generate representations to better extract contextual information and dramatically outperform rule-based and traditional ML methods. Fine-tuning of pre-trained transformer models like BERT [46] or GPT [24] is the current state of the art.

Before transformers, the two main DL architectures for RC are the Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). RNN is a type of neural network used to process sequence data. When iterating through the sequence elements, outputs in each time step depend on the previous computations, and a state containing information relative to what has been seen so far is maintained. Among variants of vanilla RNNs, the most widely used are Long Short-Term Memory (LSTM) [71] and Gated Recurrent Unit (GRU) [35]. For example, [162, 36, 101] applied RNNs to model the interaction between contexts that are conditional on the question. The major shortcoming of RNNs is that their training process is time-consuming as they cannot be processed in parallel. CNN is a type of deep learning model that utilizes layers with convolution filters that are applied to local spots of their inputs [96]. Originally introduced for computer vision, CNN models have subsequently been shown to be effective in various NLP tasks [89]. [75, 162] used CNN to learn character embeddings in the RC systems. QANet [206] used a convolution layer to capture the local structure of the text and model local interactions. CNNs can extract local information more compactly and efficiently, and be trained in parallel, which is faster than RNNs. One main drawback of CNNs is that they only extract local information but are not capable of dealing with long sequences.

In recent years, attention-based transformer models have taken over. Details of performing RC with recent transformer-based models [24, 46] is explained in Section 4.2.2

4.2.2. Learning with Limited Annotations

Effective training of deep QA models with very little labeled data can be achieved by either semi-supervised learning (SSL) or active learning (AL).

Active Learning

AL is a subfield of ML in which a learning algorithm aims to achieve good accuracy with fewer training samples by interactively querying the oracles to label new data points [208]. AL aims to concentrate the expensive labeling process on the most informative instances from a (typically large) pool of unlabeled data in order to reach high accuracy with low-cost data labeling. AL has been shown to be effective in reducing the amount of labeling effort involved in training machine learning models. A good summary of active learning works prior to the advances in DL can be found in [163].

With the impressive breakthroughs DL made in various challenging tasks, the combination of DL and AL, referred to as deep active learning (DeepAL), attracted widespread research interest in recent years. Both DL and AL are subfields of machine learning. DL is also called representation learning. It realizes the automatic extraction of data features. DL has strong learning capabilities due to its complex structure, but this also means that DL requires a large number of labeled samples to complete the corresponding training. DL is limited by the high cost of sample labeling in some professional fields that require rich knowledge. AL focuses on the study of datasets, and it is also known as query learning. An effective AL algorithm can theoretically achieve exponential acceleration in labeling efficiency. Therefore, the combination of DL and AL is expected to consider the complementary advantages of the two methods to achieve superior results [147].

DeepAL has been widely used in various NLP tasks, such as NER [168, 110], coreference resolution [100], text classification [209, 169, 139, 159] entity resolution [87], and machine translation [109]. Here are two examples of works exploring DeepAL on the QA-related task:

Asghar et al. [9] uses the DeepAL to improve open-domain dialogue systems, by combining online AL strategy with the DL model, Seq2Seq. An online active learning phase interacts with human users for incremental model improvement, learning incrementally from user feedback in each round of dialogue. The online DeepAL is used as a form of reinforcement, which eliminates the need for hand-crafted reward criteria. The user provides feedback by selecting one of the K responses as the 'best' one or suggesting a (K+1)'th response. The selection criterion is subjective and entirely up to the user.

Lin and Parikh [106] presented an empirical study of DeepAL for the Visual Question Answering (VQA) task. They explored cramming (entropy), curiosity-driven (expected model change), and goal-driven (expected error reduction) active learning approaches to select informative question-image pairs from a pool and query an oracle for answers to maximally improve its performance under a limited query budget. They also presented a new goal-driven active learning scoring function to pick question-image pairs for deep VQA models under the Bayesian Neural Network framework. The paper found that deep VQA models need large amounts of training data before they can start asking informative questions. They found that deep VQA models need large amounts of training data before they can start asking informative questions. But once they do, all three approaches outperform the random selection baseline and achieve significant query savings.

Semi-supervised learning

SSL is a type of machine learning that uses both labeled and unlabeled data to create a model. Whereas the collection of data is often cheap, labeling data can usually only be achieved at enormous costs because experts have to annotate the data manually. SSL combines supervised learning and the usage of unlabeled data [146]. It is useful when there is not enough labeled data to create a model, but there is enough unlabeled large unlabeled data to provide some information about the model, which is a common case in many real-world machine learning problems. SSL methods offer a wide range of methods for leveraging unlabeled data when learning prediction models. Most of the SSL methods are based on combinations of a supervised loss and an unsupervised loss. Classical SSL algorithms include EM-based algorithms, self-training, co-training, semi-supervised SVM (S3VM), graph-based methods, and boosting-based SSL methods [7]. A batch of novel models has been recently introduced for SSL based on representation learning techniques, such as generative models, ladder networks, and graph embeddings. Here we list several recent works on Semi-supervised QA:

Yang et al. [202] presented Generative Domain-Adaptive Nets (GDAN) to utilize unlabeled text to boost the performance of the QA model. It uses a generator to generate fake questions using a set of unlabeled documents. A discriminator that tries to distinguish real questions from fake ones. The learning procedure of both the generator and discriminator

networks continues until the discriminator is unable to recognize the fake questions. And a domain adaptation algorithm based on reinforcement learning alleviates the discrepancy between the model-generated data distribution and the human-generated data distribution. Then they combine model-generated questions with human-generated questions for training QA models.

Dhingra et al. [48] introduced a system that uses automatically constructed cloze questions to improve the performance of QA models, especially when there are few labeled examples. The system works by exploiting the document structure to create cloze-style questions from base documents, pre-training a neural network on these cloze-style questions, and further fine-tuning the model on the labeled examples. The system is designed to be helpful in cases where large domain-specific annotated corpora are limited or expensive to construct.

Salant and Berant [156] show the benefit of training a QA model in a semi-supervised fashion with a large language model. They leveraged a language model, pre-trained on large amounts of data, as a sequence encoder to forcibly facilitate context utilization. By providing rich contextualized word representations from a large pre-trained language model as well as allowing the model to choose between context-dependent and context-independent word representations, they show that contextual word representations captured by the language model are beneficial for reading comprehension.

Wang et al. [189] develop a semi-supervised learning (SSL) method to automatically generate more training examples for training a multi-perspective network. It was done by matching the distribution of candidates between labeled and unlabeled data. The intuition is to make automatically constructed data as similar as possible to existing labeled data. Their multi-perspective network models multiple perspectives to arrive at the correct answer. It consists of several parallel modules, where each module aggregates context information from a unique perspective. They model long-distance matching with attentive readers, global semantics with iterative dilated convolutions, and lexical collocation with both N-grams and neural language models (LMs). The outputs of aggregation modules are further integrated and fed into a one-time-step pointer network to get the final answer.

4.3. Answer Extraction with Reader Model

Most modern ODQA systems have a reader model performing reading comprehension (RC) to extract an answer from the earlier retrieved potentially relevant documents. RC is a well-studied NLP task that answers questions about a given document called context, by identifying a span (a continuous string of text) in the context that contains the answer. For example, given a question like "In what year was the College of Engineering at Notre Dame formed?" and a context passage that contains the clause "The College of Engineering was established in 1920", a reader will output 1920. A reader model performing RC needs to have a semantic understanding of the question and context and be able to locate the position of an answer span. To compute the probability of each possible answer span, it is common to assume that the probability of a span being the answer can be estimated by the probability of tokens being the start and end of the answer. The highest scored span is considered an answer.

Recently, RC has seen significant progress since the advances in the attention mechanism [195, 6, 198]. The neural attention mechanism allows the system to focus on the most relevant part of the context paragraph in order to answer a question. The attention mechanisms help improve the ability of RC systems to model complex interactions between a context paragraph and a query. A real game-changer is a new neural network architecture, called Transformer, introduced in the paper *"Attention is All You Need"* [184]. It is a type of neural network architecture that essentially utilizes the concept of self-attention. The self-attention layer allows each word position in an input sequence to attend to all positions in the sequence, which captures long-range dependencies between words, such as syntactic, semantic, and coreference relations. The transformer also performs multi-head attention, which allows the model to jointly capture different attention from different subspaces, e.g., jointly attend to information that might indicate both coreference and syntactic relations [172].

However, training a transformer-based model from scratch is computationally expensive. A pre-trained transformer model that has been already trained on massive amounts of text data (for example, BERT was trained on both BooksCorpus and Wikipedia) allows it to understand the language well and capture a surprising amount of common knowledge,

which is crucial for the QA task. And with small changes in their architecture and pre-training objective, they can be fine-tuned to perform various types of downstream NLP tasks. Pre-trained transformer models can be treated as representations encoders by simply dropping the original output decision layer, with the learned embedding representations of the input text encoded in the final hidden state vectors. Fine-tuned these pre-trained transformer models for downstream tasks can be easily done by replacing the original output layer with a new task-specific layer and continuing to train the model using a dataset of the task. For example, for the text classification task, an additional linear classifier layer projects the output embeddings of the *[CLS]* token into a vector that represents the probabilities over the output classes. Similarly, for the QA task, a question-answering header is placed on top of the pre-trained transformer model to convert the output embeddings of all the tokens to probability distributions for both the start and end tokens of the answer span.

BERT, short for "Bidirectional Encoder Representations from Transformers" [46], introduced in 2018, is an extremely widely used pre-trained language model that uses the transformer-based architecture. It is trained on masked language modeling (MLM) and next sentence prediction (NSP) tasks and these pre-training objectives are powerful in capturing the semantics of the natural language. MLM involves masking a percentage of tokens in a sample and training the model to predict the correct token in place of the mask. NSP involves taking two sentences and training the model to predict whether the second sentence is the subsequent sentence in the original document. Standard cross-entropy loss with AdamW optimizer is used to train the weights for both objectives. Figure 4.2 illustrates the pre-training and fine-tuning procedures for BERT.

To build a reader model by fine-tuning BERT, we add a question-answering head on top of the BERT model to perform the RC task. The BERT encoder generates a token embedding for every token in the input sequence, in which the question and context are concatenated with the *[SEP]* token. The output embeddings of all the tokens are fed into a question-answering head during fine-tuning to predict the start and end position of the answer span, as illustrated in Figure 4.3. The question-answering head is implemented as a feed-forward layer that has two sets of weights, one for the answer span start token and one for the end token. A dot product is calculated between the weights and the embeddings and returns a two-dimensional vector. A softmax activation function is applied to the vector

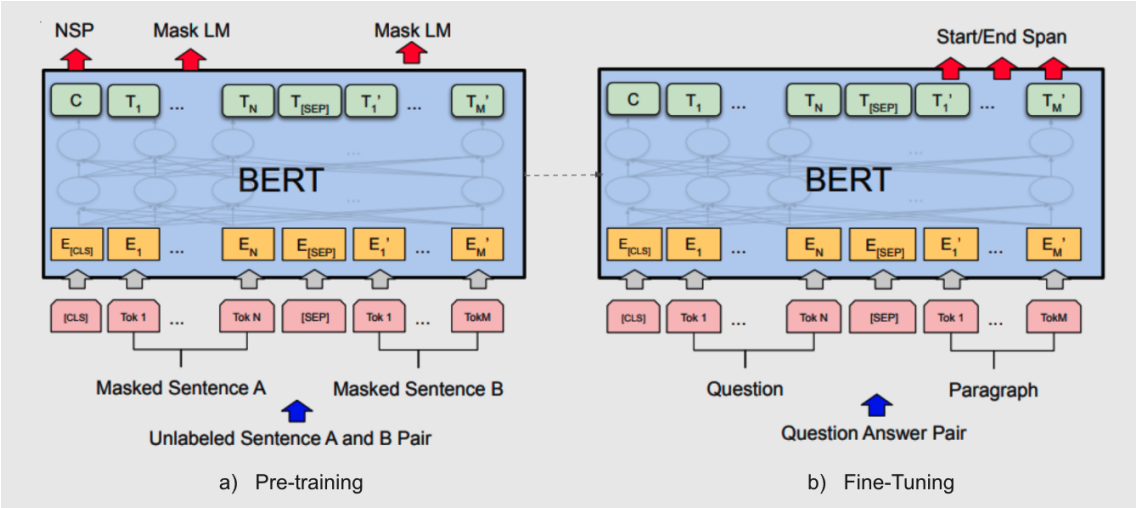


Figure 4.2: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token for separating questions/answers (Figure taken from [46]).

to produce two probabilities for each context token, which represent the likelihood of the token being the start and end of the answer span respectively. The model then identifies the span within the context with the highest sum of start and end probability values as the predicted answer to the question. The training objective for fine-tuning the reader is to match the predicted span with the correct answer span for each question instance, using a training loss of the negative sum of the log-likelihoods for the start and end tokens.

4.4. Methodology: Deep Active Learning

DL has led to impressive results on the QA task, but they are not robust enough and are extremely vulnerable to adversarial examples. To improve system performance, a large and high-quality set of training data is often required, but this is not always available. To improve the practical applicability of QA systems, the goal is to attain the best gain from this limitation. AL is a machine learning methodology where the learner is actively involved in the learning process by selecting which instances to learn from. This is in contrast to traditional learning algorithms where the data is passively presented to the learner. AL assumes

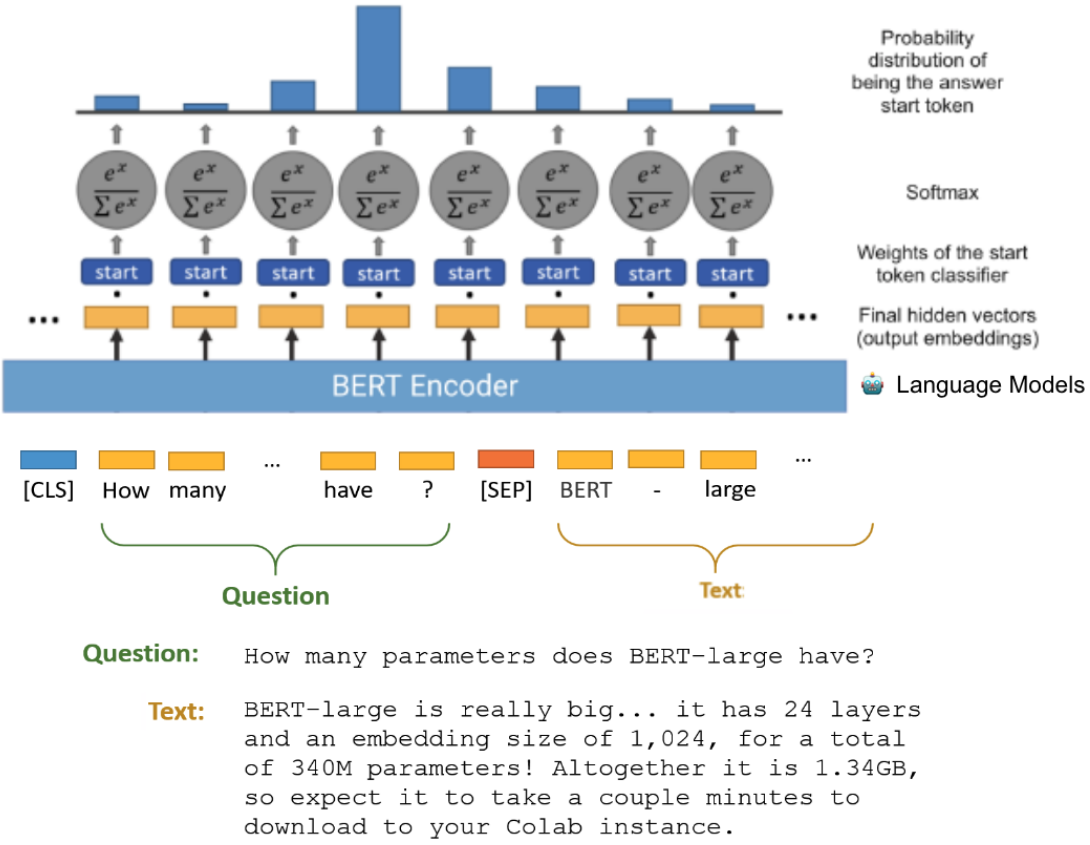


Figure 4.3: Prediction of start token of answer span, similar for the answer end token.

that different samples in the same dataset have different values for the update of the current model, and tries to select the samples with the highest value to construct the training set [147]. To do so, the AL process employs AL strategies to select informative unlabeled data samples to transfer more information to the model, and maximize the value of labeling a small set of examples. The combination of DL and AL is referred to as deep active learning (DeepAL). Figure 4.4 illustrates a typical DeepAL architecture.

DeepAL has been widely used in different subtasks of NLP, such as NER, coreference resolution, entity resolution, dependency parsing, sentiment classification, and machine translation. But the potential of using AL in conjunction with pre-trained models such as BERT for the QA task has not yet been fully explored. In this work, we study various AL strategies in conjunction with the most widely used DL model BERT for the QA task.

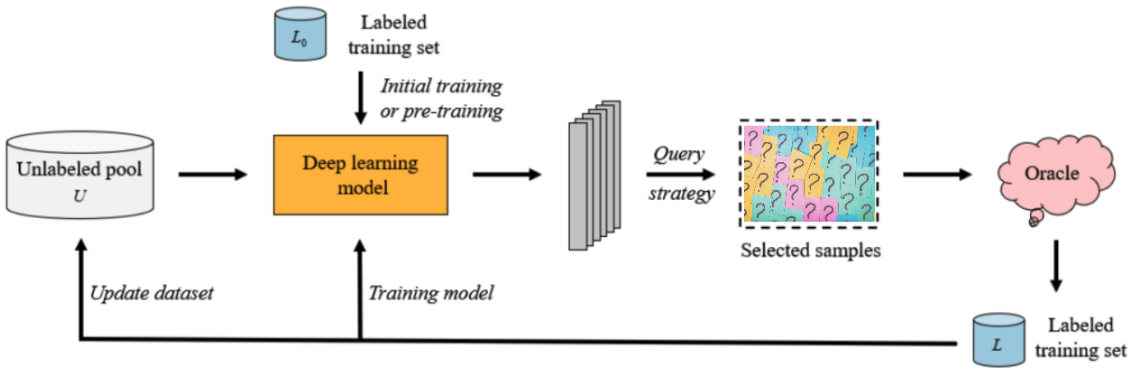


Figure 4.4: A typical example of DeepAL model architecture (Figure taken from [147])

DeepAL can help to build QA models with fewer annotation costs. In each iteration, the active learner selects a batch of the most informative unlabeled question-answer pairs and queries their labels from oracle to update the model effectively. Acquisition functions (or query strategies) for the AL learner, are used to determine how informative each question-answer pair is. As modern transformer models set the state-of-the-art on numerous tasks, including question answering, we use the transformer model in combination with several AL query strategies for efficient fine-tuning on the QA task.

Besides several common active learning acquisition strategies explained in Section 4.4.2, we explore a novel perturbation-based acquisition strategy (Section 4.4.3) to update the QA models and improve the robustness of the model gradually. This strategy takes as the most informative unlabeled samples whose predictive probability distributions diverge the most from their predictions before adding the distracting sentence measured by KL divergence.

4.4.1. Active Learning Iteration

In an active learning iteration: (1) a query strategy selects new instances from the unlabeled pool for labeling according to an estimation of their informativeness, (2) annotate or query the oracle to collect the ground-truth labels for these newly added instances. (3) the model is updated using the newly labeled instance or retrained from scratch using all the labeled instances so far.

Algorithm 1: Active Learning Approach

Input: Unlabeled data pool $\mathcal{U}$, AL acquisition function $\phi(\cdot,\cdot,\cdot)$, AL selected unlabeled samples $\mathcal{D}_{(t)}al$

while $|\mathcal{D}_{(t)}u| > 0$ **do**

  $M_t \leftarrow$ Continue fine-tuning M_{t-1} with $\mathcal{D}_{(t)}al$;

  $\mathcal{D}_{(t)}al \longleftarrow \arg\max_{*x \in \mathcal{D}_{(t)}u} \phi(M_t, x, 10\%)$;

  $\mathcal{D}_{(t+1)}l \longleftarrow \mathcal{D}_{(t)}l \cup label(\mathcal{D}_{(t)}al)$;

  $\mathcal{D}_{(t+1)}u \longleftarrow \mathcal{D}_{(t)}u \setminus \mathcal{D}_{(t)}al$

end

As shown in Algorithm 1, initially, one percent of the dataset is randomly selected to be annotated as $D_{(0)}l$ for fine-tuning the already pre-trained BERT-BASE model, and all the rest is a large unlabeled data pool $D_{(0)}u$. Then an active learner iteratively selects examples for annotation between rounds of training. Assume in the current AL iteration t, labeled data with answer annotation for training is $D_{(t)}l$ and a pool of unlabeled questions is $D_{(t)}u$. After continuing the fine-tuning of the pre-trained model on $D_{(t)}l$, the active learner applies the acquisition function to acquire a subset $D_{(t)}al$, consisting of 10% of unlabeled questions from the rest of the unlabeled dataset $D_{(t)}u$. The newly acquired questions are then labeled, and so they are removed from D_u and are added to the labeled dataset D_l along with their gold answer: $D_{(t+1)}l = D_{(t)}l \cup D_{(t)}al$, and $D_{(t+1)}u = D_{(t)}u - D_{(t)}al$. This procedure continues until all unlabeled questions are exhausted.

4.4.2. Common Acquisition Strategies

Instead of randomly choosing examples from an unlabeled corpus, AL acquisition strategies formulate an acquisition function that measures the usefulness of unlabeled data based on specified criteria. The goal is to find data samples that will transfer more information to the model so that the model can reach almost the same accuracy as a supervised method with a much less amount of data. According to the results of the acquisition function, selected significant unlabeled instances are annotated and added to the current labeled dataset and used to update the model in the next iteration. Here we briefly describe several common AL acquisition strategies used to select informative unlabeled data samples during the active learning process. And the comparison is illustrated in Figure 4.5.

Uncertainty-based Uncertainty-based acquisition strategies aim to select the unlabeled data samples with the least confidence (largest uncertainty). Its variations include margin-based, breaking ties, and entropy-based strategies. The confidence score is measured based on the output predictions of the model and is assigned by the acquisition function to each unlabeled question as a measurement of the model's uncertainty. For the QA task, the model outputs multiple candidate answer spans, among which the candidate span with the highest probability value is the predicted answer for the question. The probability value of the predicted answer span is considered the confidence score for the question. The least confidence uncertainty-based strategy selects questions whose predicted answer span has the lowest probability value.

Density/Clustering This acquisition strategy finds representative data samples by comparing the similarity among data samples. The goal is to find a batch of samples that represents well the overall pattern of the unlabeled data pool. A typical method for selecting the most representative points is to project data into an embedding space and then measure the similarity of data samples. The embeddings are generated from the BERT encoder in the model of the current iteration. For each unlabeled question in D_u, the BERT encoder generates embeddings for the concatenated questions and their context. Then using a clustering algorithm such as K-Means group the questions into clusters. The k selected questions for annotation are sampled proportionally from each cluster per the size of the cluster.

Maximal Diversity Similar to the clustering sampling strategy, it exploits heterogeneity in the feature space by projecting both unlabeled questions and labeled questions to the feature space using the embedding of the fine-tuned model. The goal is to query the unlabeled questions that have maximally distant from the labeled ones. Considering the heterogeneity of the labeled questions, we cluster the labeled questions in D_l, and return unlabeled questions with maximum distance to their closest labeled question centroids.

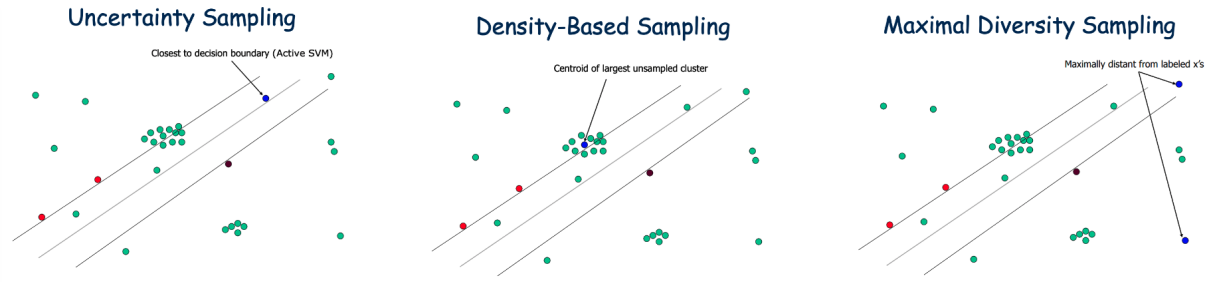


Figure 4.5: Common Active Learning Strategies (Figure taken from [13])

4.4.3. Perturbation Active Learning

We introduce a novel acquisition strategy, PAL, which considers the robustness of the model as a signal of informativeness. We hypothesize that a robust model should produce similar probability distributions on the original context's part after perturbation of the context with an additional distracting sentence. From another aspect, among unlabeled questions D_u, the unlabeled instances which produce very different predictive likelihoods under perturbations should be more informative if labeled. Thus, our PAL strategy adds such questions during data acquisition to improve the robustness of the current model.

Creating Perturbation for Unlabeled Candidates The first step of our PAL acquisition strategy finds the distracting sentence from the context of the most similar labeled questions using the embedding of the fine-tuned model. Specifically, we use the *[CLS]* token embedding from $Encoder_t$ of the current fine-tuned model M_t to represent the contexts for all the questions in $D_{(t)}l$ and $D_{(t)}u$. We use a K-Nearest-Neighbors (KNN) implementation in order to query the questions with the most similar contexts in $D_{(t)}l$ for each candidate $u_i \in D_{(t)}u$[1]. Our distance metric $d(.)$ is Euclidean distance. To find the most similar context in $D_{(t)}l$ for each u_i, we select the top k instead of selecting a predefined threshold. This way, we create a neighborhood set N_c that consists of k most similar contexts in $D_{(t)}l$ for each u_i. We then break the contexts in N_c into sentences and get the *[CLS]* token embeddings of each sentence with $Encoder_t$. Similarly, we compute the Euclidean

[1]The benchmark dataset SQuAD contains many questions with the same context, which we explicitly excluded when choosing the most similar sentence as the distractor.

distance between each of these sentences to the context of u_i using embedding and choose the most similar sentence as the distractor sentence. A perturbed instance $u_i^{'}$ is generated by appending the distractor sentence to the original context of u_i.

Scoring Robustness to Perturbation of Unlabeled Candidates In the second step, we use the current trained model M_t to obtain the output answer start and end token probabilities for each of the candidate unlabeled question u_i and its corresponding perturbed question $u_i^{'}$. Then we compute the Kullback-Leibler (KL) divergences[2] in the model predictive probabilities between each (u_i, $u_i^{'}$) pair for answer start and end tokens separately, and the negative sum of them is referred to as the robustness score s_{u_i} for u_i.

Rank Unlabeled Candidates and Select Batch We apply these steps to all candidate questions $u_i \in D_{(t)}u$ and obtain a score s_{u_i} for each. A lowest s_{u_i} score indicates that the unlabeled question u_i whose predictive probability distributions diverge the most from their predictions after adding the distracting sentence, suggesting that it is more sensitive to the distracting perturbation, making it a good candidate for data acquisition to select to improve the robustness of the current model. To this end, our acquisition function selects b unlabeled questions that have the lowest score s_{u_i} from the current unlabeled pool $D_{(t)}u$.

4.5. Experiments and Results

We conduct our evaluation using the Stanford Question Answering Dataset (SQuAD) [142]. Before we feed the texts of each question-context pair to the model, we pre-process them with *'Tokenizer'* from *HuggingFace's* [3] *Transformers* library, which tokenizes the inputs to convert the tokens to their corresponding IDs in the vocabulary. Then we compute the offset mappings that map tokens to character positions in the context and save the start

[2]The vanilla Kullback-Leibler divergence is not a symmetric metric, instead, we used the symmetrised divergence: $D_{\mathrm{KL}}(P \parallel Q) + D_{\mathrm{KL}}(Q \parallel P)$.
`https://en.wikipedia.org/wiki/Kullback-Leibler_divergence`

[3]https://huggingface.co

and end positions of the answer span as the labels. We call *AutoModelForQuestionAnswering* to initiate a model that loads the BERT-base encoder with a question-answering header. This model predicts the start and the end position of the answer, and calculates the cross-entropy loss by taking the difference between the predicted and labeled start and the end positions.

Following Algorithm 1, we experimented with the application of the AL strategies for choosing the instances to be labeled in each iteration during fine-tuning BERT for answer prediction. In the beginning, all questions in the training dataset are considered to be unlabeled. To start, 1% of the dataset is selected and labeled for training the model. In the following iterations, we continue fine-tuning the model on the newly labeled 10% of the rest of the unlabeled dataset selected by the active learning acquisition functions. This process was repeated until $D_{(i)}u$ is empty. Table 4.1 compares the performance of our PAL strategy with several common AL strategies at every n-th training step (with a batch size of 12).

AL Strategies	200	300	400	500	600	700	800	AUC
Confidence	52.4	66.5	71.7	**74.8**	**77.2**	**77.5**	79	71.3
Clustering	53.6	67.1	68.4	73.6	76.3	76.5	77.7	70.5
Diversity	50.4	65.7	71.4	71.6	75.6	74.7	78.2	70
PAL	**57.7**	**70.1**	**72.6**	74.1	76.2	78.5	**79.9**	**72.7**

Table 4.1: Fine-tuning the BERT-base model with various AL acquisition strategies for the QA task. The F1 scores are evaluated at every n-th training step on the SQuAD dataset[4].

The experimental results show that:

1. In general, the uncertainty-based strategy outperforms the other two common AL strategies as it always searches for the "valuable" samples around the current decision boundary, but the optimal decision boundary cannot be found unless a certain number

[4]SQuAD dataset is a large annotated QA dataset, containing 100,000+ question-answer pairs on 500+ articles. For a context, there are one or more question-answer pairs are associated. For the purpose to demonstrating the benefits of active learning on a smaller annotated dataset in general, we experiment fine-tuning on subset of the whole training dataset, consisting of about 20,000 unique question-answer pairs, one pair per context.

of samples have already been labeled, which explains its lower performance at the beginning.

2. The clustering strategy performs better when the number of labeled samples is very small, while the uncertainty-based criterion usually overtakes the clustering strategy afterward. This is expected and consistent with the observations from previous works [208]. The main reason is that the clustering strategy could obtain the entire structure of a database in the beginning stage, but it is insensitive to the data samples that are close to the decision boundary and requires querying a large number of instances before reaching the optimal decision boundary.

3. Both uncertainty-based and clustering sampling strategies can only guarantee their optimal performance over a period of time in the entire AL processes, and the optimal period differs.

4. The diversity-based sampling is less effective than the other two sampling strategies for the QA task.

5. Our PAL method is effective and outperforms the commonly used AL acquisition strategies on a fixed annotation budget.

4.6. Discussion

The main challenge in the development of novel AL methods is that there is no objectively superior way to determine how informative each data label is. As clustering and diversity-based methods are highly representation-dependent, there are two main approaches to acquiring data for active learning - uncertainty sampling and representation-based sampling. Uncertainty sampling relies on the model's predictive confidence to select difficult examples, while representation-based sampling exploits heterogeneity in the feature space. Algorithms based on uncertainty may end up choosing uncertain yet uninformative repetitive data, while representation-based methods may tend to select diverse yet easy examples for the model [153]. The two approaches are orthogonal to each other, since uncertainty sampling is usually based on the model's output, while representation exploits information

from the input (i.e. feature) space [114]. Our PAL acquisition method utilizes both the input feature (embeddings in the feature space) and the model's output (predictive probability distributions) to select the most informative instances.

Instead of difficulty or heterogeneity, we consider the robustness of the model against perturbation as a signal of informativeness for active learning, by choosing the unlabeled questions where the model produces very different predictive likelihoods after perturbation. Hybrid data acquisition functions that combine uncertainty and representation-based sampling has also been presented in several works [207, 154, 210, 167]. Incorporating our PAL strategy with both uncertainty and representation-based sampling is promising, which integrates the advantages of multi-criteria as they are different informative dimensions.

In addition, it is also possible to combine AL and self-learning to further reduce the amount of annotation needed for training a QA model. Self-learning is a commonly used semi-supervised learning technique to exploit unlabeled data. The self-learning method finds highly reliable instances based on its own predictions of the current model to teach itself, while AL queries the most informative instances based on query acquisition algorithms. The combination of AL and self-learning updates the training data with the most informative and highly reliable instances.

4.7. Summary

State-of-the-art ODQA systems commonly contain a deep RC model. Deep RC models are not robust enough and are extremely vulnerable to adversarial examples. To improve system performance, a large amount of labeled training data is often required. However, a large and high-quality set of labeled training data is not always available, and in practice, the acquired labels are not only expensive and also noisy. To improve the practical applicability of QA systems, the goal is thus to attain the best gain from this limitation. Active learning is a machine learning methodology in which a model is trained using a set of most informative labeled data that is selected by the model with certain acquisition strategies. It is a widely-used training strategy for maximizing predictive performance subject to a fixed annotation budget. Common acquisition strategies for AL are based on uncertainty and representations. In this work, we introduce a novel active learning strategy, PAL, for the QA

task. The strategy works by sampling unlabeled questions where the model produces very different predictive likelihoods after perturbation with a distractor sentence. We compare our PAL strategy to three other AL acquisition strategies: model confidence, clustering, and question diversity. The experimental results confirm the effectiveness of our PAL strategy, compared to several widely adopted AL acquisition strategies.

CHAPTER 5

CONCLUSION

5.1. Summary

In this dissertation, we address Open-domain question answering (ODQA), which is a task of answering general domain questions using a large collection of texts (e.g., Wikipedia). Most ODQA systems rely on a powerful information retrieval method to efficiently retrieve potentially relevant information as context from the given large corpus and then use a reading comprehension system to extract an answer from the context. It requires the system not only to be able to accurately identify the information that is relevant to the question but also to be able to understand the question and the documents well enough to be able to identify the answer string within the retrieved documents. It is even more challenging when coping with complex questions that require reasoning over multiple pieces of evidence and synthesizing information from multiple sources. With the development of technologies, the focus of research has shifted to complex QA, in particular, Multi-Hop Question Answering (MHQA) that requires synthesizing information from multiple sources. This is significantly more challenging than its single-hop counterpart. Since evidence secondary hops away from the question could share few common words and little semantic relation with the question, the task to retrieve and re-rank such contexts is challenging and suffers from semantic drift. Further challenges for solving MHQA are to output not only the answer but also the supporting facts that led to the answer. This is crucial for the wide adoption of QA systems for most high-stake applications such as finance, law, and healthcare.

We start by first giving a brief background of ODQA, then discussing several key challenges to improve ODQA systems, especially for complex questions. We present several solutions to deal with the challenges by improving individual components in the typical architecture of ODQA shown in Figure 1.2. Section 2 presents an improvement on the question processing component for MHQA, by reformulating the questions with identified bridge phrases, before presenting it to the QA model to boost the possibility of finding the relevant evidence for downstream performance. And Section 3 focuses on improving the re-ranker of the retrieval-reranking component in the architecture for MHQA. It presents two ensemble models to combine diverse relevance signals for better capturing the relevance of the evidences for re-ranking. Section 4.1 introduce an AL learning strategy to improve the reader module to advance MHQA or ODQA in general.

In Section 2, we introduced an unsupervised graph-based approach, **STEP** (**S**teiner **T**ree-based **E**xpansion **P**hrase identification), to identify bridge phrases for MHQA by dynamically connecting scattered information pieces from the question and the available context, organizing them as a noun phrases graph, modeling the bridge phrases identification task as a Steiner tree problem, and identify the minimal subgraph that connects all question phrases, in which the Steiner points are extracted as bridge phrases. Our method is modular and agnostic to any downstream multi-hop QA components. We show the output of our algorithm can be used to expand the query used for evidence retrieval, as well as to provide enhanced context for the answer extraction component, and to provide post-hoc explanations for provided answers. We evaluate our method on the HotpotQA dataset and show that: (a) it improves evidence retrieval for both traditional information retrieval methods such as BM25 and neural retrievers, (b) it yields better answer extraction performance, and (c) it generates better explanations for provided answers.

In Section 3, we call the attention that fine-grained aspects of a model's capability in capturing types of relevance signals should be considered for MHQA evidence re-ranking. Specifically, textual entailment should be explicitly taken into consideration when measuring relevance, to result in a more accurate relevant context. The two ensemble models we introduce jointly consider diverse relevance signals. Our experimental results demonstrate that not only are the individual base models necessary in evidence re-ranking but cooperate advantageously to produce a better ranking for MHQA when used together.

State-of-art ODQA systems are commonly based on reading comprehension (RC): a neural model extracts an answer to a question from a given context document. Learning a neural model for RC requires a large amount of annotated training data, yet data availability in practice is limited. And annotating datasets for QA tasks is very costly as it requires intensive manual labor and often expert knowledge, and noisy examples in annotated data are common. This pushes for more robust learning algorithms that maximize performance subject to a fixed annotation budget and are less sensitive to noise. In Section 4.1, we utilized active learning (AL) training strategies to select the most informative unlabeled data samples for efficient fine-tuning on the QA task. We also presented a novel perturbation-based active learning (PAL) acquisition strategy. It selects the unlabeled instances that produce very different predictive likelihoods under perturbations of an additional distracting sentence in the context. The experimental results confirm the effectiveness of our PAL strategy compared to existing common strategies such as uncertainty-based or representation-based sampling strategies.

The contributions of this dissertation are listed as follows:

(1) We introduce a new strategy, for the identification of bridge phrases for multi-hop QA by dynamically connecting scattered information pieces from the question and the available context, organizing them as a noun phrases graph, modeling the bridge phrases identification task as a Steiner tree problem, and then applying Takahashi et al. [176]'s algorithm to identify the minimal subgraph that connects all question phrases, in which the Steiner points are extracted as bridge phrases.

(2) Our bridge phrases identification method is modular and agnostic to any downstream multi-hop QA components, i.e., it can be used as query expansion for evidence retrieval; it can be used to generate an enhanced context for answer prediction, and it can be used to generate post-hoc explanations for given answers. We show how the output of our algorithm can be used to expand the query used for evidence retrieval, as well as to provide enhanced context for the answer extraction component. Lastly, we show how our method can be used to provide post-hoc explanations to provide answers.

(3) We evaluate our bridge phrases identification method on the HotpotQA dataset [203], and show that: (a) it improves evidence retrieval for both traditional information retrieval

methods such as BM25 [181] and neural retrievers [145], (b) it yields better answer extraction performance, and (c) it generates better explanations for provided answers.

(4) We call attention to the fine-grained aspects of relevance for multi-hop QA evidence re-ranking. In particular, textual entailment relation should be explicitly taken into consideration along with semantic equivalence when measuring the relevance of candidate evidences for complex question answering, in order to cover a more accurate relevant context that is not only lexical match based but also require inference to identify.

(5) We present two ensemble ranking models that combine diverse relevance signals captured by three off-the-shelf base models. Our experimental results demonstrate that not only are the individual base models necessary in evidence re-ranking but cooperate advantageously to produce a better ranking for multi-hop questions when used together.

(6) We empirically show the effectiveness of our ensemble ranking models by evaluating on the HotpotQA dataset and show they outperform all the base models and also several ensemble baselines.

(7) We introduce a perturbation-based active learning acquisition strategy that involves adding a distractor sentence to the QA context and selecting question instances that are most sensitive to the perturbation to build a QA reader model with less annotation and more robustness.

(8) We conduct an empirical study on a combination of AL techniques and BERT-based deep learning reader model for the QA task, and demonstrating our PAL strategy is more effective than several commonly used AL strategies.

5.2. Future Directions

Taking a step back and reflecting, the field of ODQA has seen an incredible amount of progress in recent years, especially since BERT debuted in 2018. But there are still many open questions, less explored directions, and much more work to be done. Below, I list and discuss some of these directions for future research which I believe could be promising to be explored.

5.2.1. Knowledge Base as External Resources

While an enormous amount of information is encoded in the vast amount of text on the web, information obviously also exists in more structured forms. Answering natural language questions by mapping them to a query over a structured database is known as KBQA (knowledge-based question answering) [83]. It relies on a large-scale structured knowledge base, such as Freebase, DBpedia, or Wikidata, which generally consists of many knowledge triples. KBQA models identify the KB triples related to the questions to locate the answers. Compared to large text corpus, KBs have low coverage and are sparse and incomplete, which brings an upper bound to the performance of KBQA models. But they are easier and more efficient to extract answers from if related knowledge can be found directly. Most QA models assume answering questions is possible using a single information source, either a large text corpus (e.g. Wikipedia articles) or the KBs alone. In practice, some questions are better answered using text, some questions are better answered using KBs, and some complex questions are better answered when combining both types of information. Enhancing the ODQA model with rich knowledge so that it does not assume information is in a KB or retrieved documents provide a promising direction to access and utilize the tremendous amount of information on the Internet and large-scale KBs. It helps break through the limitation that the scope of knowledge required to answer questions is restricted to the given context, thus has wide applications in a more practical setting and is an ongoing research challenge. Two ways to develop QA models to incorporate external knowledge from KBs: 1) Leverage relevance knowledge to guide the retriever-reranker or reader of existing ODQA systems; Several important research questions include: whether to route an incoming question using a single information source and which to use, or using a combination of text corpus and KBs; how to determine the relevance between knowledge and context to select possible related knowledge from the KBs. 2) Synthesize information from the two heterogeneous sources and build effective QA models on the combined information. To effectively answer complex questions using a combination of text and knowledge bases, it is important to figure out how to encode and integrate knowledge with textual representations of the context and questions, and disambiguation is required to avoid misleading irrelevant information.

5.2.2. Human-in-the-loop Interactive Learning

Current ODQA systems do not provide easy mechanisms for the end-users to correct problems when models make mistakes. Another area of future research is to apply an interactive learning process for building QA models to allow them to optimize their learning behavior through interaction with humans. Specifically, with a human user in the learning loop, it potentially allows for more accurate models as users are able to review outputs, make corrections, and provide input meant to improve the QA model. The open research questions include 1) how to utilize visualization techniques to visualize and present the model and the prediction details of each step during the learning process to the users. 2) design an interactive interface to enable human supervision and intervention in the learning process of the model, including dynamically filtering information shown to the users, and providing hints and guidance for the users to provide feedback. For instance, people usually use a series of queries to search for the right answer, but it's not natural for a computer program. Can the model reformulate the query of the same question and issue them concurrently as a simulation? Besides the answer labels as direct supervision, can human users provide extra information (such as feedback, interactions, and preference) as rich guidance to further help train the model? For example, reasoning path for MHQA, rankings of the relevance of candidate evidences, ranking of candidate answers, and corrections of answers and evidences, etc. In addition, would meaningful human interactions such as scrolling, search history, and clicks be useful from the perspective of evolving the model? 3) how to incorporate human interaction, preference, and feedback into an existing QA model in effective ways? 4) how often or how to determine the critical steps to engage human intervention to make meaningful progress in a cost-efficient manner?

5.2.3. More Challenging QA Tasks

The progressive development of QA systems follows a rule of "creep-before-you-walk". Researchers begin with simple questions which can be answered with a single document. Over the past several years, complex questions that require synthesis multiple pieces of evidence to infer the answer have been the focus of much attention. Looking into the future, researchers have proceed to more challenging questions. Here I list several challenging QA

tasks that ongoing research attempts to deal with and are important directions for future work.

Various complex QA This dissertation focus on the bridge entity type of complex question, there are various other types of challenging complex questions, including QA with common sense reasoning, arithmetic MHQA, Temporal MHQA, etc. More focus on these such challenging questions could also lead to better QA systems.

Commonsense Reasoning aims to empower machines with the human ability to make presumptions about ordinary situations in our daily life [104]. How machines can be taught to make commonsense inferences, particularly in the MHQA setting, is still a challenging task. For some questions, commonsense knowledge might additionally be required to perform a reasoning step or the answer does not exist explicitly in the text and has to be inferred with commonsense reasoning. Even though recent ODQA systems based on large PLM has achieved great performance improvement on most QA benchmarks. These systems do not possess common sense. [14] argue that a PLM may not be able to capture every grounded common-sense knowledge even with large corpora. More techniques for incorporating commonsense reasoning into the ODQA system can be a promising direction to explore.

Arithmetic QA aims to answer numerical questions that require a QA system to perform the arithmetic computation to solve them. For example, answering "How many yards did Kasay kick in total?" from the context "John Kasay hitting a 45-yard field goal ... with Kasay again hitting a 49-yard field goal ..." requires some arithmetic computation. While current ODQA systems have difficulty performing arithmetic operations, [131, 69, 161]. combining the latest neural methods and symbolic reasoning is promising to enable the capability of solving arithmetic comparisons and computations for arithmetic QA.

Temporal QA is a task of answering questions that contain a temporal expression, a temporal signal, or whose answer is some kind of temporal information, e.g., a date [78]. Understanding time is important, and vagueness with respect to time is inherent in natural language. Building a general time-aware ODQA system that can help acquire temporal information (such as timestamps, time scope, and time periods) and figure out the timeline or temporal order from cues in the text could be another promising research direction.

Conversational QA Humans gather information through conversations involving a series of interconnected questions and answers. It is essential for machines to be able to answer conversational questions in order to assist in information gathering [144]. A conversational question contains multiple rounds of interaction between the machine and the human [174]. In order to answer the question, the model asks follow-up questions to the user to get more information, and questions asked in the next turn are related to the history of the conversation. The conversation history acts as part of the context to help with answer prediction. The model predicts the answers when there is enough information after reading the full conversation history. The task of conversational QA can also be seen as an MHQA problem. Each follow-up question can be seen as a retrieval hop, and figuring out the missing information can be seen as a reasoning hop [174]. More research works in conversational QA is vital for enabling natural human-machine communication and leading to the wide adaption of ODQA systems.

Multi-modal QA Understanding audio-visual content and the ability to have an informative conversation about it have both been challenging areas for intelligent systems. There has been an explosion of interest in tasks which combine multiple modalities such as audio, vision, and language together [123]. The task of Visual Question Answering (VQA) is to generate natural language answers from the referenced visual content for the given question. The visual content can be either images or video clips. Extending QA over images to videos adds further complexity and challenges. VQA involves the use of techniques from both natural language processing and computer vision. In order to combine the modalities, different levels of visual representation and textual representation of questions need to be mapped to a common feature space. Example features that can be extracted from visual content include object identification, counting, appearance, interactions, social relationships, and inferences about why or how something is occurring [127]. Multi-modal QA is a more natural and intuitive setting, and it enables applications at the intersection between language and the real world. Though it has received extensive attention, the progress in research direction is still far away from the end, and more research works need to be done to better solving complicated multi-modal QA.

5.3. Final Remarks

To conclude, we present several methods for the advancement of ODQA systems, discuss the challenges and point to future research directions. We hope our work would enable researchers to be informed of the recent progress and also the open research problems in ODQA research area, to stimulate further investigation in this field.

REFERENCES

[1] Abbasiantaeb, Z. and S. Momtazi (2021). Text-based question answering from information retrieval and deep neural network perspectives: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **11**(6), p. e1412.

[2] Agley, J., Y. Xiao, R. Nolan, and L. Golzarri-Arroyo (2022). Quality control questions on Amazon's Mechanical Turk (MTurk): A randomized trial of impact on the USAUDIT, PHQ-9, and GAD-7. *Behavior research methods*, **54**(2), pp. 885–897.

[3] Allam, A. M. N. and M. H. Haggag (2012). The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, **2**(3).

[4] Alromima, W., I. F. Moawad, R. Elgohary, and M. Aref (2016). Ontology-based query expansion for Arabic text retrieval. *Int. J. Adv. Comput. Sci. Appl*, **7**(8), pp. 223–230.

[5] Alvarez-Melis, D. and T. S. Jaakkola (2017). A causal framework for explaining the predictions of black-box sequence-to-sequence models. *arXiv preprint arXiv:1707.01943*.

[6] Antol, S., A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh (2015). Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433.

[7] Armoogum, S. and X. Li (2019). Big data analytics and deep learning in bioinformatics with hadoop. In *Deep learning and parallel computing environment for bioengineering systems*, pp. 17–36. Elsevier.

[8] Arras, L., F. Horn, G. Montavon, K.-R. Müller, and W. Samek (2017). " What is relevant in a text document?": An interpretable machine learning approach. *PloS one*, **12**(8), p. e0181142.

[9] Asghar, N., P. Poupart, X. Jiang, and H. Li (2016). Deep active learning for dialogue generation. *arXiv preprint arXiv:1612.03929*.

[10] Auer, S., C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web*, pp. 722–735. Springer.

[11] Azad, H. K. and A. Deepak (2019). Query expansion techniques for information retrieval: a survey. *Information Processing & Management*, **56**(5), pp. 1698–1735.

[12] Bajaj, P., D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, et al. (2016). Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

[13] Balcan, M.-F. and R. Urner (2015). Active learning. *Survey in the Encyclopedia of Algorithms*, **2**.

[14] Bauer, L., Y. Wang, and M. Bansal (2018). Commonsense for generative multi-hop question answering tasks. *arXiv preprint arXiv:1809.06309*.

[15] Beltagy, I., M. E. Peters, and A. Cohan (2020). Longformer: The Long-Document Transformer.

[16] Berger, A., R. Caruana, D. Cohn, D. Freitag, and V. Mittal (2000). Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 192–199.

[17] Bian, J., Y. Liu, E. Agichtein, and H. Zha (2008). Finding the right facts in the crowd: factoid question answering over social media. In *Proceedings of the 17th international conference on World Wide Web*, pp. 467–476.

[18] Biran, O. and C. Cotton (2017). Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, pp. 8–13.

[19] Bollacker, K., C. Evans, P. Paritosh, T. Sturge, and J. Taylor (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250.

[20] Bordes, A., S. Chopra, and J. Weston (2014). Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

[21] Boreshban, Y., S. M. Mirbostani, G. Ghassem-Sani, S. A. Mirroshandel, and S. Amiriparian (2021). Improving question answering performance using knowledge distillation and active learning. *arXiv preprint arXiv:2109.12662*.

[22] Bouadjenek, M. R., H. Hacid, M. Bouzeghoub, and A. Vakali (2016). Persador: personalized social document representation for improving web search. *Information Sciences*, **369**, pp. 614–633.

[23] Bouziane, A., D. Bouchiha, N. Doumi, and M. Malki (2015). Question answering systems: survey and trends. *Procedia Computer Science*, **73**, pp. 366–375.

[24] Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, **33**, pp. 1877–1901.

[25] Buchanan, E. M. and J. E. Scofield (2018). Methods to detect low quality data and its implication for psychological research. *Behavior research methods*, **50**(6), pp. 2586–2596.

[26] Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96.

[27] Caballero, M. (2021). A Brief Survey of Question Answering Systems. *International Journal of Artificial Intelligence & Applications (IJAIA)*, **12**(5).

[28] Cakir, F., K. He, X. Xia, B. Kulis, and S. Sclaroff (2019). Deep metric learning to rank. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1861–1870.

[29] Callison-Burch, C. and M. Dredze (2010). Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's Mechanical Turk*, pp. 1–12.

[30] Cao, X. and Y. Liu (2022). Coarse-grained decomposition and fine-grained interaction for multi-hop question answering. *Journal of Intelligent Information Systems*, **58**(1), pp. 21–41.

[31] Cao, Y., M. Fang, and D. Tao (2019). BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. *arXiv preprint arXiv:1904.04969*.

[32] Cao, Z., T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136.

[33] Chen, D., A. Fisch, J. Weston, and A. Bordes (2017). Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

[34] Chen, R.-C., D. Spina, W. B. Croft, M. Sanderson, and F. Scholer (2015). Harnessing semantics for answer sentence retrieval. In *Proceedings of the Eighth Workshop on Exploiting Semantic Annotations in Information Retrieval*, pp. 21–27.

[35] Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[36] Choi, E., D. Hewlett, A. Lacoste, I. Polosukhin, J. Uszkoreit, and J. Berant (2016). Hierarchical question answering for long documents. *arXiv preprint arXiv:1611.01839*.

[37] Chowdhary, K. (2020). *Fundamentals of artificial intelligence*. Springer.

[38] Chu, W. and Z. Ghahramani (2005). Preference learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pp. 137–144.

[39] Clark, C. and M. Gardner (2017). Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

[40] Clark, P. and O. Etzioni (2016). My computer is an honor student—but how intelligent is it? Standardized tests as a measure of AI. *AI Magazine*, **37**(1), pp. 5–12.

[41] Cossock, D. and T. Zhang (2006). Subset ranking using regression. In *International conference on computational learning theory*, pp. 605–619. Springer.

[42] Crammer, K. and Y. Singer (2001). Pranking with ranking. *Advances in neural information processing systems*, **14**.

[43] Dasgupta, S. (2011). Two faces of active learning. *Theoretical computer science*, **412**(19), pp. 1767–1781.

[44] De Cao, N., W. Aziz, and I. Titov (2018). Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920*.

[45] Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, **41**(6), pp. 391–407.

[46] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[47] Dhingra, B., Q. Jin, Z. Yang, W. W. Cohen, and R. Salakhutdinov (2018). Neural models for reasoning over multiple mentions using coreference. *arXiv preprint arXiv:1804.05922*.

[48] Dhingra, B., D. Pruthi, and D. Rajagopal (2018). Simple and effective semi-supervised question answering. *arXiv preprint arXiv:1804.00720*.

[49] Ding, M., C. Zhou, Q. Chen, H. Yang, and J. Tang (2019). Cognitive graph for multi-hop reading comprehension at scale. *arXiv preprint arXiv:1905.05460*.

[50] Dor, L. E., A. Halfon, A. Gera, E. Shnarch, L. Dankin, L. Choshen, M. Danilevsky, R. Aharonov, Y. Katz, and N. Slonim (2020). Active learning for BERT: an empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7949–7962.

[51] Esposito, M., E. Damiano, A. Minutolo, G. De Pietro, and H. Fujita (2020). Hybrid query expansion using lexical resources and word embeddings for sentence retrieval in question answering. *Information Sciences*, **514**, pp. 88–105.

[52] Etzioni, O. (2011). Search needs a shake-up. *Nature*, **476**(7358), pp. 25–26.

[53] Fang, Y., S. Sun, Z. Gan, R. Pillai, S. Wang, and J. Liu (2019). Hierarchical graph network for multi-hop question answering. *arXiv preprint arXiv:1911.03631*.

[54] Ferrucci, D., E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. (2010). Building Watson: An overview of the DeepQA project. *AI magazine*, **31**(3), pp. 59–79.

[55] Fu, R., H. Wang, X. Zhang, J. Zhou, and Y. Yan (2021). Decomposing complex questions makes multi-hop QA easier and more interpretable. *arXiv preprint arXiv:2110.13472*.

[56] Fu, Y., X. Zhu, and B. Li (2013). A survey on instance selection for active learning. *Knowledge and information systems*, **35**(2), pp. 249–283.

[57] Gao, L., Z. Dai, T. Chen, Z. Fan, B. V. Durme, and J. Callan (2021). Complement lexical retrieval model with semantic residual embeddings. In *European Conference on Information Retrieval*, pp. 146–160. Springer.

[58] Gillick, D., S. Kulkarni, L. Lansing, A. Presta, J. Baldridge, E. Ie, and D. Garcia-Olano (2019). Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506*.

[59] Gilpin, L. H., D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pp. 80–89. IEEE.

[60] Goldberg, Y. (2019). Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

[61] Gordon, J., B. Van Durme, and L. Schubert (2010). Evaluation of commonsense knowledge with Mechanical Turk.

[62] Guo, L., X. Su, L. Zhang, G. Huang, X. Gao, and Z. Ding (2018). Query expansion based on semantic related network. In *Pacific Rim International Conference on Artificial Intelligence*, pp. 19–28. Springer.

[63] Guu, K., K. Lee, Z. Tung, P. Pasupat, and M. Chang (2020). Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pp. 3929–3938. PMLR.

[64] Guu, K., K. Lee, Z. Tung, P. Pasupat, and M.-w. Chang (2020). REALM: Retrieval-Augmented Language Model Pre. *Training*.

[65] Hartmanis, J. (1982). Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, **24**(1), p. 90.

[66] He, K., X. Zhang, S. Ren, and J. Sun (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

[67] He, W., K. Liu, J. Liu, Y. Lyu, S. Zhao, X. Xiao, Y. Liu, Y. Wang, H. Wu, Q. She, et al. (2017). Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*.

[68] Heilman, M. and N. A. Smith (2010). Rating computer-generated questions with Mechanical Turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's mechanical turk*, pp. 35–40.

[69] Hendrycks, D., C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt (2021). Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

[70] Hermann, K. M., T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom (2015). Teaching machines to read and comprehend. *Advances in neural information processing systems*, **28**.

[71] Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation*, **9**(8), pp. 1735–1780.

[72] Honnibal, M., I. Montani, S. Van Landeghem, and A. Boyd (2020). spaCy: Industrial-strength Natural Language Processing in Python. doi:10.5281/zenodo.1212303.

[73] Huang, P.-S., X. He, J. Gao, L. Deng, A. Acero, and L. Heck (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pp. 2333–2338.

[74] Hwang, W., J. Yim, S. Park, and M. Seo (2019). A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

[75] Indurthi, S., S. Yu, S. Back, H. Cuayáhuitl, et al. (2018). Cut to the chase: A context zoom-in network for reading comprehension. Association for Computational Linguistics.

[76] Jhamtani, H. and P. Clark (2020). Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. *arXiv preprint arXiv:2010.03274*.

[77] Jia, R. and P. Liang (2017). Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

[78] Jia, Z., A. Abujabal, R. Saha Roy, J. Strötgen, and G. Weikum (2018). Tempquestions: A benchmark for temporal question answering. In *Companion Proceedings of the The Web Conference 2018*, pp. 1057–1062.

[79] Jiang, Y. and M. Bansal (2019). Self-assembling modular networks for interpretable multi-hop reasoning. *arXiv preprint arXiv:1909.05803*.

[80] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142.

[81] Johnson, J., M. Douze, and H. Jégou (2019). Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, **7**(3), pp. 535–547.

[82] Joshi, M., E. Choi, D. S. Weld, and L. Zettlemoyer (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

[83] Jurafsky, D. and J. H. Martin (2018). Speech and language processing (draft). *preparation [cited 2020 June 1] Available from: https://web. stanford. edu/˜ jurafsky/slp3*.

[84] Kahaduwa, H., D. Pathirana, P. L. Arachchi, V. Dias, S. Ranathunga, and U. Kohomban (2017). Question Answering system for the travel domain. In *2017 Moratuwa Engineering Research Conference (MERCon)*, pp. 449–454. IEEE.

[85] Kaisser, M. and J. B. Lowe (2008). Creating a research collection of question answer sentence pairs with amazon's mechanical turk. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*.

[86] Karpukhin, V., B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih (2020). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

[87] Kasai, J., K. Qian, S. Gurajada, Y. Li, and L. Popa (2019). Low-resource deep entity resolution with transfer and active learning. *arXiv preprint arXiv:1906.08042*.

[88] Khot, T., P. Clark, M. Guerquin, P. Jansen, and A. Sabharwal (2020). Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8082–8090.

[89] Kim, Y. (2019). Convolutional neural networks for sentence classification. arXiv 2014. *arXiv preprint arXiv:1408.5882*.

[90] Köppel, M., A. Segner, M. Wagener, L. Pensel, A. Karwath, and S. Kramer (2019). Pairwise learning to rank by neural networks revisited: Reconstruction, theoretical analysis and practical performance. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 237–252. Springer.

[91] Kramer, S., G. Widmer, B. Pfahringer, and M. De Groeve (2001). Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, **47**(1-2), pp. 1–13.

[92] Kratzwald, B. and S. Feuerriegel (2019). Learning from on-line user feedback in neural question answering on the web. In *The World Wide Web Conference*, pp. 906–916.

[93] Kushman, N., Y. Artzi, L. Zettlemoyer, and R. Barzilay (2014). Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 271–281.

[94] Kwon, H., H. Trivedi, P. Jansen, M. Surdeanu, and N. Balasubramanian (2018). Controlling information aggregation for complex question answering. In *European Conference on Information Retrieval*, pp. 750–757. Springer.

[95] LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *nature*, **521**(7553), pp. 436–444.

[96] LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), pp. 2278–2324.

[97] Lee, J., S. Yun, H. Kim, M. Ko, and J. Kang (2018). Ranking paragraphs for improving answer recall in open-domain question answering. *arXiv preprint arXiv:1810.00494*.

[98] Lee, K., M.-W. Chang, and K. Toutanova (2019). Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*.

[99] Lei, J., L. Yu, M. Bansal, and T. L. Berg (2018). Tvqa: Localized, compositional video question answering. *arXiv preprint arXiv:1809.01696*.

[100] Li, B. Z., G. Stanovsky, and L. Zettlemoyer (2020). Active learning for coreference resolution using discrete annotation. *arXiv preprint arXiv:2004.13671*.

[101] Li, J., B. Li, and X. Lv (2018). Machine Reading Comprehension Based on the Combination of BIDAF Model and Word Vectors. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering*, pp. 1–4.

[102] Li, P., Q. Wu, and C. Burges (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in neural information processing systems*, **20**.

[103] Liang, Z., Y. Zhao, and M. Surdeanu (2020). Using the Hammer Only on Nails: A Hybrid Method for Evidence Retrieval for Question Answering. *arXiv preprint arXiv:2009.10791*.

[104] Lin, B. Y., X. Chen, J. Chen, and X. Ren (2019). Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*.

[105] Lin, J., R. Nogueira, and A. Yates (2021). Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies*, **14**(4), pp. 1–325.

[106] Lin, X. and D. Parikh (2017). Active learning for visual question answering: An empirical study. *arXiv preprint arXiv:1711.01732*.

[107] Lin, Y., H. Ji, Z. Liu, and M. Sun (2018). Denoising Distantly Supervised Open-Domain Question Answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

[108] Liu, D.-R., Y.-H. Chen, M. Shen, and P.-J. Lu (2015). Complementary QA network analysis for QA retrieval in social question-answering websites. *Journal of the Association for Information Science and Technology*, **66**(1), pp. 99–116.

[109] Liu, M., W. Buntine, and G. Haffari (2018). Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pp. 334–344.

[110] Liu, M., Z. Tu, Z. Wang, and X. Xu (2020). LTP: a new active learning strategy for BERT-CRF based named entity recognition. *arXiv preprint arXiv:2001.02524*.

[111] Liu, X., Y. Shen, K. Duh, and J. Gao (2017). Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

[112] Luo, F. and M. Surdeanu (2022). A STEP towards Interpretable Multi-Hop Reasoning: Bridge Phrase Identification and Query Expansion. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 4552–4560.

[113] Ma, X., K. Sun, R. Pradeep, and J. Lin (2021). A replication study of dense passage retriever. *arXiv preprint arXiv:2104.05740*.

[114] Margatina, K., G. Vernikos, L. Barrault, and N. Aletras (2021). Active learning by acquiring contrastive examples. *arXiv preprint arXiv:2109.03764*.

[115] Mavi, V., A. Jangra, and A. Jatowt (2022). A Survey on Multi-hop Question Answering and Generation. *arXiv preprint arXiv:2204.09140*.

[116] Mihaylov, T., P. Clark, T. Khot, and A. Sabharwal (2018). Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

[117] Min, S., D. Chen, H. Hajishirzi, and L. Zettlemoyer (2019). A discrete hard EM approach for weakly supervised question answering. *arXiv preprint arXiv:1909.04849*.

[118] Min, S., V. Zhong, L. Zettlemoyer, and H. Hajishirzi (2019). Multi-hop Reading Comprehension through Question Decomposition and Rescoring.

[119] Molino, P., L. M. Aiello, and P. Lops (2016). Social question answering: Textual, user, and network features for best answer prediction. *ACM Transactions on Information Systems (TOIS)*, **35**(1), pp. 1–40.

[120] Nakade, V., A. Musaev, and T. Atkison (2018). Preliminary research on thesaurus-based query expansion for Twitter data extraction. In *Proceedings of the ACMSE 2018 Conference*, pp. 1–4.

[121] Nallapati, R. (2004). Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 64–71.

[122] Ng, H. T., L. H. Teo, and J. L. P. Kwan (2000). A machine learning approach to answering questions for reading comprehension tests. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 124–132.

[123] Nguyen, D. T., S. Sharma, H. Schulz, and L. E. Asri (2018). From film to video: Multi-turn question answering with multi-modal context. *arXiv preprint arXiv:1812.07023*.

[124] Nguyen, T., M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng (2016). MS MARCO: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

[125] Nie, Y., S. Wang, and M. Bansal (2019). Revealing the importance of semantic retrieval for machine reading at scale. *arXiv preprint arXiv:1909.08041*.

[126] Nishida, K., K. Nishida, M. Nagata, A. Otsuka, I. Saito, H. Asano, and J. Tomita (2019). Answering while Summarizing: Multi-task Learning for Multi-hop QA with Evidence Extraction.

[127] Pandya, H. A. and B. S. Bhatt (2021). Question answering survey: Directions, challenges, datasets, evaluation matrices. *arXiv preprint arXiv:2112.03572*.

[128] Pang, L., Y. Lan, J. Guo, J. Xu, L. Su, and X. Cheng (2019). Has-qa: Hierarchical answer spans model for open-domain question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6875–6882.

[129] Pang, L., J. Xu, Q. Ai, Y. Lan, X. Cheng, and J. Wen (2020). Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 499–508.

[130] Panovich, K., R. Miller, and D. Karger (2012). Tie strength in question & answer on social network sites. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*, pp. 1057–1066.

[131] Patel, A., S. Bhattamishra, and N. Goyal (2021). Are NLP Models really able to Solve Simple Math Word Problems? *arXiv preprint arXiv:2103.07191*.

[132] Pei, C., Y. Zhang, Y. Zhang, F. Sun, X. Lin, H. Sun, J. Wu, P. Jiang, J. Ge, W. Ou, et al. (2019). Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*, pp. 3–11.

[133] Perez, E., P. Lewis, W.-t. Yih, K. Cho, and D. Kiela (2020). Unsupervised question decomposition for question answering. *arXiv preprint arXiv:2002.09758*.

[134] Peters, M. E., M. Neumann, L. Zettlemoyer, and W.-t. Yih (2018). Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*.

[135] Petroni, F., T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel (2019). Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

[136] Pfannschmidt, K., P. Gupta, and E. Hüllermeier (2018). Deep architectures for learning context-dependent ranking functions. *arXiv preprint arXiv:1803.05796*.

[137] Pîrtoacă, G.-S., T. Rebedea, and S. Ruseti (2019). Answering questions by learning to rank–Learning to rank by answering questions. *arXiv preprint arXiv:1909.00596*.

[138] Poon, H., J. Christensen, P. Domingos, O. Etzioni, R. Hoffmann, C. Kiddon, T. Lin, X. Ling, A. Ritter, S. Schoenmackers, et al. (2010). Machine reading at the university of washington. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pp. 87–95.

[139] Prabhu, A., C. Dognin, and M. Singh (2019). Sampling bias in deep active classification: An empirical study. *arXiv preprint arXiv:1909.09389*.

[140] Qi, P., X. Lin, L. Mehr, Z. Wang, and C. D. Manning (2019). Answering complex open-domain questions through iterative query generation. *arXiv preprint arXiv:1910.07000*.

[141] Qiu, L., Y. Xiao, Y. Qu, H. Zhou, L. Li, W. Zhang, and Y. Yu (2019). Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6140–6150.

[142] Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

[143] Rao, Y., J. Lu, and J. Zhou (2017). Attention-aware deep reinforcement learning for video face recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 3931–3940.

[144] Reddy, S., D. Chen, and C. D. Manning (2019). Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, **7**, pp. 249–266.

[145] Reimers, N. and I. Gurevych (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

[146] Reinders, C., H. Ackermann, M. Y. Yang, and B. Rosenhahn (2019). Learning convolutional neural networks for object detection with very little training data. In *Multimodal scene understanding*, pp. 65–100. Elsevier.

[147] Ren, P., Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang (2021). A survey of deep active learning. *ACM computing surveys (CSUR)*, **54**(9), pp. 1–40.

[148] Riezler, S., A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu (2007). Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 464–471. Association for Computational Linguistics, Prague, Czech Republic.

[149] Riloff, E. and M. Thelen (2000). A rule-based question answering system for reading comprehension tests. In *ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*.

[150] Robertson, S. and H. Zaragoza (2009). *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[151] Robertson, S. E., S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. (1995). Okapi at TREC-3. *Nist Special Publication Sp*, **109**, p. 109.

[152] Rolínek, M., V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius (2020). Optimizing rank-based metrics with blackbox differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7620–7630.

[153] Roy, N. and A. McCallum (2001). Toward optimal active learning through sampling estimation of error reduction. Int. Conf. on Machine Learning.

[154] Ru, D., J. Feng, L. Qiu, H. Zhou, M. Wang, W. Zhang, Y. Yu, and L. Li (2020). Active sentence learning by adversarial uncertainty sampling in discrete space. *arXiv preprint arXiv:2004.08046*.

[155] Sachan, D. S., M. Patwary, M. Shoeybi, N. Kant, W. Ping, W. L. Hamilton, and B. Catanzaro (2021). End-to-end training of neural retrievers for open-domain question answering. *arXiv preprint arXiv:2101.00408*.

[156] Salant, S. and J. Berant (2017). Contextualized word representations for reading comprehension. *arXiv preprint arXiv:1712.03609*.

[157] Samek, W., T. Wiegand, and K.-R. Müller (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

[158] Sarker, I. H. (2021). Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, **2**(6), pp. 1–20.

[159] Schröder, C., A. Niekler, and M. Potthast (2021). Uncertainty-based query strategies for active learning with transformers. *arXiv preprint arXiv:2107.05687*.

[160] Schröder, C., A. Niekler, and M. Potthast (2022). Revisiting Uncertainty-based Query Strategies for Active Learning with Transformers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2194–2203.

[161] Schubotz, M., P. Scharpf, K. Dudhat, Y. Nagar, F. Hamborg, and B. Gipp (2018). Introducing mathqa: a math-aware question answering system. *Information Discovery and Delivery*.

[162] Seo, M., A. Kembhavi, A. Farhadi, and H. Hajishirzi (2016). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

[163] Settles, B. (2009). Active learning literature survey.

[164] Shan, X., C. Liu, Y. Xia, Q. Chen, Y. Zhang, A. Luo, and Y. Luo (2020). BISON: BM25-weighted selfattention framework for multi-fields document search. *arXiv preprint arXiv:2007.05186*.

[165] Shashua, A. and A. Levin (2002). Ranking with large margin principle: Two approaches. *Advances in neural information processing systems*, **15**.

[166] Shekarpour, S., K. Höffner, J. Lehmann, and S. Auer (2013). Keyword query expansion on linked data using linguistic and semantic features. In *2013 IEEE Seventh International Conference on Semantic Computing*, pp. 191–197. IEEE.

[167] Shen, D., J. Zhang, J. Su, G. Zhou, and C. L. Tan (2004). Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd annual meeting of the Association for Computational Linguistics (ACL-04)*, pp. 589–596.

[168] Shen, Y., H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar (2017). Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*.

[169] Siddhant, A. and Z. C. Lipton (2018). Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*.

[170] Simmons, R. F., S. Klein, and K. McConlogue (1964). Indexing and dependency logic for answering English questions. *American Documentation*, **15**(3), pp. 196–204.

[171] Song, L., Z. Wang, M. Yu, Y. Zhang, R. Florian, and D. Gildea (2018). Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*.

[172] Storks, S., Q. Gao, and J. Y. Chai (2019). Recent advances in natural language inference: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*.

[173] Storks, S., Q. Gao, A. Reganti, and G. Thattai (2021). Best of Both Worlds: A Hybrid Approach for Multi-Hop Explanation with Declarative Facts. *arXiv preprint arXiv:2201.02740*.

[174] Sun, H., W. W. Cohen, and R. Salakhutdinov (2021). Iterative Hierarchical Attention for Answering Complex Questions over Long Documents. *arXiv preprint arXiv:2106.00200*.

[175] Swezey, R., A. Grover, B. Charron, and S. Ermon (2020). Pirank: Learning to rank via differentiable sorting. *arXiv preprint arXiv:2012.06731*.

[176] Takahashi, H. et al. (1980). An approximate solution for the Steiner problem in graphs.

[177] Talmor, A. and J. Berant (2018). The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

[178] Tang, J., S. Hu, Z. Chen, H. Xu, and Z. Tan (2022). Incorporating Phrases in Latent Query Reformulation for Multi-Hop Question Answering. *Mathematics*, **10**(4), p. 646.

[179] Tenney, I., P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. Van Durme, S. R. Bowman, D. Das, et al. (2019). What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

[180] Terdalkar, H. and A. Bhattacharya (2019). Framework for question-answering in Sanskrit through automated construction of knowledge graphs. In *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pp. 97–116.

[181] Trotman, A., A. Puurula, and B. Burgess (2014). Improvements to BM25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium*, pp. 58–65.

[182] Tu, M., K. Huang, G. Wang, J. Huang, X. He, and B. Zhou (2020). Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 9073–9080.

[183] Vakulenko, S., J. D. Fernandez Garcia, A. Polleres, M. de Rijke, and M. Cochez (2019). Message passing for complex question answering over knowledge graphs. In *Proceedings of the 28th acm international conference on information and knowledge management*, pp. 1431–1440.

[184] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is all you need. *Advances in neural information processing systems*, **30**.

[185] Voorhees, E. M. et al. (1999). The trec-8 question answering track report. In *Trec*, volume 99, pp. 77–82.

[186] Vrandečić, D. (2012). Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference on world wide web*, pp. 1063–1064.

[187] Wang, A., A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

[188] Wang, H., Q. Zhang, and J. Yuan (2017). Semantically enhanced medical information retrieval system: a tensor factorization based approach. *Ieee Access*, **5**, pp. 7584–7593.

[189] Wang, L., S. Li, W. Zhao, K. Shen, M. Sun, R. Jia, and J. Liu (2018). Multi-perspective context aggregation for semi-supervised cloze-style reading comprehension. *arXiv preprint arXiv:1808.06289*.

[190] Wang, S., M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang (2018). R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[191] Wang, W., J. Auer, R. Parasuraman, I. Zubarev, D. Brandyberry, and M. Harper (2000). A question answering system developed as a project in a natural language processing course. In *ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*.

[192] Wang, W., N. Yang, F. Wei, B. Chang, and M. Zhou (2017). Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 189–198.

[193] Wang, X., N. Golbandi, M. Bendersky, D. Metzler, and M. Najork (2018). Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 610–618.

[194] Wang, Z., P. Ng, R. Nallapati, and B. Xiang (2021). Retrieval, re-ranking and multi-task learning for knowledge-base question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 347–357.

[195] Weston, J., S. Chopra, and A. Bordes (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

[196] Wu, H., W. Wu, M. Zhou, E. Chen, L. Duan, and H.-Y. Shum (2014). Improving search relevance for short queries in community question answering. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pp. 43–52.

[197] Xia, F., T.-Y. Liu, J. Wang, W. Zhang, and H. Li (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pp. 1192–1199.

[198] Xiong, C., S. Merity, and R. Socher (2016). Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pp. 2397–2406. PMLR.

[199] Yang, W., Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin (2019). End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

[200] Yang, Y., D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y.-H. Sung, et al. (2019). Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*.

[201] Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, **32**.

[202] Yang, Z., J. Hu, R. Salakhutdinov, and W. W. Cohen (2017). Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*.

[203] Yang, Z., P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning (2018). HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering.

[204] Yates, A., R. Nogueira, and J. Lin (2021). Pretrained Transformers for Text Ranking: BERT and Beyond. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 1154–1156.

[205] Yih, W.-t., K. Toutanova, J. C. Platt, and C. Meek (2011). Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning*, pp. 247–256.

[206] Yu, A. W., D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

[207] Yuan, M., H.-T. Lin, and J. Boyd-Graber (2020). Cold-start active learning through self-supervised language modeling. *arXiv preprint arXiv:2010.09535*.

[208] Zhan, X., H. Liu, Q. Li, and A. B. Chan (2021). A Comparative Survey: Benchmarking for Pool-based Active Learning. In *IJCAI*, pp. 4679–4686.

[209] Zhang, Y., M. Lease, and B. Wallace (2017). Active discriminative text representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

[210] Zhu, J., H. Wang, T. Yao, and B. K. Tsou (2008). Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 1137–1144.