EXPLAINABLE AND EFFICIENT KNOWLEDGE ACQUISITION FROM TEXT

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Peng Qi
August 2020

Abstract

In a world where almost everything seems to come with a tl;dr,[1] how do we make effective use of the large amount of knowledge that surrounds us and is growing every day? This dissertation focuses on addressing this question for the growing amount of knowledge that is encoded in the form of text with the help of natural language processing (NLP) systems. At a high level, it attempts to tackle two distinct problems: how to enable NLP systems to handle our complex information needs by enabling them to perform complex reasoning, and how to communicate efficiently and ask useful questions in a conversation when the request cannot be stated completely in the form of a single question.

This dissertation presents several distinct approaches to tackle these problems. As these approaches are designed to solve relatively complex reasoning problems on our behalf, it is important to build trust between the user and the system to make sure the system is not just arriving at the right answers, but also doing so for the right reasons. Therefore, all of the approaches presented in this dissertation are also aimed at making the NLP systems involved more explainable for human understanding, and sometimes more controllable in their behavior through the same mechanism for explanation. Specifically, I first present my work on making use of linguistic information to aid the extraction of knowledge bases from textual data. Here, linguistically-motivated techniques combined with neural networks results in a new state of the art on knowledge extraction from text, which enables robust complex reasoning with this knowledge. Then, I move on to describe how we can complement knowledge-based approaches to question answering by extending it into a schema-less text-based setting. Here, we collect one of the first large-scale datasets for open-domain text-based multi-hop question answering, and then I present a system that iteratively retrieves supporting documents from a large collection of text to answer these text-based complex questions. Finally, as we improve NLP systems' capability of performing complex reasoning to answer questions, I note that it is important that they also accommodate our information needs that are sometimes too complex or under-defined to express in a single complex question. To this end, I present how to train NLP systems to ask inquisitive questions to gather knowledge in the face of information asymmetry. This

[1] "tl;dr" stands for "too long; didn't read". Coincidentally, this dissertation also has one that you are reading right now.

not only helps them gather important information to help us resolve our information needs, but also allows systems to reason about how we will gather information in an interaction, and present textual knowledge in a more efficient manner to reduce unnecessary confusion. By defining the informativeness of inquisitive questions and optimizing for information gathering, the resulting system generates curiosity-driven questions to help the system learn more about previously unknown knowledge in a conversation.

By demonstrating and examining these systems, I also hope to show how designing NLP systems for explainability can help us attain various notions of efficiency necessitated by the need to process and present textual knowledge from large collections of text we wish to make use of.

Acknowledgments

I would not have imagined writing this dissertation just a few years ago, and certainly would not have anticipated the challenging but exciting journey that led up to it.

Inspired by Prof. Matt Might's illustration,[2] I have always wanted to pursue a Ph.D. to make my own contributions to help humans understand more about the world around us. It was with the help of my early research mentors like Dr. Xiaolin Hu from Tsinghua University that I was fortunate to have the opportunity to pursue a Master's program in Computer Science at Stanford.

Since joining Stanford, I have been fortunate to be surrounded by great mentors and colleagues, and to embark on a series of adventures. Thinking I would continue doing research in deep learning with a focus on computer vision, I was aspiring to join one of the SAIL groups to conduct some research during my M.S. program. I was fortunately contacted by Andrew Maas, a Ph.D. student working with Prof. Andrew Ng at the time, to work on research projects with him. The only twist in the plot is that the research would be focused on automatic speech recognition. Looking back, this research experience helped re-spark my interest in natural language processing, and introduced me, for the first time, to the Stanford NLP Group that I would later happily identify as my academic family. I was also fortunate to get to know Dan, who was teaching his speech class for the first time in years – but more on Dan later. I had a lot of fun coding up neural networks in NumPy and Cython and manually deriving back-propagation with wonderful colleagues like Ziang and Awni.

When I started my Ph.D. in 2015, I was welcomed by members of the NLP Group, all of whom were extremely helpful with my transition into an independent researcher. Many offered me great advice that I am still benefiting from to this day. Among others, Gabor was a constant source of encouragement in the office, not only of my research ideas, but also of good research practices. Gabor has convinced me that there is always a smaller proof of concept that does not take forever to run even in the age of neural networks, which has probably saved me countless hours waiting for experiments to fail. I also owe a lot of my skills of clearly presenting my research ideas to Arun, whose comments for the draft of my first ACL paper (among other drafts) helped make it a lot more accessible to someone even if they don't work in the exact same subfield.

Amongst great colleagues and friends, some of my peers have had a big influence on my Ph.D.

[2] `http://matt.might.net/articles/phd-school-in-pictures/`

life. Though we had both been in the Master's program in the same year, I have only gotten to know Yuhao in more depth since we worked on KBP together during my Ph.D., who has been a great friend and collaborator since. Besides "wasting time" on side projects like Stanza together, I have learned a lot from him about how to ask and define important research questions through various collaborations, some of which featured in this dissertation. Urvashi has been another source of unrelenting support and friendship during my Ph.D., who is also a lot of fun to work with, starting from our joint rotation project. Among other things, her unbiased feedback on my various half-baked research ideas have greatly helped refine my research taste and presentation skills. Meanwhile, Urvashi and Yuhao are probably the two people that have heard the most rants of mine during my Ph.D. and invariably lifted me up when research life is less than smooth, for which I am eternally thankful and indebted.

I would also like to thank Tim for bringing me along on the ride for various projects on dependency parsing, which stemmed from our shared interest. Although only traces of this collaboration can be found in the dissertation now, it has greatly helped me integrate myself into the research community at my first conferences (thank you, Universal Dependency friends!). Speaking of parsing, I would also like to thank Danqi for her help and feedback on this front, but more importantly the discussions regarding research and career decisions later along the way. Also, special thanks to Robin for "commiserating" with me in the past stressful year on job searching.

I also owe much of my research to the help of great mentors and collaborators. I am grateful to my internship mentors, Jason, Douwe, and Kyunghyun, for their inspiration, patience, support, and confidence in me. I am also fortunate to have met Zhilin and Saizheng during this internship, who, among other things, daydreamed with me about the future of NLP research and later collaborate with me on a project that later inspired half of the work in this dissertation. Kudos to Ashwin and Abi for inviting me to be part of the Alexa Prize team, which opened up my research to more directions, and planted the seed in my heart for one of the projects in this dissertation. Thanks should also go to my collaborators on various projects, including mentors like He, Yoshua, William, and Ruslan, and colleagues Chris L, Matt, Kevin, Xiaowen (Vera), Leo, and Zijian, for their help and trust in me in collaborations.

I am grateful for the great mentors I have had the privilege to work with and learn from at Stanford, all of whom have heavily influenced not just what research topic I pursue, but also how I strive to become a good researcher. Chris Potts has in various scenarios inducted me on how semantics and pragmatics are treated from a formal linguistics perspective. This has helped me a great deal not just in reasoning about compositionality and pragmatics in my own research, but also given me a distinct perspective to approach natural language processing and generation that I would otherwise have taken much longer to digest without his apt help. Percy has been a constant source of inspiration for me to think about big, open problems since my rotation with him in my first year as a Ph.D. student. Although we have only worked closely together for a relatively short period of time, he has instilled in me the habit of always looking for more precise expressions and solid experimentation when communicating my research ideas, rather than relying merely on intuition. This has helped me greatly in carrying out research projects that are more technically challenging and complex in my Ph.D. Dan has been a constant source of encouragement and support starting from my M.S. years, which has greatly helped me build confidence in research in my earlier years. I have constantly been affected by his genuine excitement when he hears about my research ideas and research projects I am collaborating on, be it on speech recognition during a short walk after class, text summarization with Urvashi during our rotation, or more recently on my dissertation projects. Aside from his great capabilities to draw connections between the topic under discussion to very relevant published works in the literature, Dan has taught me a lot about what being a great mentor means. Finally, Chris Manning has been the best advisor I could have asked for and more. A constant voice of reason when discussing my research projects, Chris has taught me by example how to both focus on the big picture and pay attention to the finest detail. I have always been amazed by how he is able to provide feedback for my research not just from a 10000-foot view, but also on the details of model implementation and writing. I am also grateful that he has been supportive of my highly "stochastic" exploration of research topics at times, during which he has constantly pointed out important next steps and milestones when I was too buried into details and missed the big picture. Finally, I am thankful to Chris for how he prioritizes student need, their careers and personal development, as well as how he goes out of his way to care for students' physical and mental well-being.

Last but not least, I want to thank the people without whose love and support I would not have been here writing this dissertation in the first place. To my parents, thank you for everything. Most of all, thank you for your support in what I have chosen to pursue, and for making me the person that I am although it was not always easy for you. It is probably never said out loud enough in a Chinese family, but Mom and Dad, I love you. To my parents-in-law, I am grateful to have you in my life. I also appreciate your support for my pursuit of a Ph.D., among other important matters in my life. Finally, to Xue, I would not be here with my heart so full if it were not for your love, care, and support. Thank you for taking care of me when I am under stress, making me feel you are always around though we have been an ocean part, and accommodating my philosophizing about my research. Thank you for being my pillar, my anchor, and my motivation in life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the dawn of human languages, they have served as the medium through which we record the various phenomena in the world around us, and document our perception and theories about them. It is through language that we pass this knowledge down over human generations, in which written languages (text) play an important role in documenting amd communicating, sometimes across thousands of years, our history, literature, philosophy, and scientific and engineering advances.

To us, it is natural to acquire knowledge from text, and it is a skill we are trained to perform and perfect over years. We do this effortlessly most of the time. For instance, one can easily answer the following question *"When and where do scientists think the common ancestor of modern humans originates from?"* after reading the following excerpt from the English Wikipedia, even without any prior knowledge on the topic:

> In September 2019, scientists proposed that the earliest *H. sapiens* (and last common human ancestor to modern humans) arose between 350,000 and 260,000 years ago through a merging of populations in East and South Africa. (Wikipedia contributors, 2020a)

However, as our understanding of the world widens into more fields of study and deepens into ever increasing levels of detail, it has become increasingly difficult for any individual to acquire all there is to know about any topic of interest from reading. For instance, the first edition of the Encyclopædia Britannica, published in the 1770s, contained only three volumes; today one of the largest online encyclopedias, Wikipedia, has over 6 million entries of textual and multimedia information in its English version alone, which roughly translates to 2,700 volumes of print material, and is still growing at an approximately 3% annual rate, at the time of writing this dissertation.[1] Similarly, in the scientific community, one can observe an "explosion" of written materials on preprint servers such as arXiv. In June 2020, there were around 17,000 submissions to arXiv in total in diverse scientific disciplines, which is much more than any single scientist can catch up with, let alone effectively

[1] Data source: `https://en.wikipedia.org/wiki/Wikipedia:Statistics`.

operationalize in their own research.[2] More broadly, the same applies to generic data available on the web. It is estimated that on average, more than 4 million blog posts and more than 500 million tweets are written and published on the Internet every day, not to mention more than 100 billion emails composed and sent.[3] These represent and contain an enormous amount of textual knowledge that we can make use of, and potentially solve real-world problems.

Can we make use of computer systems to help us in making more effective use of this textual knowledge? More specifically, can we build natural language processing solutions that can scale to large text collections, so that they can acquire knowledge on our behalf and present the result to us and save us the time digging through and reading large piles of text? In this dissertation, I present several approaches to tackle this problem from different angles.

Before diving into the details of what this dissertation is about, I will first review previous work on knowledge acquisition from textual data to provide context for the contributions of this dissertation.

1.1 Background

1.1.1 Question Answering as Knowledge Acquisition

What can we expect from a natural language processing system that acquires knowledge from textual data on our behalf? Different people might have very different expectations and desiderata for such a system. Many would probably imagine a system that understands text just like human does, but is orders of magnitude more scalable than the most avid reader in the world, and less forgetful than the best-trained memory guru. But there is certainly more to acquiring knowledge than just taking in the data and storing it, and recalling portions or all of it when prompted. After all, although a consumer-grade hard drive today can store a compressed version of the English Wikipedia many times over, no one would really consider that the hard drive has *acquired* the knowledge therein, but is rather merely *storing* it.

What is knowledge acquisition, then? Many different definitions can be given, which focus on very different aspects of the matter. However, I argue that most attempts to define whether a system or person has acquired certain knowledge would fall along the lines of the following statement: the system should be able to take the raw stimuli that is said to contain or convey knowledge, process it, and later use it to help solve problems that would not have been possible if said stimuli or underlying knowledge has not been presented.

This definition of “knowledge” and “processing” is intentionally vague, as they are usually opaque to external observers, and could appear as simple as storing the raw stimuli (*e.g.*, who is to say there is much more involved in memorizing the multiplication table?). The second part of the definition

[2]Data source: `https://arxiv.org/stats/monthly_submissions`.
[3]Data source: `https://www.internetlivestats.com/`

is more crucial, because it has an observable outcome, which we could take as the solution to our problem, or a means to assess how much knowledge the system has acquired (*e.g.*, actually performing multiplication of two numbers).

There are many different ways a system that acquires knowledge from large amounts of text can potentially help us solve problems. For instance, as social networks become more and more pervasive, a system that reads the latest feeds on social networks can help local authorities discover and respond to disasters at an unprecedented speed and accuracy, and distribute guidance to those affected to save lives, with scalable access to more up-to-date information about the evolution of situation and resource needs. In times of less suffering, systems that are capable of acquiring knowledge from text can also help us automate elaborate problem-solving processes such as looking up the solution to a concrete problem encountered when programming, or time-sensitive decision-making procedures such as stock trading and IFTTT (if this then that) services.

In this dissertation, I focus on one particular type of problem that can be solved through acquiring knowledge from text—*question answering*. Being able to answer questions from knowledge is an important challenge for the natural language processing community, or even the broader artificial intelligence community, for which many different types of intelligent behaviors are required to give a satisfactory solution. On the other hand, being able to answer questions gives natural language processing systems great flexibility to serve our information needs, because we can virtually express all of our information needs from a textual source of knowledge in the form of a question. Being able to answer questions based on knowledge provided in the form of text allows these systems a particularly important source of knowledge, because abundant amount of text data state facts about the world, relations between entities and events, underlying mechanisms behind observed phenomena, as well as internal (emotional) states of agents that wander around and interact with each other in the world.

Question answering (QA) can take many different forms. From the perspective of how textual knowledge is represented, one could distinguish between text-based QA from knowledge-based QA,[4] where the former represents knowledge in the form of text, and the latter distills it into a symbolic and more structured representation. From the perspective of answer selection mechanism, there is a broad spectrum of multiple-choice QA, extractive QA, QA through semantic parsing approaches, and generative QA. While multiple-choice QA predicts answers from a handful of candidates, extractive approaches typically select the answer from the context that contains the textual knowledge to answer the question directly or with minimal normalization (*e.g.*, normalizing dates like "March 31st, 2020" or "31/03/2020" to a computer-recognizable, unified format). Semantic parsing approaches go a step further to perform calculations or execute program on knowledge that is potentially extracted and normalized from text to arrive at the answer (*e.g.*, *How much older is Stanford University than Tsinghua University?*), while generative QA allows the system to generate virtually any answer to

[4]This term ("knowledge-based question answering") is taken from Jurafsky and Martin (2019, Chapter 25).

Categorization criterion	Categories
Form of knowledge representation	knowledge-based✓, text-based✓
Answer selection mechanism	multiple-choice, extractive✓, semantic parsing*, generative
Mode of interaction	single-turn✓, multi-turn✓
Nature of answers	factoid✓, non-factoid
Reasoning complexity	single-step, multi-step✓

Table 1.1: Different ways to categorize question answering tasks and systems. ✓ indicates directions that are the primary focus of this dissertation, and * indicates directions that this dissertation touches upon but does not focus its technical contributions on.

any given question.

To answer the questions we might have from a textual source of knowledge, question answering systems need to be able to interact with us through an interface. From the perspective of mode of interaction, QA systems can be broadly categorized as single-turn and multi-turn. Single-turn systems treat each question as query that is independent from other questions it might have been asked in the same interaction; multi-turn systems instead take a more interactive approach to answer questions in the context of conversations. Last but not least, from the perspective of the nature of the answers, QA systems can be roughly categorized as focusing on factoid QA and non-factoid QA. While factoid QA systems typically treat the textual data as a more direct source of knowledge that states facts about the world we are interested in getting answers about, non-factoid systems place more emphasis on the underlying mechanisms of observed facts, and instead lean towards questions that are more likely to start with “how” or “why”.

In this dissertation, I focus on question answering tasks that are factoid and mostly extractive, because these represent some of the most common scenarios in which QA systems can help us acquire knowledge from text in practice. Specifically, a strong focus of this dissertation is to enable QA systems to perform multi-step reasoning, so that they can not only answer simple questions like *“When was Albert Einstein born?”*, but also *“When was the physicist who pioneered the theory of relativity born?”*. This is an important capability for QA systems to be able to help us make the most out of the knowledge encoded in large collections of text, because not every question will correspond to a short snippet of text containing the answer in the form of a statement. I approach this problem from a knowledge-based perspective on the one hand, by building better systems to extract knowledge reliably from text, but also from a text-based perspective on the other hand, where the problem is a lot more challenging but practically useful. In my text-based solution, I take inspiration from long-standing research in information retrieval that studies complex information needs, and build a system that, instead of relying on the human user to decompose and execute the information need step by step, automates the process with the help of natural language processing techniques. Taking this approach one step further, I also investigate how QA systems can better resolve our complex information needs by simulating information acquisition in a conversation, which is a crucial capability for NLP systems when interacting with users, because users might not always have their information needs fully specified and clarified ahead of time. I summarize the different types of question answering tasks in Table 1.1, and highlight the different categories that this dissertation touches upon. In the following subsections, I will begin by reviewing a brief history of open-domain text-based QA and knowledge-based QA, the foci of this dissertation, before providing a broader context of examples for other types of question answering tasks.

1.1.2 Open-domain Text-based Question Answering

Question answering from text has long been a topic of the community's research interest, dating back to the early days of computer science. One example of this task is answering the question about the origin of *Homo sapiens* in the beginning of this chapter. In this dissertation, I refer to **question answering** the task of finding the answer to a question from a large collection of text.

Simmons et al. (1964) were among the first to present a complete system that answers questions from textual statements by aligning the dependency representation (a form of syntactic representation) of the question and the statement, and looking for the alignment of the question word as the answer. For instance, *"What do worms eat?"* transforms to *worms* $\leftarrow$ *eat* $\leftarrow$ *what* in their formulation, and *"Worms eat grass."* transforms to *worms* $\leftarrow$ *eat* $\leftarrow$ *grass*. Therefore, the answer to the question is *grass*, because it matches with the WH-word in the original question after alignment. However, such systems were often designed and tested on a relatively small amount of text data; their practical application on real text collections remained limited.

Around the turn of the millennium, question answering sparked new research interest with the growing amount of textual data available with the rise of the Internet. The Text REtrieval Conference (TREC) organized a number of question answering competitions to promote the development of question answering systems that can directly find answers in a large collection of text documents.[5] To cope with the large amount of text data available, most systems took a two-stage approach, where they first used information retrieval techniques to locate documents or paragraphs that are more likely to contain the answer to a given question, then use more fine-grained systems to sift through these retrieved documents to return the final answer (see Figure 1.1 for an illustration). This latter task of answering a question given a small, fixed collection of textual context is also sometimes referred to as **reading comprehension**, to distinguish it from the original task of finding the answer from a potentially much larger set of documents. These competitions nurtured some of the first practical question answering systems, but they were often still limited in the types of questions they could robustly answer given the reading comprehension technology available at the time.

Around the early 2010s, better graphics processing units (GPU) and deep neural networks greatly

[5] `https://trec.nist.gov/data/qamain.html`

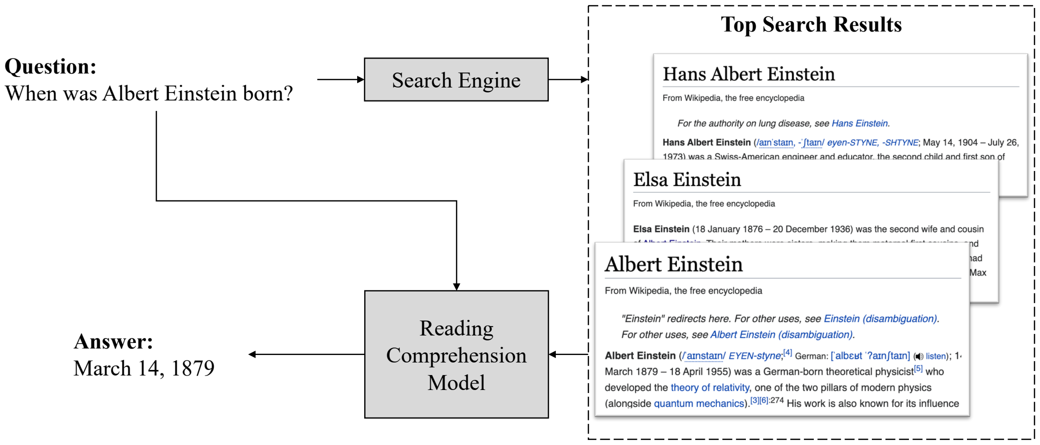


Figure 1.1: An illustration of an information-retrieval-based question answering system answering the question *"When was Albert Einstein born?"* on the English Wikipedia. The system first queries a search engine indexing Wikipedia for relevant documents to the question, then attempts to extract an answer from top search results conditioned on the question with a reading comprehension model.

enhanced the machine learning toolkit, which in turn rendered the lacking amount and variety of question/answer pairs one of the major limitations for development of better reading comprehension systems. At the same time, free, high-quality textual data (*e.g.*, Wikipedia) and easy-to-use crowd-sourcing technology (*e.g.*, Amazon Mechanical Turk) have become more accessible. The research community took advantage of these, and produced larger question answering datasets. Rajpurkar et al. (2016) collected one of the first datasets featuring more than 100,000 question/answer pairs on Wikipedia articles, which fueled the development of many reading comprehension techniques.

These datasets enabled the development of many effective **text-based question answering** models. One effective family of models involve using text encoders such as recurrent neural networks (RNN) to obtain distributed representation for each word in the question and context, and then performing soft alignment between the two (termed "bi-directional attention flow") before using further nonlinear transforms to arrive at the answer (Seo et al., 2017; Clark and Gardner, 2018). Another popular approach leverages large self-attention models called Transformers pretrained on large amounts of text (*e.g.*, BERT from Devlin et al. (2019)), and has been very successful on these reading comprehension tasks, among others. In parallel, taking inspiration from the TREC question answering challenges, Chen et al. (2017) proposed one of the first open-domain question answering systems that follow the retrieve-and-read two-stage approach. When presented a question, the system first finds relevant documents from an bigram retrieval index using the question as search query, then runs a neural-network-based reading comprehension model on top search results. This system combines a computationally efficient text retrieval system with an accurate neural reading comprehension system to extract answers to questions from Wikipedia, and has served as the basis

Background: The two speakers are talking about Spandau Ballet, an English band. Specifically, they are asked to talk about the band's international success and decline in 1983–1989

Question: What was the first indication of Spandau Ballet's success at theinternational level?

Answer: The follow-up album, Parade, was released in June 1984, and its singles were again big successes in the charts in Europe, Oceania and Canada.

Question: What were the notable songs from the album Parade?

Answer: The album's opening song, "Only When You Leave".

Question: How did the opening song do on the charts?

Answer: Became the band's last American hit.

Figure 1.2: An example conversation comprising of questioning and answering between two interlocutors on an article from the English Wikipedia. This example is originally from the QuAC dataset Choi et al. (2018).

of many open-domain question answering system in subsequent work.

The question answering tasks I have mentioned so far focus on answering a single question correctly solely based on a text collection. However, what is most common in human knowledge acquisition is a sequence of questions and answers in context of, and in relation to, each other in a dialogue on a coherent topic. Inspired by this observation, Choi et al. (2018) and Reddy et al. (2019) pioneered the collection of large-scale datasets for **conversational question answering** (see Figure 1.2 for an example of one such conversation). The availability of these datasets also sparked great interest from the community to develop question answering models that incorporate conversational context in multi-turn dialogues (Huang et al., 2019).

Despite their success on reading comprehension tasks with or without conversational context, most of these question answering systems are still heavily reliant on matching local textual patterns. For instance, Jia and Liang (2017) show that the prediction of these systems can be easily altered by adding irrelevant but superficially similar statements to the question into the context where these systems draw answers from. Moreover, it remains unclear whether techniques developed on these datasets will generalize to more complex questions that require multiple supporting facts from multiple documents to answer. For instance, a question like *"Which species originated at the same time from Africa as humans did?"* will be difficult for systems employing a retrieve-and-read two-stage approach, as the question itself does not contain any retrievable cues that might lead to potential answers.

1.1.3 Knowledge-based Question Answering

Aside from answering questions directly from text without explicit processing of the knowledge, another commonly used alternative to make use of textual knowledge is **knowledge-based question**

Species	Relation	Value
Homo sapiens	time of origin	between 348,000 and 258,000 BCE
Homo sapiens	place of origin	East and South Africa

Figure 1.3: An example of knowledge tuples in a knowledge base about the origin of *Homo sapiens*.

answering.

Knowledge-based question answering approaches assume that the knowledge in text can usually be extracted and stored in a structured form. When applied to large text collections, they often involve two steps: knowledge base construction and question answering given this constructed knowledge base. Simply put, knowledge bases are a collection of fact tuples, each roughly corresponding to a natural language statement of a fact about the world this knowledge base is about. For instance, the textual knowledge in our example paragraph about *Homo sapiens* can be roughly expressed as knowledge tuples in Figure 1.3. Entities in such knowledge bases are usually typed, *e.g.*, that *Homo sapiens* is a species, *East and South Africa* are geographical regions, and so on. Once this knowledge is extracted from text, we can use it to answer questions by mapping questions into the same schema of knowledge representation. For instance, the question *"Where do modern humans originate from?"* can be translated into a tuple (*Homo sapiens*, place of origin, ?) where "?" represents a missing value, and answering this question given the knowledge base is akin to a database lookup. Similarly, *"Which species originate from East Africa?"* can be translated into a query tuple (?, place of origin, East Africa).

Knowledge-based question answering systems can be extremely computationally efficient when they are built on top of highly optimized database infrastructure, and are one of the main technologies in commercial question answering systems. Moreover, with entities in knowledge bases properly typed and normalized, they can address complex questions (*e.g.*, *"Which species originated at the same time from Africa as humans did?"*) much more easily by combining factoids in the knowledge base (Zhang et al., 2016; Chaganty et al., 2017). However, for such systems to work well in practice, at least two non-trivial problems need to be solved. One is mapping natural language questions into the knowledge schema representation, *i.e.*, from the question sentence into the tuple with missing values. This problem is known to the research community as **semantic parsing**. It is increasingly challenging with the number of entities and the types of relations that the knowledge base covers. One simple reason is that the same entity or relation can be expressed in many different forms. For instance, *place of origin* can be phrased as *where ... originate from?* or *where ... come from?*, etc. One solution here could be to simplify the interface and put the "burden of mapping" on the user by providing a lists of all entities or relations that are plausibly covered by the knowledge base. To avoid overwhelming users that are interested in querying the knowledge base with the problem of mapping their query into a knowledge schema, the NLP community has also built semantic parsers that convert natural language questions or statements into logical forms that can execute against existing knowledge bases such as Freebase[6] (Berant et al., 2013; Jia and Liang, 2016), SQL queries against existing relational databases (Dong and Lapata, 2016; Zhong et al., 2017), and logical forms that query against knowledge in Wikipedia tables (Pasupat and Liang, 2015).

On the other hand, the arguably more difficult task in building knowledge-based QA systems is constructing these knowledge bases from text in the first place. The simplifying solution I discussed for converting the question would not work, as it would entail that humans are actually doing all the work of constructing the knowledge base, which is time-consuming, if not impractical, in an ever-changing world.

The task of building knowledge bases from text, or **knowledge base population** (KBP), has been a long-standing topic of research interest. The Text Analysis Conference (TAC) has organized an annual KBP challenge over ten years (from 2009 to 2018), where university and industry teams build NLP systems to construct knowledge bases from text.[7] More recently, with the help of crowd-sourcing, Zhang et al. (2017) have collected and released TAC Relation Extraction Dataset (TACRED), one of the largest datasets for this purpose. They demonstrated that this dataset enables neural network models to perform **relation extraction**, the task of extracting the relation between entities in a sentence or document and a crucial step of KBP, more reliably than traditional feature-based approaches. It has also been shown that this improvement directly results in improved KBP performance, as evaluated by the accuracy of answers to a set of questions that are designed to evaluate knowledge bases built on the provided collection of text data. However, these neural models are often overly reliant on the linear order of words in the context, and as a result are sensitive to the noise in textual data when the entities of interest are farther apart from each other.

Before diving into the contributions of this dissertation in Section 1.2, which addresses many of the aforementioned issues, I will introduce a few other forms of question answering in the next subsection, for a more comprehensive overview of developments in this area of natural language processing research.

1.1.4 Other Forms of Question Answering

So far, we have reviewed how the NLP community has approached question answering as a means for making use of the knowledge encoded raw text, by extracting answers or basic supporting factoids from it. However, before we dive into how this dissertation furthers research in this general direction, a few notes are due to complete the big picture.

To begin with, extracting answers directly from text can be undoubtedly limiting, as often times the answer is not immediately contained in the context provided. For instance, consider the following question *"Are Homo sapiens and Homo erectus the same species?"*, to which the answer is either *yes* or *no* (the answer is *no*). However, it is unlikely that these candidate answers are

[6] https://en.wikipedia.org/wiki/Freebase_(database)

[7] See "Knowledge Base Population" at https://tac.nist.gov/tracks/index.html.

directly contained in contexts that introduce basic concepts either about *Homo sapiens* or *Homo erectus*, thus extractive approaches will usually need to model these common, non-extractive answers with a special set of candidate answers. Such approaches scale poorly with the size of the pool of potential candidate answers, though. Therefore, the research community has also extensively studied **generative question answering**, where instead of extracted from the context, the answer to a question is generated as a sequence of words either drawn from a vocabulary or from the context. One prominent example of generative question answering datasets is MS MARCO (Bajaj et al., 2016), in which questions are from actual search query logs of Bing, Microsoft's search engine, and answers were written by humans, instead of extracted from a context. The result is a much more diverse set of answers with more flexible answering strategies. However, this flexibility does not come for free. In particular, the evaluation of free-form text generation is notoriously difficult, and most automatic evaluation metrics have been shown to correlate poorly with human judgement of quality Liu et al. (2016); Novikova et al. (2017).

Aside from generative question answering, answers might also come indirectly from the context, and require some calculation/computation to take form, which I refer to as **quasi-symbolic question answering**. For instance, to answer the question *"How many NBA teams did Michael Jordan play for?"* from his biography requires one to count the number of NBA teams mentioned in it that he joined. Recently, Dua et al. (2019) presented a question answering dataset that features questions of this nature that are collected adversarially against existing question answering datasets like SQuAD, *i.e.*, the questions were worded such that these models will perform very poorly out of the box. As a result a majority of the question-answer pairs collected involve some kind of arithmetic or numeric reasoning (see Figure 1.4 for an example of a question-answer pair with its answering context from this dataset). As can be seen from the example, the nature of these questions is very similar to the kind of questions that semantic parsing addresses, where one can arrive at the answer by generating a program over logical forms and executing it for the result. The main difference is that in this setting, the evidence provided to answer the question is not in a readily structured format, and thus require techniques either similar to knowledge base population to structurize, or more symbolic in general on top of the dominating approach of fuzzy text matching.

Making use of the knowledge in text does not require that we always construct the answer from scratch, either. Aside from extractive, generative, and quasi-symbolic approaches to answer questions from text, one form of question answering tasks that assess NLP systems' text understanding capabilities is **multiple-choice question answering**. Much like the reading comprehension tasks we see in exams, systems are presented context to read ranging from a short article (*e.g.*, RACE; Lai et al., 2017) to books of scientific knowledge (*e.g.*, OpenBookQA; Mihaylov et al., 2018). When constructed carefully, these datasets can usually present significant challenges to test the reading comprehension capabilities and real-world knowledge of NLP models. However, they are also naturally disadvantaged in their output space (the average number of answer choices for each question),

Context: That year, his Untitled (1981), a painting of a haloed, black-headed man with a bright red skeletal body, depicted amid the artists signature scrawls, was sold by Robert Lehrman for \$16.3 million, well above its \$12 million high estimate.

Question: How many more dollars was the Untitled (1981) painting sold for than the 12 million dollar estimation?

Answer: 4300000

Figure 1.4: An example of a question-answer pair and the context from which the answer is derived from, originally featured in the DROP dataset by Dua et al. (2019). As can be seen from this example, the answer to the question cannot be directly found in the context, but requires some computation (in this case, subtraction) of relevant values found in it.

which usually gives systems more leeway to guess the correct answer.

The kinds of question answering tasks I mentioned so far are largely concerned with **factoid question answering**, where the answer to a question is supported by facts mentioned in the text up to paraphrasing and symbolic computation. However, these are far from what we humans are capable of understanding and answering by reading text, and thus also not the only capabilities we would expect NLP systems to possess. For instance, someone with a reasonable grasp of world knowledge should be answer questions like *"How do you cook dinosaur eggs?"* or *"Why does Microsoft Excel allow a maximum of 1,048,576 rows in each sheet?"* even if they probably have never encountered these exact scenarios in their work or life; even if they fail to provide the "correct" answer (assuming there is one to each of these questions), it is usually not difficult to provide statements that have the right bearing on the answer. Recently, the research community has started paying attention to these **non-factoid questions answering** tasks, specifically answering how-questions and why-questions (*e.g.*, Hashemi et al., 2020; Dulceanu et al., 2018). However, the answers to these questions tend to be much harder to evaluate even compared to those of generative question answering, much owing to their open-ended nature. Therefore, some non-factoid question answering datasets resort to evaluation strategies similar to those of multiple choice question answering, in which a few candidate answers are provided, and NLP systems are asked to choose or rank these, and benchmarked on how well their prediction match human annotation on the same task.

1.2 Contributions of this Dissertation

As previously mentioned, this dissertation will be mainly focused on extractive, factoid question answering as a means to make use of the vast amount of knowledge in large text collections. While recent advances in natural language processing have significantly improved reading comprehension models to extract answers from textual context via fuzzy text matching, the problem of answering more complex questions has been largely set aside when these models are incorporated into question answering systems. While most of these models have focused on fuzzy, superficial text matching, to answer more complex questions, our systems need to be capable of (a) distilling and locating the multiple pieces of information that might be relevant to answering a given question, (b) performing the multi-step reasoning needed given this context, and (c) helping the user interactively when the question is under-specified, by reasoning about the user's underlying information needs behind the interaction. To this end, I focus on two related but distinct problems that were previously under-investigated in question answering research:

(a) **Multi-step reasoning.** Many advances in textual question answering have focused on improving the reading comprehension component where the model is provided with the question and a context comprised of a handful of text documents or paragraphs, and the task is to extract the correct answer from this context (Seo et al., 2017; Devlin et al., 2019). To make these systems more applicable to real-world settings where textual knowledge usually comes in the form of a large text collection, the community has also devoted much effort in building and improving the retrieval component that comes before this reading comprehension component (Chen et al., 2017; Wang et al., 2018). However, these systems often still fall short at answering questions that require multi-step reasoning, especially when all clues required are not directly related to the question itself.

    Knowledge-based question answering systems are perfectly suited for these task, although in reality, knowledge bases often suffer from the problem of incompleteness, since the relation extraction systems that build these KBs are still far from perfect. Therefore, in the first part of this dissertation, I present how I address this problem by improving the robustness of relation extraction systems especially to complex sentences that describe the relation between entities. Here, I leverage linguistic information to help relation extraction systems handle long contexts where entity relations might be described, and combine it with powerful neural networks to set a new state of the art on this task.

    In the meantime, although knowledge-based QA approaches are equipped with the capability of multi-step reasoning thanks to their strongly-typed structured knowledge, knowledge bases are still limited by their knowledge schema in what types of questions can be modeled and answered. Thus, in the second part of this dissertation, I present an approach that aims to enable question answering systems to reason for multiple steps to answer complex questions directly from text. As a first step, I collect one of the first large-scale text-based question answering datasets that features questions requiring multi-step reasoning among multiple Wikipedia pages to answer. Then, I propose an open-domain question answering system that iterates between retrieving evidence from Wikipedia and reading the retrieved context to answer complex questions. The result is a system that is explainable in its reasoning steps, and is capable of making use of an efficient off-the-shelf information retrieval system to answer complex questions directly from a large collection of texts.

(b) **Improving communication efficiency.** Being able to answer questions correctly from textual data is a good starting point, but it is by no means all that we should ask from NLP systems. It is important to note that not all of our information needs can be easily expressed in a single complex question sometimes due to missing information. For instance, a customer might not have all the facts about what exact issue they are facing with a product, before they communicate with a helpful customer support agent.

On the other hand, not all of our questions can be satisfactorily with a simple fact without understanding the underlying intent behind the question. To see why faithful, correct answers might not be sufficiently helpful in communication, consider the example dialogues in Figure 1.5. We can probably agree that the answerer appears cryptic and reluctant to engage in the conversation on the left, and most real-world conversations between humans are closer to the one on the right.

Being able to support more complex information needs interactively and communicate efficiently requires us to understand the underlying information need or intent of the questioner. Although conversational implicature (the example in Figure 1.5) and pragmatic inference (Frank and Goodman, 2012) is almost second nature for most of us, our NLP systems are far from being able to achieve this level of social awareness when interfacing with humans.

In this dissertation, I conduct a systematic review of the community's effort toward enabling NLP systems to reason pragmatically and communicate efficiently (Andreas and Klein, 2016; Monroe et al., 2017; Shen et al., 2019), and present one of the first practical approaches to understanding a speaker's information needs by simulating their information acquisition behavior. I propose a framework to characterize how we can think about communicating more information with each conversational turn, as well as two automatic metrics to help quantify this effect. I then present a system that, when trained to optimize these metrics, results in more human-like information-seeking behavior, and thus communicates more effectively with a conversation partner.

Later in this section, I will explain in more detail what concrete contributions this dissertation makes on these topics, as well as its overall structure. Before diving into further details on what this dissertation is about, I would like to take a minor digression and talk about why *explainable* and *efficient* are in its title, and what I take them to mean in this context.

1.2.1 Explainability and Efficiency: What? Why? And How?

What do we mean when we say a system is explainable? Why do we need it, and how should we evaluate/achieve it? My view on this matter is largely in line with what Doshi-Velez and Kim (2017) aptly explained in their essay on what they term *interpretability*. I would personally prefer the term *explainability* in which the system is the agent in the act of explaining its computation and decisions

Question: Do you have plans this weekend?
Answer: Yes!
Question: What are your plans?
Answer: I'm going to sail in the Bay.
Question: Cool! Are you going with with someone?
Answer: Yes, a couple of friends from school.

(a)

Question: Do you have plans this weekend?
Answer: Yes! I'm going to sail in the Bay with a couple of friends from school!

(b)

Figure 1.5: Two hypothetical dialogues about someone's weekend plans. The questioner's original question implies interest in learning some details about the answerer's weekend plans, not just the truth value of it. It is critical that the answerer understand this underlying intent to offer the answer on the right, which is more efficient for communication.

by presenting evidence, rather than us having to *interpret* its behavior after the fact – but this is perhaps a minor technicality on word senses.

Without repeating the arguments of Doshi-Velez and Kim (2017), I will briefly echo and borrow some of the points in my own words, and refer the reader to the original essay for the authors' original views. I think of explainability as the property by which a system presents additional data to supplement and/or support its decision or behavior, so that someone can better understand its behavior. This definition is vague and in fact difficult to further refine, since the nature of explanations is that there is often a better explanation, but it is difficult to prove or define that an explanation is the best. I would venture to add that explanations should further be *communicable* in the information-theoretic sense especially in the case of natural language processing systems, *i.e.*, the explanation should be language-like or can be conveyed succinctly as a set of symbols rather than involving a large amount of continuous values. This is because, unlike other stimuli commonly studied in artificial intelligence, language is largely discrete in nature, and an explanation should probably not require more efforts to process to arrive at the output of a system than starting from the raw input.

Why might we want NLP systems, and AI systems in general, to be explainable? If we were to seriously consider the prospect of deploying these systems for everyday use, then they will inevitably have the power to influence how we perceive the world, if not how parts of it operate. It is natural, then, that we should have the right to demand explanations to various predictions and decisions these systems are generating, and make sure they have the desired properties and behavior. This is especially useful given that our systems are still performing at a level far from human-level accuracy in most scenarios, and their mechanisms not adequately understood or tested in real-world applications. I would further argue that explainability is sometimes also desirable from a development perspective, as they provide us with concrete anchors to understand the behavior of a complex system, and how to improve it. Doshi-Velez and Kim (2017) go into further depth about what systems require explainability more than others, and I refer the reader to their argument.

Now that we have covered what explainability is and why we want it in our systems, how do we evaluate it, and how do we implement it? Instead of providing a single "metric" for explainability, I would like to outline a few desiderata and how one might approach it at a high level. Generally speaking, I believe that aside from being *communicable*, good explanations should at least also be *consistent*, *i.e.*, agree with the system's behavior to the fullest extent possible. This is perhaps obvious, because deceitful explanations do not really help us understand a system's behavior. Moreover, in an ideal world, a system should further be *controllable* via its explanations of its intermediate decisions, if any. This is similar to the *causal* notion of interpretations mentioned by Doshi-Velez and Kim (2017), and allows us not just to passively observe a system's output and its justification, but collaborate with it and help it correct course, if necessary.

Once an explanation satisfies some or all of these criteria, I argue that we can evaluate its quality by observing how succinct the explanation is, and how much uncertainty it reduces in predicting the observed output for an agent that does not share internal states or mechanisms with the one generating the predictions in the first place. The succinctness assumption is natural, as explanations that capture the information of the entire input, or that describe the exact step-by-step instructions of numerical calculations of a computational model is complete but impractical at communicating the mechanism that maps the input to the output. The "different agent" assumption is also important, because an explanation is only meaningful when the exact mechanism to replicate the input-output mapping is not already shared. One example where any explanation is not very useful is when rote computation is involved and both agents know the rules, *e.g.*, the addition of two integers. On the other hand, this assumption also captures the fact that the agent receiving the explanation might approach the input-output mapping using different mechanisms than those of the agent generating it. This allows us to reason about the fact that different recipients might appreciate explanations of different natures on a given task or topic.

One way to express these assumptions slightly more formally is as follows. Consider we are given a system that maps input X to output $Y = f(X)$, and we are handed explanations that depend on this input/output pair $E = e(X, Y)$. To an agent a that is receiving these explanations and trying to predict Y from X and E, the quality of the explanation can be roughly expressed as

$$Q(E|X, Y, a) = I(a(X, E), Y) - \lambda C(E). \tag{1.1}$$

Here, $I(\cdot, \cdot)$ stands for mutual information, $C(\cdot)$ is the cost of communicating the explanation (*e.g.*, its entropy $H(E)$), and λ a constant that controls how overly verbose explanations are penalized. Note that this formulation is agnostic to the process by which the explanation is generated, so good *post hoc* justification is also acceptable provided it satisfies our requirements about communicability and consistency. The implementation of explainable systems can come in many different flavors, and the evaluation of their explanations is highly context-dependent. In this dissertation, I mainly explore implementations that are explainable by design – where parts of the system take communicable explanations or analysis as input, which is in turn either from an external system or an upstream component. This is appealing because (a) it ensures a succinct, communicable format of explanations, and (b) the resulting system is naturally amenable to forward simulation from explanations and external control, thus is consistent with the explanations.

Now that we have covered explainability, what about efficiency? At a first glance, this is much easier to define, as we can easily characterize the *computational efficiency* of a system by the time it takes to achieve the same task. Computational efficiency is indeed an important benchmark if we want our approaches to be able to handle large amounts of textual data (and data of other modalities) in a reasonable amount of time and provide users with the answers they are seeking. However, this is only part of the picture in the context of knowledge acquisition NLP systems. As we approach more and more complex tasks, it has made not only our systems more data-demanding, but also our data collection processes more time- and resource-consuming. To make steady progress on these tasks, we should on the one hand be creative about what data to use and how we use it, but also actively design our systems to be more *statistically efficient*, *i.e.*, that they should be capable of learning the desired task with fewer annotated examples of that task. Last but not least, as I have illustrated in the beginning of this section with Figure 1.5, we should also make sure that our systems are *efficient in communication*, so that they can helpfully get the information we need for us without boring us by taking forever to communicate it. In this dissertation, I demonstrate that designing NLP systems to be explainable is not just a worthy goal to increase transparency, but this principle will also help shed light on how we achieve these various overloaded senses of efficiency to help us navigate the large amount of knowledge that is encoded in text.

1.2.2 Topics and Structure of this Dissertation

The rest of this dissertation is organized into five chapters. In the first few chapters, I present several approaches to more robust and flexible multi-step reasoning with textual knowledge, both from a knowledge base perspective (Chapter 2) and a purely text-based perspective (Chapters 3 and 4). I will then demonstrate how to approach communication efficiency by imitating an agent collecting knowledge from text in a conversation in Chapter 5, before concluding in Chapter 6. More specifically, the topic covered in each chapter is briefly summarized as follows.

In Chapter 2, I present an approach to *relation extraction*, a task that focuses on discerning the potential relation between entities in a piece of text, and a crucial component in knowledge base population and therefore knowledge-based multi-step question answering. This approach combines explainable syntactic analysis with powerful neural networks to improve the robustness of relation extraction systems to longer contexts more effectively. Specifically, a graph convolutional network is applied to to syntactic trees over a sentence that potentially describes the relation between two entities, where the syntactic tree is also processed by a novel, linguistically-motivated pruning technique to maximally retain relevant information while removing noise. The proposed model outperformed various state-of-the-art models at the time that made use of powerful neural models and/or the same syntactic representation. This work was originally presented at EMNLP 2018 (Zhang et al., 2018c), in which Yuhao Zhang made equal contribution.

In Chapter 3, I turn to text-based approaches to multi-step reasoning with textual knowledge to move away from the rigid knowledge schema that knowledge-based approaches are constrained by. I present a large-scale question answering dataset called HotpotQA, which features more than 100,000 crowd-sourced questions that require at least two supporting documents from Wikipedia to answer, along with their answers. This dataset is collected with a novel approach of using the hyperlink graph in Wikipedia to sample pages that are more likely to support natural questions that involve multiple pages to answer. This dataset is also designed to encourage the development of more explainable question answering systems, where we provide supporting facts for each answer in the dataset in the form of sentences that are necessary to arrive at the answer. This work was originally presented at EMNLP 2018 (Yang et al., 2018a), where Zhilin Yang (from Carnegie Mellon University) and Saizheng Zhang (from Université de Montréal/MILA) contributed equally.

In Chapter 4, I present a question answering system that iterates between retrieving more evidence documents and reading the retrieved context to attack the problem of multi-step reasoning in HotpotQA. Compared to the traditional single-step retrieve-and-read approaches, this system is capable of retrieving evidence to answer the question as necessary, instead of relying on context overlap with the original question to gather it. To retrieve supporting facts, this system generates explainable natural language queries to search Wikipedia for supporting documents instead of using the question as search query as is common in previous work. For the task of query generation, I adopt the formulation of a question answering system to extract queries from the context of retrieved documents. Since there is no direct supervision for search queries in HotpotQA, I also propose to build an *oracle system* to generate supervision signal, which is generally applicable to multi-step reasoning questions of arbitrary complexity. The resulting question answering system is capable of answering open-domain questions on arbitrary collections of text only with the help of an efficient text-based retrieval system, the reasoning steps of which are also explainable and allow for intervention. This work was originally presented at EMNLP-ICJNLP 2019 (Qi et al., 2019).

In Chapter 5, I shift gears and discuss how we should reason about and improve communication efficiency of NLP models when serving information to users. Specifically, I propose to start with modeling the intent of the information inquirer by predicting what questions they might ask next. I present a framework for evaluating how much new information is presented and communicated in open-domain information-seeking dialogue, where two agents engage in a conversation about diverse topics, and one is trying to learn something they did not know ahead of time. I propose two novel automatic evaluation metrics within this framework to quantify the amount of new information revealed in a conversation, as well as how specific and relevant to the current conversation an upcoming question utterance is. The proposed metric for quantifying how much new information is revealed is also explainable, which allows for a closer inspection of model behavior to identify areas for improvement. I show that when a question generation system is optimized to consider the proposed metrics, the resulting generated questions are more interesting, human-like, and effective at moving the conversation forward. A preprint of this work is available on arXiv (Qi et al., 2020).

Finally, in Chapter 6, I share some concluding remarks on what has been presented in the previous chapters, and offer notes on possible future research directions.

Chapter 2

Building Knowledge Bases for Multi-step Reasoning

With the development of the Internet, large user-maintained knowledge collections like Wikipedia[1] have made well-curated textual knowledge much more accessible to the average person (amongst other modalities such as visual and auditory). Gone are the days when one needs to visit the library and look up an encyclopedia like the Encyclopædia Britannica, as the latest information about any topic of interest is often just a search click away. However, despite the effectiveness of search engines, we are still required to read potentially large chunks of text to arrive at the answer to questions of interest. Moreover, this textual form of knowledge is not amenable to efficient composition and aggregation to answer more complex questions, *e.g.*, *"How many people live along the East Coast of the United States?"*, unless the answer is already available in textual form. To make efficient use of the knowledge that exists in these large text corpora, one approach is to preprocess it and convert it into a structured form. In the pre-Internet era, people have manually compiled summaries, tables, and compendia for this reason, however, these can still be greatly improved upon with the help of computer systems.

Knowledge bases are one of the most widely used formats to store knowledge in a structured format, which typically comprise of two main components, *entities* that represent real-world entities (*e.g.*, people, organizations, places), their *properties* (*e.g.*, numeric values as a person's age or an organization's employee count, dates as a person's date of birth or an organization's date of creation) and the *relations* that hold amongst these entities or properties (*e.g.*, Person X lives in State Y, or Person X was born on Date Z). One example of publicly available knowledge bases that record facts about the world is Freebase,[2] which has been migrated and merged into a larger project known as WikiData (Vrandečić and Krötzsch, 2014). Since many real-world relations are either binary, or can

[1] https://www.wikipedia.org/
[2] https://en.wikipedia.org/wiki/Freebase_(database)

Subject	Relation	Object	Subject Type	Object Type
New York State	is on coast	East Coast	U.S. State	U.S. coastal area
Massachusetts	is on coast	East Coast	U.S. State	U.S. coastal area
California	is on coast	West Coast	U.S. State	U.S. coastal area
New York State	population	19453561	U.S. State	numeric
Massachusetts	population	6892503	U.S. State	numeric
...	...	...	...	...

Figure 2.1: An example of a knowledge base containing knowledge tuples about which coast U.S. states are on and what their population is. The populations for New York State and Massachusetts are extracted from the Wikipedia articles about each state, respectively. Specifically, the New York State page states "The United States Census Bureau estimates that the population of New York was 19,453,561 on July 1, 2019, a 0.39% increase since the 2010 United States Census" (Wikipedia contributors, 2020c), and the Massachusetts one states "The United States Census Bureau estimated that the population of Massachusetts was 6,892,503 on July 1, 2019, a 5.27% increase since the 2010 United States Census" (Wikipedia contributors, 2020b).

be decomposed into binary ones, knowledge bases are usually represented as a collection of typed 3-tuples of (Subject Entity, Relation, Object Entity), for instance, (Person X, lives in, State Y). This representation is amenable to efficient aggregation and inference among many instances of real-world knowledge. For instance, consider the knowledge base in Figure 2.1 and our question at the beginning of this chapter (*"How many people live along the East Coast of the United States?"*). One should be able to arrive at the answer to this question easily with something akin to a join operation in a relational database on the state entities in this knowledge base, followed by an aggregating function (summation, in this case).

Making use of the existing knowledge in knowledge bases to answer questions requires us to be able to map questions or information requests into the knowledge schema the fact tuples are defined in. This task, known as *semantic parsing* in the NLP community, has been a focus of much research efforts in recent years (Berant et al., 2013; Pasupat and Liang, 2015; Jia and Liang, 2016; Dong and Lapata, 2016; Zhong et al., 2017). On the other hand, another prerequisite for these knowledge bases to be useful for answering our questions about the world is that they should contain fact tuples that actually reflect facts about the world as comprehensively as possible. To this end, the facts in knowledge bases can come from multiple sources, including structured sources like application programming interfaces (APIs),[3] semi-structured sources like tables and lists on the web, and unstrcctured sources like collections of texts. Since textual data is much easier and natural to generate for us humans, it is one of the primary means we use to record and convey knowledge about the world. To make use of the knowledge that is encoded in large amounts of text, we need to be able to extract these relation tuples between real-world entities from raw text. This task is known to the NLP community as *relation extraction*.

[3]For instance, stock prices can be accessed directly from NASDAQ's API: `https://dataondemand.nasdaq.com/docs/index.html`.

In September [1998]$_{\text{Object}}$ Larry Page and Sergey Brin
founded [Google]$_{\text{Subject}}$ while they were Ph.D. students
at Stanford University in California.

Figure 2.2: An example of relation extraction. From this sentence, we can extract that the relation between the subject, *Google*, and the object, *1998*, is that the latter is the date when the former was founded.

Formally, relation extraction involves discerning whether a relation exists between two entities in a sentence (often termed *subject* and *object*, respectively, see Figure 2.2 for an example). Successful relation extraction is the cornerstone of applications requiring relational understanding of unstructured text on a large scale, such as question answering (Yu et al., 2017), knowledge base population (Zhang et al., 2017), and biomedical knowledge discovery (Quirk and Poon, 2017).

One of the most straightforward modeling choices is to make use of word-sequence information in the original sentence, especially the part of the text between the two entities. For instance, knowing that the words between a subject person and an object location contains *"born in"*, one can easily conclude that the relation between the two is that the location is the person's place of birth. Despite its simplicity, these patterns have been extremely useful in relation extraction systems either as hand-coded patterns and sparse features (Angeli et al., 2015). More recently, Zhang et al. (2017) proposed a neural network model that operates on word sequences for relation extraction, where the model is augmented with a position-aware mechanism to extract soft patterns with information about relative position offsets of words in the sentence to the subject and object entities. Despite the success of these sequence-based approaches, they are usually susceptible to superficial changes in the sentence. For instance, *"SUBJECT_PERSON, the second son of PERSON_X, was born in OBJECT_PLACE."* contains many distracting words that are irrelevant between the two entities, and would therefore introduce noise into the relation extraction model. What is more, surface patterns tend to be less reliable in the face of paraphrases and distracting content. For instance, the phrase *"born in"* might not always be present as a consecutive phrase, consider *"SUBJECT_PERSON was born as the second son of PERSON_X in OBJECT_PLACE"*; furthermore, its presence does not guarantee that the "place of birth" relation holds between entities, consider *"SUBJECT_PERSON's mother was then pregnant with her brother, who was later born in OBJECT_PLACE "*.

These challenging examples require more robust methods to correctly discern the relation between the entities in the sentence, and fortunately the structured nature of natural language is on our side in this case. More specifically, let's take a look at the syntactic structure of these examples, as represented in the *dependency parse* format. From Figure 2.3, we can readily see that despite their differences in the surface form of the sentences, the underlying syntactic relation between the subject and object entities are almost identical in the first three examples where the relation between the entities is "place of birth", whereas that of the last example is very different.

Relation extraction models have historically made use of this syntactic information as dependency

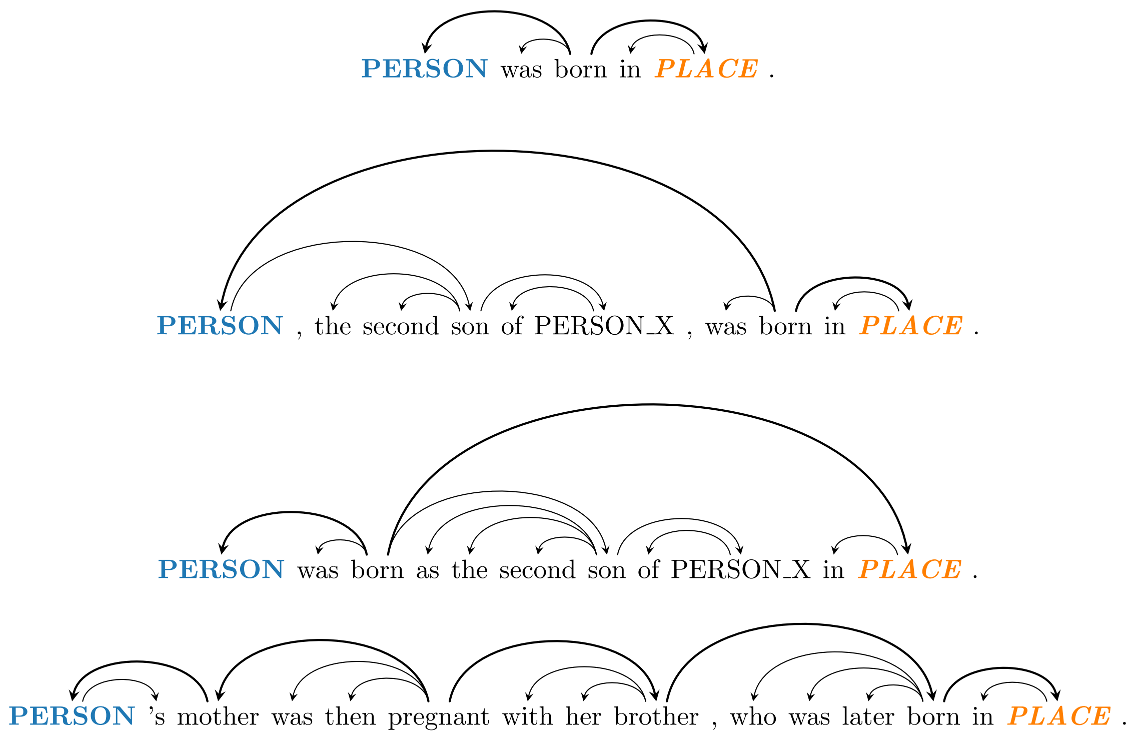

PERSON was born in PLACE .
PERSON , the second son of PERSON_X , was born in PLACE .
PERSON was born as the second son of PERSON_X in PLACE .
PERSON 's mother was then pregnant with her brother , who was later born in PLACE .

Figure 2.3: The dependency parse of example sentences that are challenging for relation extraction models that only focus on the surface form of the sentence, in the Universal Dependencies format (Nivre et al., 2020). The shortest path between the **subject** and ***object*** entities is highlighted with bold edges, and the dependency relations with punctuation marks are omitted for clarity.

patterns, most of which are extracted along the shortest path between the entities in the tree. More recently, the NLP community has also adopted neural networks to model these syntactic structures in various ways. Despite their apparent appeal of robustness to longer sentences where the subject and object entities are farther apart, these syntax-based models have not been able to outperform their sequence-based counterparts when large amounts of data are available. In this chapter, I will present how we combined these syntactic structures with graph neural networks, and explain why previous syntax-based methods might suffer from either too much noise or missing critical context by introducing our linguistically-motivated tree-pruning technique. When this model and our pruning technique are combined, we achieved state-of-the-art performance on, TACRED, one of the largest relation extraction datasets available at the time, and outperformed previous syntax-based methods on SemEval 2010 Task 8, a smaller benchmark. Specifically, on both datasets, our model not only outperforms existing dependency-based neural models by a significant margin when combined with the new pruning technique, but also achieves a 10–100x speedup over existing tree-based models. On TACRED, besides outperforming strong baselines, our model also exhibits complementary strengths to sequence models. We show that combining these two model types through simple prediction interpolation further improves the state of the art.

Our main contributions are: (i) we propose a neural model for relation extraction based on graph convolutional networks, which allows it to efficiently pool information over arbitrary dependency structures; (ii) we present a new path-centric pruning technique to help dependency-based models maximally remove irrelevant information without damaging crucial content to improve their robustness; (iii) we present detailed analysis on the model and the pruning technique, and show that dependency-based models have complementary strengths with sequence models.

2.1 Relation Extraction with Syntax Trees

In relation extraction, one of the most commonly used forms of syntactic structure is dependency trees, as I have shown earlier in this chapter. Models making use of dependency parses of the input sentences, or *dependency-based models*, have proven to be very effective in relation extraction. These models are effective, because they capture long-range syntactic relations that are obscure from the surface form alone into short paths in the syntactic structure (*e.g.*, when long clauses or complex scoping are present). Traditional feature-based models are able to represent dependency information by featurizing dependency trees as overlapping paths along the trees (Kambhatla, 2004). However, these models face the challenge of sparse feature spaces and are brittle to lexical variations. More recent neural models address this problem with distributed representations built from their computation graphs formed along parse trees. One common approach to leverage dependency information is to perform bottom-up or top-down computation along the parse tree or the subtree below the lowest common ancestor (LCA) of the entities (Miwa and Bansal, 2016). Another popular approach,

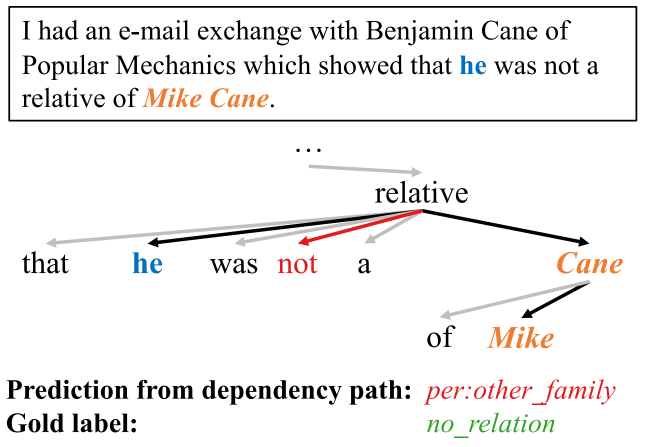

I had an e-mail exchange with Benjamin Cane of Popular Mechanics which showed that he was not a relative of Mike Cane.
…
relative
that
he
was
not
a
Cane
of
Mike
Prediction from dependency path: per:other_family
Gold label: no_relation

Figure 2.4: An example modified from the TAC KBP challenge corpus. A subtree of the original UD dependency tree between the subject ("he") and object ("Mike Cane") is also shown, where the shortest dependency path between the entities is highlighted in bold. Note that negation ("not") is off the dependency path.

inspired by Bunescu and Mooney (2005), is to reduce the parse tree to the *shortest dependency path* between the entities (Xu et al., 2015a,b). Besides these neural network approaches, kernel methods have also been applied to capture local similarities in dependency patterns (Zelenko et al., 2003; Culotta and Sorensen, 2004).

However, these models suffer from several drawbacks. Neural models operating directly on parse trees are usually difficult to parallelize and thus computationally inefficient, because aligning trees for efficient batch training is usually non-trivial, especially for trees of indeterminate branching factors like dependency trees. Models based on the shortest dependency path between the subject and object are computationally more efficient, but this simplifying assumption has major limitations as well. Figure 2.4 shows a real-world example where crucial information (i.e., negation) would be excluded when the model is restricted to only considering the dependency path.

In this chapter, we propose a novel extension of the graph convolutional network (Kipf and Welling, 2017; Marcheggiani and Titov, 2017) that is tailored for relation extraction. Our model encodes the dependency structure over the input sentence with efficient graph convolution operations, then extracts entity-centric representations to make robust relation predictions. We also apply a novel *path-centric pruning* technique to remove irrelevant information from the tree while maximally keeping relevant content, which further improves the performance of several dependency-based models including ours.

2.2 Models

In this section, we first describe graph convolutional networks (GCNs) over dependency tree structures, and then we introduce an architecture that uses GCNs at its core for relation extraction.

2.2.1 Graph Convolutional Networks over Dependency Trees

The graph convolutional network (Kipf and Welling, 2017) is an adaptation of the convolutional neural network (LeCun et al., 1998) for encoding graphs. Given a graph with n nodes, we can represent the graph structure with an $n \times n$ adjacency matrix $\mathbf{A}$ where $A_{ij} = 1$ if there is an edge going from node i to node j. In an L-layer GCN, if we denote by $h_i^{(l-1)}$ the input vector and $h_i^{(l)}$ the output vector of node i at the l-th layer, a graph convolution operation can be written as

$$h_i^{(l)} = \sigma\big(\sum_{j=1}^{n} A_{ij} W^{(l)} h_j^{(l-1)} + b^{(l)}\big), \tag{2.1}$$

where $W^{(l)}$ is a linear transformation, $b^{(l)}$ a bias term, and σ a nonlinear function (e.g., ReLU). Intuitively, during each graph convolution, each node gathers and summarizes information from its neighboring nodes in the graph.

We adapt the graph convolution operation to model dependency trees by converting each tree into its corresponding adjacency matrix $\mathbf{A}$, where $A_{ij} = 1$ if there is a dependency edge between tokens i and j. However, naively applying the graph convolution operation in Equation (2.1) could lead to node representations with drastically different magnitudes, since the degree of a token varies a lot. This could bias our sentence representation towards favoring high-degree nodes regardless of the information carried in the node (see details in Section 2.2.2). Furthermore, the information in $h_i^{(l-1)}$ is never carried over to $h_i^{(l)}$, since nodes never connect to themselves in a dependency tree.

We resolve these issues by normalizing the activations in the graph convolution before feeding it through the nonlinearity, and adding self-loops to each node in the graph:

$$h_i^{(l)} =\sigma\big(\sum_{j=1}^{n} \tilde{A}_{ij} W^{(l)} h_j^{(l-1)} / d_i + b^{(l)}\big), \tag{2.2}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ with $\mathbf{I}$ being the $n \times n$ identity matrix, and $d_i = \sum_{j=1}^{n} \tilde{A}_{ij}$ is the degree of token i in the resulting graph. Figure 2.5 shows an example for what these adjacency matrices look like for the sentence *"He was not a relative of Mike Cane"*, a simplified version of the sentence in Figure 2.4.

Stacking this operation over L layers gives us a deep GCN network, where we set $h_1^{(0)}, \ldots, h_n^{(0)}$ to be input word vectors, and use $h_1^{(L)}, \ldots, h_n^{(L)}$ as output word representations. All operations in

$$
\begin{array}{c|cccccccc|}
 & \text{He} & \text{was} & \text{not} & \text{a} & \text{relative} & \text{of} & \text{Mike} & \text{Cane} \\
\text{He} & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\text{was} & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
\text{not} & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
\text{a} & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
\text{relative} & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
\text{of} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
\text{Mike} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\text{Cane} & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
$$

Figure 2.5: An example of the adjacency matrices used in GCN computation for a simplified version of the sentence in Figure 2.4. Off-diagonal elements shaded in blue are those that appear in the original adjacency matrix $\mathbf{A}$, to which the diagonal elements shaded in orange are added to arrive at the final matrix $\tilde{\mathbf{A}}$ used in our GCN models.

this network can be efficiently implemented with matrix multiplications, making it ideal for batching computation over examples and running on GPUs. Moreover, the propagation of information between tokens occurs in parallel, and the runtime does not depend on the depth of the dependency tree.

Note that the GCN model presented above uses the same parameters for all edges in the dependency graph. We also experimented with: (1) using different transformation matrices W for top-down, bottom-up, and self-loop edges; and (2) adding dependency relation-specific parameters for edge-wise gating, similar to (Marcheggiani and Titov, 2017). We found that modeling directions does not lead to improvement, and adding edge-wise gating further hurts performance. We hypothesize that this is because the presented GCN model is usually already able to capture dependency edge patterns that are informative for classifying relations, and modeling edge directions and types does not offer additional discriminative power to the network before it leads to overfitting. For example, the relations entailed by "A's son, B" and "B's son, A" can be readily distinguished with "'s" attached to different entities, even when edge directionality is not considered. Therefore, in all of our experiments, we treat the dependency graph as undirected, i.e. $\forall i, j, A_{ij} = A_{ji}$.

2.2.2 Encoding Relations with GCN

We now formally define the task of relation extraction. Let $\mathcal{X} = [x_1, ..., x_n]$ denote a sentence, where x_i is the i^{th} token. A subject entity and an object entity are identified and correspond to two spans in the sentence: $\mathcal{X}_s = [x_{s_1}, \ldots, x_{s_2}]$ and $\mathcal{X}_o = [x_{o_1}, \ldots, x_{o_2}]$. Given $\mathcal{X}$, $\mathcal{X}_s$, and $\mathcal{X}_o$, the goal of relation extraction is to predict a relation $r \in \mathcal{R}$ (a predefined relation set) that holds between the entities or "no relation" otherwise.

After applying an L-layer GCN over word vectors, we obtain hidden representations of each

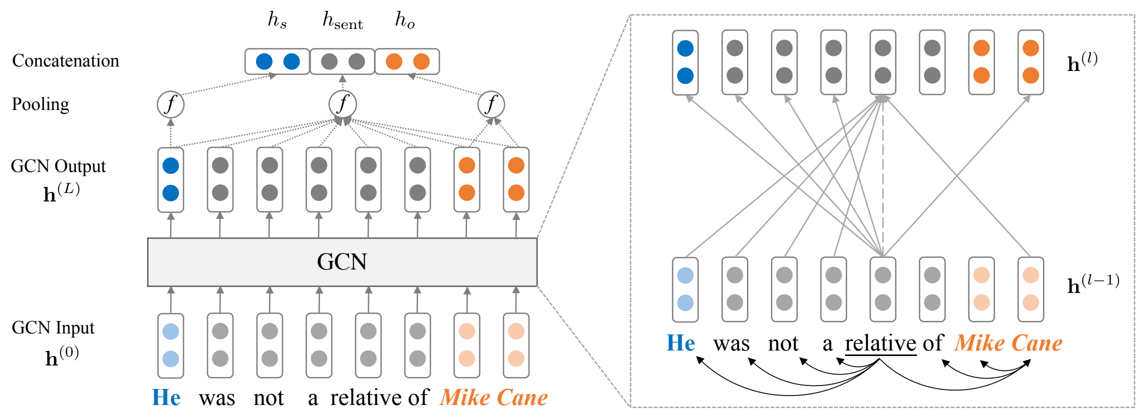


Figure 2.6: Relation extraction with a graph convolutional network. The left side shows the overall architecture, while on the right side, we only show the detailed graph convolution computation for the word "relative" for clarity. A full unlabeled dependency parse of the sentence is also provided for reference.

token that are directly influenced by its neighbors no more than L edges apart in the dependency tree. To make use of these word representations for relation extraction, we first obtain a sentence representation as follows (see also Figure 2.6 left):

$$h_{\text{sent}} = f\big(\mathbf{h}^{(L)}\big) = f\big(\text{GCN}(\mathbf{h}^{(0)})\big), \tag{2.3}$$

where $\mathbf{h}^{(l)}$ denotes the collective hidden representations at layer l of the GCN, and $f : \mathbb{R}^{d\times n} \to \mathbb{R}^d$ is a max pooling function that maps from n output vectors to the sentence vector.

We also observe that information close to entity tokens in the dependency tree is often central to relation classification. Therefore, we also obtain a subject representation h_s from $\mathbf{h}^{(L)}$ as follows

$$h_s = f\big(\mathbf{h}^{(L)}_{s_1:s_2}\big), \tag{2.4}$$

as well as an object representation h_o similarly.

Inspired by recent work on relational learning between entities (Santoro et al., 2017; Lee et al., 2017), we obtain the final representation used for classification by concatenating the sentence and the entity representations, and feeding them through a feed-forward neural network (FFNN):

$$h_{\text{final}} = \text{FFNN}\big([h_{\text{sent}}; h_s; h_o]\big). \tag{2.5}$$

This h_{final} representation is then fed into a linear layer followed by a softmax operation to obtain a probability distribution over relations.

2.2.3 Contextualized GCN

The network architecture introduced so far learns effective representations for relation extraction, but it also leaves a few issues inadequately addressed. First, the input word vectors do not contain contextual information about word order or disambiguation. Second, the GCN highly depends on a correct parse tree to extract crucial information from the sentence (especially when pruning is performed), while existing parsing algorithms produce imperfect trees in many cases.

To resolve these issues, we further apply a Contextualized GCN (C-GCN) model, where the input word vectors are first fed into a bi-directional long short-term memory (LSTM) network to generate contextualized representations, which are then used as $\mathbf{h}^{(0)}$ in the original model. This BiLSTM contextualization layer is trained jointly with the rest of the network. We show empirically in Section 2.5 that this augmentation substantially improves the performance over the original model.

We note that this relation extraction model is conceptually similar to graph kernel-based models (Zelenko et al., 2003; Culotta and Sorensen, 2004), in that it aims to utilize local dependency tree patterns to inform relation classification. Our model also incorporates crucial off-path information, which greatly improves its robustness compared to shortest dependency path-based approaches. Compared to tree-structured neural models (e.g., Tree-LSTM (Tai et al., 2015)), it not only is able to capture more global information through the use of pooling functions, but also achieves substantial speedup by not requiring recursive operations that are difficult to parallelize. For example, we observe that on a Titan Xp GPU, training a Tree-LSTM model over a minibatch of 50 examples takes 6.54 seconds on average, while training the original GCN model takes only 0.07 seconds, and the C-GCN model 0.08 seconds.

2.3 Incorporating Off-path Information with Path-centric Pruning

Dependency trees provide rich structures that one can exploit in relation extraction, but most of the information pertinent to relations is usually contained within the subtree rooted at the lowest common ancestor (LCA) of the two entities. Previous studies (Xu et al., 2015b; Miwa and Bansal, 2016) have shown that removing tokens outside this scope helps relation extraction by eliminating irrelevant information from the sentence. It is therefore desirable to combine our GCN models with tree pruning strategies to further improve performance. However, pruning too aggressively (*e.g.*, keeping only the dependency path) could lead to loss of crucial information and conversely hurt robustness. For instance, the negation in Figure 2.4 is neglected when a model is restricted to only looking at the dependency path between the entities. Similarly, in the sentence "*Her family confirmed that she was diagnosed with cancer last year, and succumbed this June*", the dependency path *she*←*diagnosed*→*cancer* is not sufficient to establish that *cancer* is the cause of death for the

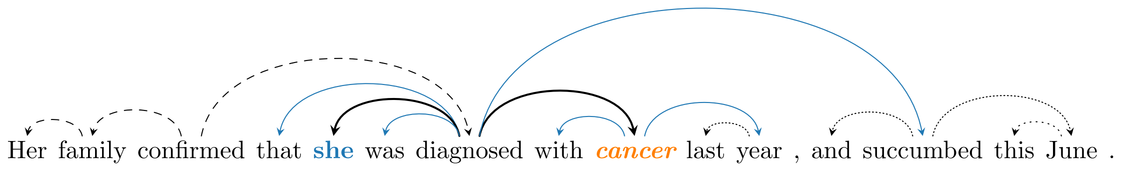

Her family confirmed that **she** was diagnosed with ***cancer*** last year , and succumbed this June .

Figure 2.7: An example of conjunctive structure in a sentence, where the shortest dependency path (highlighted in bold) between the **subject** and ***object*** entities fails to capture crucial information to discern the relation between them ("succumbed", in this case). When path-centric pruning is applied with $K = 1$, the blue dependency edges in the figure are kept along with the shortest path, which contains the crucial information ("succumbed") but not too much noise. $K = 2$ further captures all densely dotted dependency edges (*e.g.*, *succumbed*→*June*), and when $K > 2$, the entire subtree rooted at the LCA of the two entities (*i.e.*, the entire complement clause) is retained. In both cases, the pruned tree tends to capture additional information that is not immediately useful for discerning the relation between the subject and object entities (date of death, in this case). Finally, dashed dependency edges are part of the full tree, when it is used in computation. Dependency relations with punctuation marks are omitted for clarity.

subject unless the conjunction dependency to *succumbed* is also present (see Figure 2.7).

Motivated by these observations, we propose *path-centric pruning*, a novel technique to incorporate information off the dependency path. This is achieved by including tokens that are up to distance K away from the dependency path in the LCA subtree. $K = 0$, corresponds to pruning the tree down to the path, $K = 1$ keeps all nodes that are directly attached to the path, and $K = \infty$ retains the entire LCA subtree (See Figure 2.7 for a detailed example). We combine this pruning strategy with our GCN model, by directly feeding the pruned trees into the graph convolutional layers.[4] We show that pruning with $K = 1$ achieves the best balance between including relevant information (e.g., negation and conjunction) and keeping irrelevant content out of the resulting pruned tree as much as possible.

2.4 Related Work

At the core of fully-supervised and distantly-supervised relation extraction approaches are statistical classifiers, many of which find syntactic information beneficial. For example, Mintz et al. (2009) explored adding syntactic features to a statistical classifier and found them to be useful when sentences are long. Various kernel-based approaches also leverage syntactic information to measure similarity between training and test examples to predict the relation, finding that tree-based kernels (Zelenko et al., 2003; Culotta and Sorensen, 2004) and dependency path-based kernels (Bunescu and Mooney, 2005) are effective for this task.

[4]For our C-GCN model, the LSTM layer still operates on the full sentence regardless of the pruning.

Recent studies have found neural models effective in relation extraction. Zeng et al. (2014) first applied a one-dimensional convolutional neural network (CNN) with manual features to encode relations. Vu et al. (2016) showed that combining a CNN with a recurrent neural network (RNN) through a voting scheme can further improve performance. Zhou et al. (2016) and Wang et al. (2016) proposed to use attention mechanisms over RNN and CNN architectures for this task.

Apart from neural models over word sequences, incorporating dependency trees into neural models has also been shown to improve relation extraction performance by capturing long-distance relations. Xu et al. (2015b) generalized the idea of dependency path kernels by applying a LSTM network over the shortest dependency path between entities. Liu et al. (2015) first applied a recursive network over the subtrees rooted at the words on the dependency path and then applied a CNN over the path. Miwa and Bansal (2016) applied a Tree-LSTM (Tai et al., 2015), a generalized form of LSTM over dependency trees, in a joint entity and relation extraction setting. They found it to be most effective when applied to the subtree rooted at the LCA of the two entities.

More recently, Adel et al. (2016) and Zhang et al. (2017) have shown that relatively simple neural models (CNN and augmented LSTM, respectively) can achieve comparable or superior performance to dependency-based models when trained on larger datasets. In this chapter, we study dependency-based models in depth and show that with a properly designed architecture, they can outperform and have complementary advantages to sequence models, even in a large-scale setting.

Finally, we note that a technique similar to path-centric pruning has been applied to reduce the space of possible arguments in semantic role labeling (He et al., 2018). The authors showed pruning words too far away from the path between the predicate and the root to be beneficial, but reported the best pruning distance to be 10, which almost always retains the entire tree. Our method differs in that it is applied to the shortest dependency path between entities, and we show that in our technique the best pruning distance is 1 for several dependency-based relation extraction models.

2.5 Experiments

2.5.1 Baseline Models

We compare our models with several competitive dependency-based and neural sequence models.

Dependency-based models. In our main experiments we compare with three types of dependency-based models. (1) A logistic regression (LR) classifier which combines dependency-based features with other lexical features. (2) Shortest Dependency Path LSTM (SDP-LSTM) (Xu et al., 2015b), which applies a neural sequence model on the shortest path between the subject and object entities in the dependency tree. (3) Tree-LSTM (Tai et al., 2015), which is a recursive model that generalizes the LSTM to arbitrary tree structures. We investigate the child-sum variant of Tree-LSTM, and apply it to the dependency tree (or part of it). In practice, we find that modifying this model by concatenating dependency label embeddings to the input of forget gates improves its performance on relation extraction, and therefore use this variant in our experiments. Earlier, our group compared (1) and (2) with sequence models (Zhang et al., 2017), and we report these results; for (3) we report results with our own implementation.

Neural sequence model. Our group presented a competitive sequence model that employs a position-aware attention mechanism over LSTM outputs (PA-LSTM), and showed that it outperforms several CNN and dependency-based models by a substantial margin (Zhang et al., 2017). We compare with this strong baseline, and use its open implementation in further analysis.[5]

2.5.2 Experimental Setup

We conduct experiments on two relation extraction datasets: (1) **TACRED**: Introduced in (Zhang et al., 2017), TACRED contains over 106k mention pairs drawn from the yearly TAC KBP[6] challenge. It represents 41 relation types and a special *no_relation* class when the mention pair does not have a relation between them within these categories. Mentions in TACRED are typed, with subjects categorized into person and organization, and objects into 16 fine-grained types (e.g., date and location). We report micro-averaged F_1 scores on this dataset as is conventional. (2) **SemEval 2010 Task 8**: The SemEval dataset is widely used in recent work, but is significantly smaller with 8,000 examples for training and 2,717 for testing. It contains 19 relation classes over untyped mention pairs: 9 directed relations and a special *Other* class. On SemEval, we follow the convention and report the official macro-averaged F_1 scores.

For fair comparisons on the TACRED dataset, we follow the evaluation protocol used in (Zhang et al., 2017) by selecting the model with the median dev F_1 from 5 independent runs and reporting its test F_1. We also use the same "entity mask" strategy where we replace each subject (and object similarly) entity with a special *SUBJ-<NER>* token. For instance, the sentence in Figure 2.6 (*"[He]$_{SUBJ}$ was not a relative of [Mike Cane]$_{OBJ}$"*) is converted to *"SUBJ-<PER> was not a relative of OBJ-<PER> OBJ-<PER>"* before fed into our model. For all models, we also adopt the "multi-channel" strategy by concatenating the input word embeddings with POS and NER embeddings.

Traditionally, evaluation on SemEval is conducted without entity mentions masked. However, as we will discuss in Section 2.7.4, this method encourages models to overfit to these mentions and fails to test their actual ability to generalize. We therefore report results with two evaluation protocols: (1) *with-mention*, where mentions are kept for comparison with previous work; and (2) *mask-mention*, where they are masked to test the generalization of our model in a more realistic setting.

[5] https://github.com/yuhaozhang/tacred-relation
[6] https://tac.nist.gov/2017/KBP/index.html

2.5.3 Hyperparameters

We use the following hyperparameters to train the GCN model on the two datasets.

TACRED We set LSTM hidden size to 200 in all neural models. We also use hidden size 200 for the output feedforward layers in the GCN model. We use 2 GCN layers and 2 feedforward (FFNN) layers in our experiments. We employ the ReLU function for all nonlinearities in the GCN layers and the standard max pooling operations in all pooling layers. For the Tree-LSTM model, we find a 2-layer architecture works substantially better than the vanilla 1-layer model, and use it in all our experiments. For both the Tree-LSTM and our models, we apply path-centric pruning with $K = 1$, as we find that this generates best results for all models (also see Figure 2.8). We use the pre-trained 300-dimensional GloVe vectors (Pennington et al., 2014) to initialize word embeddings, and we use embedding size of 30 for all other embeddings (i.e., POS, NER). We use the dependency parse trees, POS and NER sequences as included in the original release of the dataset, which was generated with Stanford CoreNLP (Manning et al., 2014). For regularization we apply dropout with $p = 0.5$ to all LSTM layers and all but the last GCN layers.

SemEval We use LSTM hidden size of 100 and use 1 GCN layer for the SemEval dataset. We preprocess the dataset with Stanford CoreNLP to generate the dependency parse trees, POS and NER annotations. All other hyperparameters are set to be the same.

For both datasets, we work with the Universal Dependencies v1 formalism (Nivre et al., 2016).

2.5.4 Training

For training we use Stochastic Gradient Descent with an initial learning rate of 1.0. We use a cutoff of 5 for gradient clipping. For GCN models, we train every model for 100 epochs on the TACRED dataset, and from epoch 5 we start to anneal the learning rate by a factor of 0.9 every time the F_1 score on the dev set does not increase after an epoch. For Tree-LSTM models we find 30 total epochs to be enough. Due to the small size of the SemEval dataset, we train all models for 150 epochs, and use an initial learning rate of 0.5 with a decay rate of 0.95.

In our experiments we found that the output vector h_{sent} tends to have large magnitude, and therefore adding the following regularization term to the cross entropy loss of each example improves the results:

$$\ell_{\text{reg}} = \beta \cdot \|h_{\text{sent}}\|^2. \tag{2.6}$$

Here, ℓ_{reg} functions as an l_2 regularization on the learned sentence representations. β controls the regularization strength and we set $\beta = 0.003$. We empirically found this to be more effective than applying l_2 regularization on the convolutional weights.

System	P	R	F_1
LR[†] (Zhang et al., 2017)	**<u>73.5</u>**	49.9	59.4
SDP-LSTM[†] (Xu et al., 2015b)	66.3	52.7	58.7
Tree-LSTM[‡] (Tai et al., 2015)	66.0	59.2	62.4
PA-LSTM[†] (Zhang et al., 2017)	65.7	<u>64.5</u>	65.1
GCN	69.8	59.0	64.0
C-GCN	69.9	63.3	<u>66.4</u>[*]
GCN + PA-LSTM	71.7	63.0	67.1[*]
C-GCN + PA-LSTM	71.3	**65.4**	**68.2**[*]

Table 2.1: Relation extraction results on TACRED. Underscore marks highest number among single models; bold marks highest among all. † marks results reported in (Zhang et al., 2017); ‡ marks results produced with our implementation. $*$ marks statistically significant improvements over PA-LSTM with $p < .01$ under a bootstrap test.

2.6 Results

2.6.1 Results on the TACRED Dataset

We present our main results on the TACRED test set in Table 2.1. We observe that our GCN model outperforms all dependency-based models by at least 1.6 F_1. By using contextualized word representations, the C-GCN model further outperforms the strong PA-LSTM model by 1.3 F_1, and achieves a new state of the art. In addition, we find our model improves upon other dependency-based models in both precision and recall. Comparing the C-GCN model with the GCN model, we find that the gain mainly comes from improved recall. We hypothesize that this is because the C-GCN is more robust to parse errors by capturing local word patterns (see also Section 2.7.2).

As we will show in Section 2.7.2, we find that our GCN models have complementary strengths when compared to the PA-LSTM. To leverage this result, we experiment with a simple interpolation strategy to combine these models. Given the output probabilities $P_G(r|x)$ from a GCN model and $P_S(r|x)$ from the sequence model for any relation r, we calculate the interpolated probability as

$$P(r|x) = \alpha \cdot P_G(r|x) + (1 - \alpha) \cdot P_S(r|x)$$

where $\alpha \in [0, 1]$ is chosen on the dev set and set to 0.6. This simple interpolation between a GCN and a PA-LSTM achieves an F_1 score of 67.1, outperforming each model alone by at least 2.0 F_1. An interpolation between a C-GCN and a PA-LSTM further improves the result to 68.2.

System	*with-mentions*	*mask-mentions*
SVM† (Rink and Harabagiu, 2010)	82.2	–
SDP-LSTM† (Xu et al., 2015b)	83.7	–
SPTree† (Miwa and Bansal, 2016)	84.4	–
PA-LSTM‡ (Zhang et al., 2017)	82.7	75.3
Our Model (C-GCN)	**84.8***	**76.5***

Table 2.2: Relation extraction results on SemEval. † marks results reported in the original papers; ‡ marks results produced by using the open implementation. The last two columns show results from *with-mention* evaluation and *mask-mention* evaluation, respectively. $*$ marks statistically significant improvements over PA-LSTM with $p < .05$ under a bootstrap test.

2.6.2 Results on the SemEval Dataset

To study the generalizability of our proposed model, we also trained and evaluated our best C-GCN model on the SemEval test set (Table 2.2). We find that under the conventional *with-entity* evaluation, our C-GCN model outperforms all existing dependency-based neural models on this separate dataset. Notably, by properly incorporating off-path information, our model outperforms the previous shortest dependency path-based model (SDP-LSTM). Under the *mask-entity* evaluation, our C-GCN model also outperforms PA-LSTM by a substantial margin, suggesting its generalizability even when entities are not seen.

2.6.3 Effect of Path-centric Pruning

To show the effectiveness of path-centric pruning, we compare the two GCN models and the Tree-LSTM when the pruning distance K is varied. We experimented with $K \in \{0, 1, 2, \infty\}$ on the TACRED dev set, and also include results when the full tree is used. As shown in Figure 2.8, the performance of all three models peaks when $K = 1$, outperforming their respective dependency path-based counterpart ($K = 0$). This confirms our hypothesis in Section 2.3 that incorporating off-path information is crucial to relation extraction. Miwa and Bansal (2016) reported that a Tree-LSTM achieves similar performance when the dependency path and the LCA subtree are used respectively. Our experiments confirm this, and further show that the result can be improved by path-centric pruning with $K = 1$.

We find that all three models are less effective when the entire dependency tree is present, indicating that including extra information hurts performance. Finally, we note that contextualizing the GCN makes it less sensitive to changes in the tree structures provided, presumably because the model can use word sequence information in the LSTM layer to recover any off-path information that it needs for correct relation extraction.

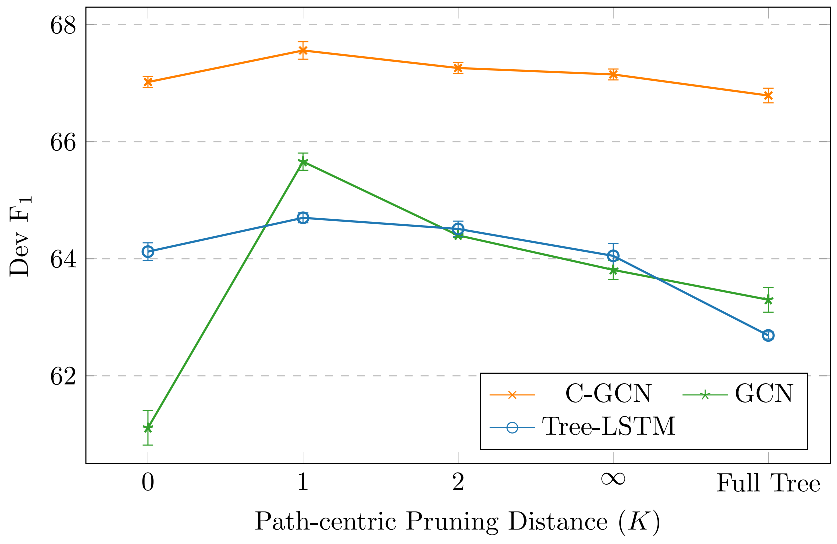


Figure 2.8: Performance of dependency-based models under different pruning strategies. For each model we show the F_1 score on the TACRED dev set averaged over 5 runs, and error bars indicate standard deviation of the mean estimate. $K = \infty$ is equivalent to using the subtree rooted at the LCA.

2.7 Analysis & Discussion

2.7.1 Ablation Study

To study the contribution of each component in the C-GCN model, we ran an ablation study on the TACRED dev set (Table 2.3). We find that: (1) The entity representations and feedforward layers contribute 1.0 F_1. (2) When we remove the dependency structure (i.e., setting $\tilde{\mathbf{A}}$ to $\mathbf{I}$), the score drops by 3.2 F_1. (3) F_1 drops by 10.3 when we remove the feedforward layers, the LSTM component and the dependency structure altogether. (4) Removing the pruning (i.e., using full trees as input) further hurts the result by another 9.7 F_1.

2.7.2 Complementary Strengths of GCNs and PA-LSTMs

To understand what the GCN models are capturing and how they differ from a sequence model such as the PA-LSTM, we compared their performance over examples in the TACRED dev set. Specifically, for each model, we trained it for 5 independent runs with different seeds, and for each example we evaluated the model's accuracy over these 5 runs. For instance, if a model correctly classifies an example for 3 out of 5 times, it achieves an accuracy of 60% on this example. We observe that on 847 (3.7%) dev examples, our C-GCN model achieves an accuracy at least 60% higher than that of the PA-LSTM, while on 629 (2.8%) examples the PA-LSTM achieves 60% higher. This complementary performance explains the gain we see in Table 2.1 when the two models

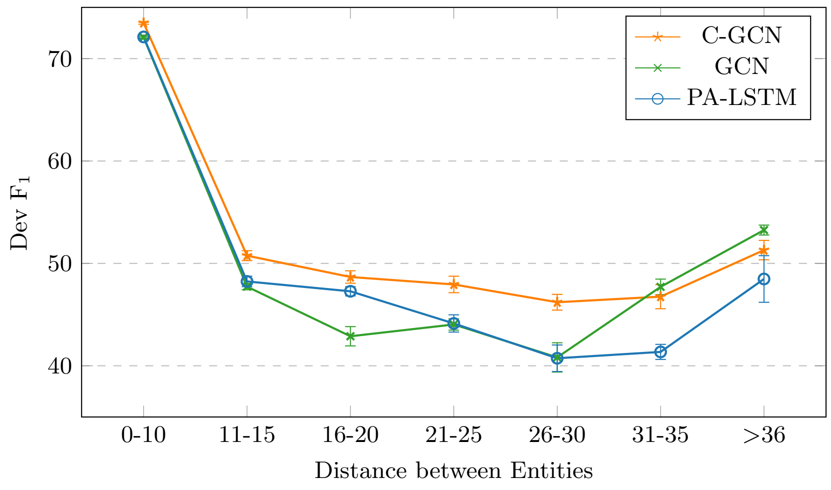


Figure 2.9: Dev set performance with regard to distance between the entities in the sentence for C-GCN, GCN and PA-LSTM. Error bars indicate standard deviation of the mean estimate over 5 runs.

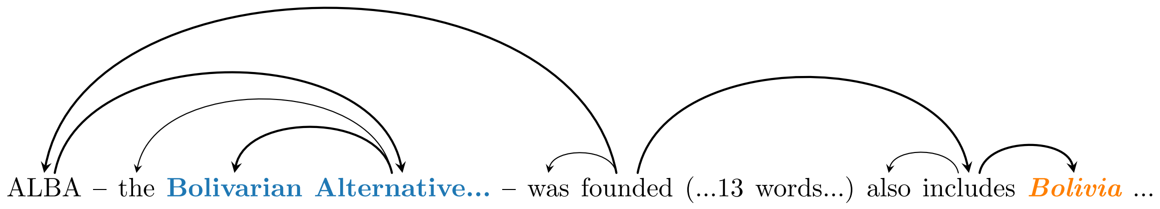
Model	Dev F_1
Best C-GCN	67.4
– h_s, h_o, and Feedforward (FF)	66.4
– LSTM Layer	65.5
– Dependency tree structure	64.2
– FF, LSTM, and Tree	57.1
– FF, LSTM, Tree, and Pruning	47.4

Table 2.3: An ablation study of the best C-GCN model. Score s are median of 5 models.

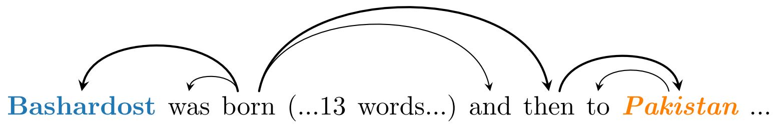
are combined.

We further show that this difference is due to each model's competitive advantage (Figure 2.9): dependency-based models are better at handling sentences with entities farther apart, while sequence models can better leverage local word patterns regardless of parsing quality (see also Figure 2.10).

We further compared the performance of both GCN models with the PA-LSTM on the TACRED dev set. To minimize randomness that is not inherent to these models, we accumulate statistics over 5 independent runs of each model, and report them in Figure 2.11. As is shown in the figure, both GCN models capture very different examples from the PA-LSTM model. In the entire dev set of 22,631 examples, 1,450 had at least 3 more GCN models predicting the label correctly compared to the PA-LSTM, and 1,550 saw an improvement from using the PA-LSTM. The C-GCN, on the other hand, outperformed the PA-LSTM by at least 3 models on a total of 847 examples, and lost by a margin of at least 3 on another 629 examples, as reported in the main text. This smaller difference is also reflected in the diminished gain from ensembling with the PA-LSTM shown in Table 2.1. We


ALBA – the **Bolivarian Alternative...** – was founded (...13 words...) also includes ***Bolivia*** ...

Full sentence: ALBA – the **Bolivarian Alternative for the Americas** – was founded by Venezuelan President Hugo Chavez and Cuban leader Fidel Castro in 2004 and also includes ***Bolivia***, Nicaragua and the Caribbean island of Dominica.


Bashardost was born (...13 words...) and then to ***Pakistan*** ...

Full sentence: **Bashardost** was born in 1965 in the southern Ghanzi province and his family migrated to Iran and then to ***Pakistan*** after successive coup and factional fighting in Afghanistan.

Figure 2.10: Dev set examples where either the C-GCN (upper) or the PA-LSTM (lower) predicted correctly in five independent runs. For each example, the predicted and pruned dependency tree corresponding to $K = 1$ in path-centric pruning is shown, and the shortest dependency path is thickened. For clarity, we also show the subject in **blue boldface** and the object in ***orange italics***. We omit edges to punctuation and some words in the sentence for clarity of illustration. The first example shows that the C-GCN is effective at leveraging long-range dependencies while reducing noise with the help of pruning (while the PA-LSTM predicts *no_relation* twice, *org:alternate_names* twice, and *org:parents* once in this case). The second example shows that the PA-LSTM is better at leveraging the proximity of the word "migrated" regardless of attachment errors in the parse (while the C-GCN is misled to predict *per:country_of_birth* three times, and *no_relation* twice).

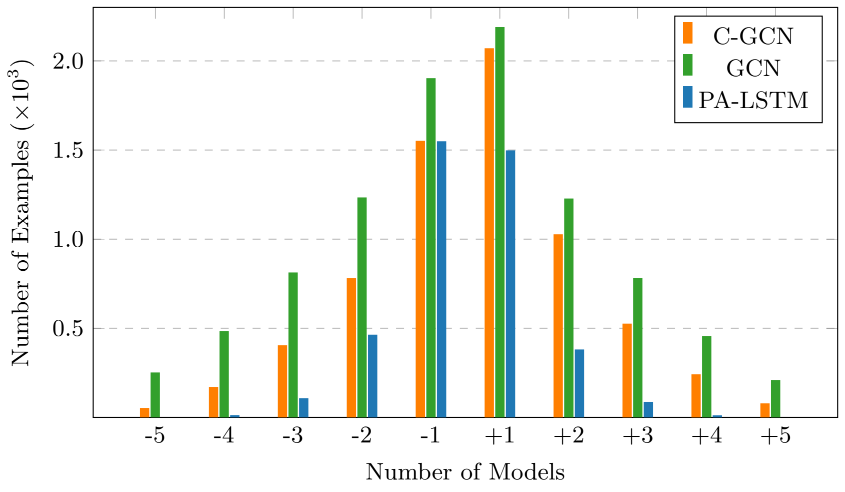


Figure 2.11: Aggregated 5-run difference compared to PA-LSTM on the TACRED dev set. For each example, if X out of 5 GCN models predicted its label correctly and Y PA-LSTM models did, it is aggregated in the bar labeled $X - Y$. "0" is omitted due to redundancy.

hypothesize that the diminishing difference results from the LSTM contextualization layer, which incorporates more information readily available at the surface form, rendering the model's behavior more similar to a sequence model.

For reference, we also include in Figure 2.11 the comparison of another 5 different runs (with different seeds) of the PA-LSTM to the original 5 runs of the PA-LSTM. This is to confirm that the difference shown in the figure between the model classes is indeed due a to model difference, rather than an effect of different random seeds. More specifically, the two groups of PA-LSTM only see 99 and 121 examples exceeding the 3-model margin on either side over the 5 runs, much lower than the numbers reported above for the GCN models.

2.7.3 Understanding Model Behavior

To gain more insights into the C-GCN model's behavior, we visualized the partial dependency tree it is processing and how much each token's final representation contributed to h_{sent} (Figure 2.12). We find that the model often focuses on the dependency path, but sometimes also incorporates off-path information to help reinforce its prediction. The model also learns to ignore determiners (e.g., "the") as they rarely affect relation prediction.

To further understand what dependency edges contribute most to the classification of different relations, we scored each dependency edge by summing up the number of dimensions each of its connected nodes contributed to h_{sent}. We present the top scoring edges in Table 2.4. As can be seen in the table, most of these edges are associated with indicative nouns or verbs of each relation. We do notice the effect of dataset bias as well: the name "Buffett" is too often associated with contexts

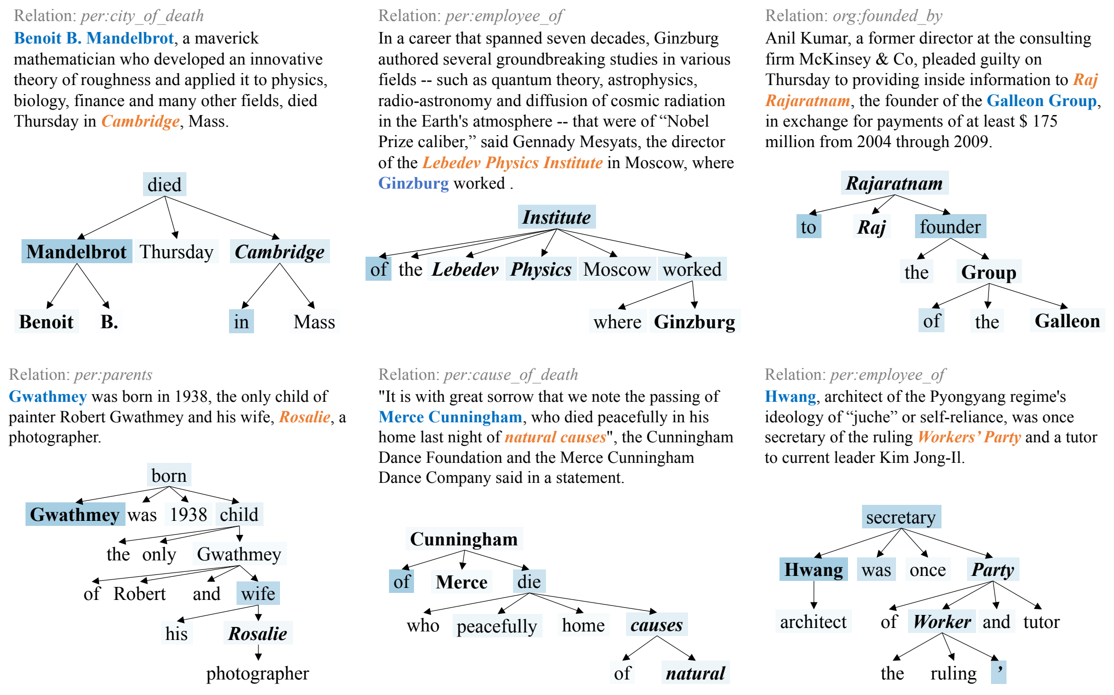

Relation: per:city_of_death
Benoit B. Mandelbrot, a maverick mathematician who developed an innovative theory of roughness and applied it to physics, biology, finance and many other fields, died Thursday in Cambridge, Mass.
died
Mandelbrot
Thursday
Cambridge
Benoit
B.
in
Mass
Relation: per:employee_of
In a career that spanned seven decades, Ginzburg authored several groundbreaking studies in various fields -- such as quantum theory, astrophysics, radio-astronomy and diffusion of cosmic radiation in the Earth's atmosphere -- that were of "Nobel Prize caliber," said Gennady Mesyats, the director of the Lebedev Physics Institute in Moscow, where Ginzburg worked .
Institute
of
the
Lebedev
Physics
Moscow
worked
where
Ginzburg
Relation: org:founded_by
Anil Kumar, a former director at the consulting firm McKinsey & Co, pleaded guilty on Thursday to providing inside information to Raj Rajaratnam, the founder of the Galleon Group, in exchange for payments of at least $ 175 million from 2004 through 2009.
Rajaratnam
to
Raj
founder
the
Group
of
the
Galleon
Relation: per:parents
Gwathmey was born in 1938, the only child of painter Robert Gwathmey and his wife, Rosalie, a photographer.
born
Gwathmey
was
1938
child
the
only
Gwathmey
of
Robert
and
wife
his
Rosalie
photographer
Relation: per:cause_of_death
"It is with great sorrow that we note the passing of Merce Cunningham, who died peacefully in his home last night of natural causes", the Cunningham Dance Foundation and the Merce Cunningham Dance Company said in a statement.
Cunningham
of
Merce
die
who
peacefully
home
causes
of
natural
Relation: per:employee_of
Hwang, architect of the Pyongyang regime's ideology of "juche" or self-reliance, was once secretary of the ruling Workers' Party and a tutor to current leader Kim Jong-Il.
secretary
Hwang
was
once
Party
architect
of
Worker
and
tutor
the
ruling
'

Figure 2.12: Examples and the pruned dependency trees where the C-GCN predicted correctly. Words are shaded by the number of dimensions they contributed to h_{sent} in the pooling operation, with punctuation omitted.

where shareholder relations hold, and therefore ranks top in that relation.

2.7.4 Entity Bias in the SemEval Dataset

In our study, we observed a high correlation between the entity mentions in a sentence and its relation label in the SemEval dataset. We experimented with PA-LSTM models to analyze this phenomenon.[7] We started by simplifying every sentence in the SemEval training and dev sets to "*subject* and *object*", where *subject* and *object* are the actual entities in the sentence. Surprisingly, a trained PA-LSTM model on this data is able to achieve 65.1 F_1 on the dev set if GloVe is used to initialize word vectors, and 47.9 dev F_1 even without GloVe initialization. To further evaluate the model in a more realistic setting, we trained one model with the original SemEval training set (unmasked) and one with mentions masked in the training set, following what we have done for TACRED (masked). While the unmasked model achieves a 83.6 F_1 on the original SemEval dev set, F_1 drops drastically to 62.4 if we replace dev set entity mentions with a special <*UNK*> token to simulate the presence of unseen entities. In contrast, the masked model is unaffected by unseen entity mentions and achieves a stable dev F_1 of 74.7. This suggests that models trained without

[7]We choose the PA-LSTM model because it is more amenable to our experiments with simplified examples.

Relation	Dependency Tree Edges		
per:children	`S-PER` ← son	son → `O-PER`	`S-PER` ← survived
per:other_family	`S-PER` ← stepson	niece → `O-PER`	`O-PER` ← stepdaughter
per:employee_of	a ← member	`S-PER` ← worked	`S-PER` ← played
per:schools_attended	`S-PER` ← graduated	`S-PER` ← earned	`S-PER` ← attended
org:founded	founded → `O-DATE`	established → `O-DATE`	was ← founded
org:number_of_employees	`S-ORG` ← has	`S-ORG` → employs	`O-NUMBER` ← employees
org:subsidiaries	`S-ORG` ← `O-ORG`	`S-ORG` → 's	`O-ORG` → division
org:shareholders	buffett ← `O-PER`	shareholder → `S-ORG`	largest ← shareholder
per:children	`S-PER` ← son	son → `O-PER`	`S-PER` ← survived
per:parents	`S-PER` ← born	`O-PER` ← son	`S-PER` ← mother
per:siblings	`S-PER` ← sister	sister → `O-PER`	brother → `O-PER`
per:other_family	`S-PER` ← stepson	niece → `O-PER`	`O-PER` ← stepdaughter
per:spouse	wife → `O-PER`	`S-PER` ← wife	his ← wife
per:city_of_death	`S-PER` ← died	died → `O-CITY`	`ROOT` → died
per:city_of_birth	`S-PER` ← born	was ← born	born → `O-CITY`
per:cities_of_residence	in ← `O-CITY`	`O-CITY` ← `S-PER`	`S-PER` ← lived
per:employee_of	a ← member	`S-PER` ← worked	`S-PER` ← played
per:schools_attended	`S-PER` ← graduated	`S-PER` ← earned	`S-PER` ← attended
per:title	`O-TITLE` ← `S-PER`	as ← `O-TITLE`	former ← `S-PER`
per:charges	`S-PER` ← charged	`O-CHARGE` ← charges	`S-PER` ← faces
per:cause_of_death	died → `O-CAUSE`	`S-PER` ← died	from ← `O-CAUSE`
per:age	`S-PER` → `O-NUMBER`	`S-PER` ← died	age → `O-NUMBER`
org:alternate_names	`S-ORG` → `O-ORG`	`O-ORG` →)	(← `O-ORG`
org:founded	founded → `O-DATE`	established → `O-DATE`	was ← founded
org:founded_by	`O-PER` → founder	`S-ORG` ← `O-PER`	founder → `S-ORG`
org:top_members	`S-ORG` ← `O-PER`	director → `S-ORG`	`O-PER` ← said
org:subsidiaries	`S-ORG` ← `O-ORG`	`S-ORG` → 's	`O-ORG` → division
org:num_of_employees	`S-ORG` ← has	`S-ORG` → employs	`O-NUMBER` ← employees
org:shareholders	buffett ← `O-PER`	shareholder → `S-ORG`	largest ← shareholder
org:website	`S-ORG` → `O-URL`	`ROOT` → `S-ORG`	`S-ORG` → :
org:dissolved	`S-ORG` ← forced	forced → file	file → insolvency
org:political/religious_affiliation	`S-ORG` → group	`O-IDEOLOGY` ← group	group → established

Table 2.4: The three dependency edges that contribute the most to the classification of different relations in the dev set of TACRED. For clarity, we removed edges which 1) connect to common punctuation (i.e., commas, periods, and quotation marks), 2) connect to common preposition (i.e., of, to, by), and 3) connect tokens within the same entities. We use `PER`, `ORG`, `CHARGE`, `CAUSE` for entity types of `PERSON`, `ORGANIZATION`, `CRIMINAL_CHARGE` and `CAUSE_OF_DEATH`, respectively. We use `S-` and `O-` to denote subject and object entities, respectively. `ROOT` denotes the root node of the tree.

entities masked generalize poorly to new examples with unseen entities. Our findings call for more careful evaluation that takes dataset biases into account in future relation extraction studies.

2.8 Conclusion

In this chapter, we have demonstrated the success of a neural architecture using syntactic structures of natural language via graph convolutional networks (GCN) for relation extraction. The proposed path-centric pruning technique not only applies to our GCN-based model, but also improves the robustness of dependency-based models for relation extraction in general by removing irrelevant content from the sentence without ignoring crucial information such as negation and conjunction. We also showed through detailed analysis that this model has complementary strengths to neural sequence models, combining which set a new state of the art on the TACRED dataset we experimented on.

This success of combining powerful neural networks with a linguistically-motivated symbolic technique is a clear demonstration of how explainable data representations, such as syntactic trees, can help NLP systems quickly benefit from expert observations as inductive bias even when there is already an abundant amount of data available. This could be due to several reasons, two of which are at play here: (1) the structure of natural language is complex but linguistic analysis resurfaces the simpler underlying patterns, and (2) most problems in natural language processing have a long-tail distribution, and regardless of the amount of data, there is often data scarcity in certain slices of the task, where inductive bias can help tremendously.

The core idea of combining linguistic structures with powerful graph neural networks has also been applied to other relation extraction tasks since the original publication of this work. Specifically, Guo et al. (2019) showed that the GCN model combined with the path-centric pruning technique we proposed achieves state-of-the-art or comparable performance on various biomedical relation extraction benchmarks without further adaptation, although a better graph neural network is able to improve the performance further. Similarly, Wu et al. (2019) shows that a drop-in improvement for the graph convolutional network component further improves the performance of relation extraction.

More recently, large pre-trained language models have also demonstrated competitive performance on relation extraction (Zhang et al., 2019; Joshi et al., 2020) or even storing and recalling relational knowledge (Petroni et al., 2019) without the help of explicit syntactic structures. Since these models have also been demonstrated to capture a non-trivial amount of syntactic information (Tenney et al., 2019; Clark et al., 2019; Hewitt and Manning, 2019), the exact causal (or correlational) relationship between syntactic information and the robustness of relation extraction remains an open research question. However, regardless of the findings from future research on this topic, it remains valid that the GCN model we proposed can be more easily explained and controlled as a result of its use of syntactic structures in its computation. Moreover, the higher-level reasoning with extracted knowledge tuples is more amenable to explaining and fact-checking the underlying reasoning steps, which makes knowledge bases superior to black-box neural network models in some practical applications.

At a higher level, I have shown how combining data-driven models and linguistic insight can improve the performance of relation extraction, which has been shown to improve NLP systems' accuracy of answering questions that require multiple steps of reasoning with textual data in evaluation of KBP systems (Zhang et al., 2017). This allows us to extract knowledge from text more reliably, and perform multi-step reasoning with scalable approaches to answer questions. However, even if we had a perfect relation extraction system, knowledge-based question answering approaches still suffer from a fundamental drawback that stop us from making use of arbitrary textual knowledge. Specifically, despite their efficiency and efficacy that stem from the structured nature of their knowledge representation, knowledge bases are limited by the predefined knowledge schema they are built with. This means that (a) an emergent relation between known types of entities would only be covered if the schema is updated to cover it, (b) emergent types of entities will not be meaningfully covered until the knowledge schema has been updated with them with sufficient support of relations involving this new entity type, and (c) the knowledge we are interested in has to be expressed in typed entities and relations, which might not be trivial in all cases. What is more, knowledge bases rely on upstream systems to disambiguate entities and relations described in natural language questions against them, which renders them potentially brittle to variations in the description of entities and relations; KBs also need to be updated to cover all instances of new entities and relations when they are added even if a schema update is not involved, which is time-consuming.

These properties of knowledge bases render knowledge-based question answering systems unsuitable for answering on-demand, natural language questions in an ever-changing world, and call for a fundamentally different approach towards answering complex questions with multi-step reasoning, that is less reliant on structured knowledge representations. In the next two chapters, I will present how we can move beyond structured knowledge and build purely text-based question answering systems towards multi-step reasoning, which allows NLP systems to answer complex questions with more flexibility.

Chapter 3

Towards Open-domain Multi-step Reasoning from Raw Text

As we have seen in the previous chapter, knowledge bases can be extremely helpful for aggregating large amounts of facts extracted from textual data and answering our questions about them. With typed entities and relations, knowledge bases are great at handling complex queries that involve multi-step reasoning. Despite their effectiveness, several practical issues prevent them from providing us access to textual knowledge to the fullest extent.

First, knowledge bases require well-defined knowledge schemas to perform multi-step reasoning, which is pre-defined and inherently limiting. To capture the expressivity of natural languages, one would need to carefully define taxonomies of entities that reflect the real world, and characterize their relations as well as how they are composed in a symbolic manner. It is extremely difficult to build and maintain a knowledge base that is capable of answering all of our queries. For instance, it was estimated that only about 25% of Google searches trigger results from Knowledge Graph, Google's knowledge base for answering real-world questions that backs Google Assistant and Search.[1] Any person who has used Google Search would also be able to tell that most of the time these are shown only as information boxes by the side of the search results, rather than a direct answer to the question.

Second, the more extensive a knowledge schema is, the less likely it is that statistical models can extract it from raw text reliably. This is usually the result of a combination of a larger output space and the long-tail nature of the distribution of data samples in different types of entities and relations in natural language. On the other hand, although text-based approaches to question answering would have to deal with the same, if not more skewed, long tail effect in natural language processing, they can more easily benefit from the fuzzy matching between distributed semantic

[1] Source: `https://medium.com/backchannel/how-google-search-dealt-with-mobile-33bc09852dc9`

representations of words in context. That is, text-based systems can get away with matching words that it doesn't fully grasp based on their context and spelling to answer questions, while knowledge based approaches must map questions and textual knowledge into a rigid knowledge schema. This is similar to how anyone with a basic command of the English language can potentially answer one or two medical questions when presented a medical textbook where symptoms and diseases are described in a similar manner, without having to grasp the underlying medical knowledge accurately.

Third, even assuming that we are able to define a knowledge schema that covers a wide range of entities and relations of interest, and that we have access to a highly accurate relation extraction system to distill this information from text, updating the knowledge schema still presents significant challenges. Specifically, when new texts are made available, we need to apply the relation extraction system to them to extract new fact tuples. What is worse, when the knowledge schema itself is updated, we not only have to update the relation extraction system, but also need to potentially run the relation extraction system over all of the texts that have already been processed to update the knowledge base, before serving user queries regarding this new schema.

This is in direct conflict with the nature of how textual data and knowledge is being generated and accessed in the Internet age, where everyone can contribute to an exponentially growing body of text, and is interested in learning about the latest development on various topics as soon as they become available. To handle less structured user questions against a large collection of text on-demand, we need fundamentally different approaches to make use of textual data.

One solution is to move away from knowledge-based question answering, which requires preprocessing textual knowledge into a rigid, structured format, and towards more open-domain, retrieval-based question answering techniques. Instead of representing knowledge about the world and questions about it in a structured format, open-domain approaches simply assume access to a large collection of text that may contain the answer to questions of interest. To find the answer to questions, these open-domain question answering systems often employ information retrieval techniques to navigate large amounts of text to narrow down the set of documents that might contain the answer to a given question, then use more fine-grained fuzzy semantic matching algorithms to find the answer from within this set of documents.

One example of an open-domain question answering system is DrQA, which was originally presented by Chen et al. (2017). As can be seen in Figure 1.1 (on page 6), this system operates in two stages to answer each question: first, it searches Wikipedia for articles that potentially contains the answer with the question as the search query; then, top search results are concatenated, and fed into a neural network model to extract the answer.

The development of these systems is greatly enabled by the availability of large-scale question answering datasets such as SQuAD (Rajpurkar et al., 2016, 2018),TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019), among others. Most of these datasets share a similar format: the answer to a question is typically extracted from a local context of a paragraph or a

Context:
Ernest Christy Cline (born March 29, 1972) is an American novelist, slam poet, and screenwriter. He is known for his novels Ready Player One and Armada; he also co-wrote the screenplay for the film adaptation of Ready Player One, directed by Steven Spielberg.

Question: Which novel written by Ernest Cline was adapted into a film?

Answer: Ready Player One

Figure 3.1: An example of a context containing multiple instances of the correct answer Here, question answering systems need to rely on techniques like distant supervision to find the correct instance of the answer span. In this example, the context contains two copies of the answer, but only the context surrounding the second copy directly answers the question.

sentence, which covers all of the natural language context one needs to answer the question. When there are multiple potential spans of text that match the answer in the paragraph or paragraphs provided, there is often little or no annotation with regard to which instance is from the correct context (see Figure 3.1). Therefore, systems would need to rely on techniques like distant supervision to predict the answer from the context, and the evaluation metric typically cannot discern if the answer is from a context that directly supports the answer.

In this chapter, I present a new text-based large-scale question answering dataset that addresses these two issues. Specifically, this new dataset, HotpotQA, features natural language questions that require combining the textual knowledge from two Wikipedia articles to answer. This allows us to develop and evaluate natural language processing systems that are capable of reasoning with textual knowledge in the open domain, free from any pre-defined knowledge schemas. Moreover, each example in this dataset is provided with not only the question/answer pair and the correct context that the answer is drawn from, it also features crowd-sourced annotations for the sentences within this context that a human would use to support the answer, which is incorporated into its evaluation metric. On the one hand, this allows us to concretely gauge whether question answering systems on this dataset are obtaining the right answer for the right reasons; on the other hand, it also allows these systems to predict and provide these supporting sentences as a rationale or explanation for human verification.

Next, before introducing how this dataset is collected and its core features, I will briefly review most relevant question answering datasets, and highlight the main contributions of HotpotQA.

3.1 Background

The ability to perform reasoning and inference over natural language is an important aspect of intelligence, and can greatly help us navigate the large amount of knowledge that is encoded in text in practice. To this end, a few large-scale QA datasets have been proposed, which sparked significant progress in this direction. However, existing datasets have limitations that hinder further advancements of machine reasoning over natural language, especially in testing QA systems' ability to perform *multi-hop reasoning*, where the system has to reason with information taken from more than one document to arrive at the answer.

First, some datasets mainly focus on testing the ability of reasoning within a single paragraph or document, which involves *single-hop reasoning* most of the time. For example, in SQuAD (Rajpurkar et al., 2016) questions are designed to be answered given a single paragraph as the context, and most of the questions can in fact be answered by matching the question with a single sentence in that paragraph. As a result, it has fallen short at testing systems' ability to reason over a larger context. TriviaQA (Joshi et al., 2017) and SearchQA (Dunn et al., 2017) create a more challenging setting by using information retrieval to collect multiple documents to form the context given existing question-answer pairs. Nevertheless, most of the questions can be answered by matching the question with a few nearby sentences in one single paragraph, which is limited as it does not require more complex reasoning over multiple pieces of evidence (*e.g.*, over multiple paragraphs or documents).

Second, existing datasets that target multi-hop reasoning, such as QAngaroo (Welbl et al., 2018) and ComplexWebQuestions (Talmor and Berant, 2018), are constructed using existing knowledge bases (KBs) or schemas. As a result, these datasets are constrained by the schema of the KBs they use, and therefore the diversity of questions and answers is inherently limited. This limits our ability to develop text-based question answering systems and testing their capabilities on truly flexible user queries.

Third, all of the above datasets only provide distant supervision; *i.e.*, the systems only know what the answer is, but do not know what supporting facts lead to it. This makes it difficult for models to learn about the underlying reasoning process, as well as to make explainable predictions.

To address the above challenges, we aim at creating a QA dataset that requires reasoning over multiple documents, and doing so in natural language, without constraining itself to an existing knowledge base or knowledge schema. We also want it to provide the system with strong supervision about what text the answer is actually derived from, to help guide systems to perform meaningful and explainable reasoning.

We present HotpotQA, a large-scale dataset that satisfies these desiderata. HotpotQA is collected by crowdsourcing based on Wikipedia articles, where crowd workers are shown multiple supporting context documents and asked explicitly to come up with questions requiring reasoning about all of the documents. This ensures it covers multi-hop questions that are more natural, and are not designed with any pre-existing knowledge base schema in mind. It is important to note, however, that involving multiple documents in the reasoning process to answer a question is merely a sufficient, but not necessary, condition, for the question to involve multi-hop reasoning (Figure 3.2 shows an example that involves more than two hops of reasoning over two paragraphs of text, for instance). Here, we add this additional requirement to ensure a higher yield of natural language

Paragraph A, Return to Olympus:
[1] *Return to Olympus is the only album by the alternative rock band Malfunkshun.* [2] *It was released after the band had broken up and after lead singer Andrew Wood (later of Mother Love Bone) had died of a drug overdose in 1990.* [3] Stone Gossard, of Pearl Jam, had compiled the songs and released the album on his label, Loosegroove Records.

Paragraph B, Mother Love Bone:
[4] *Mother Love Bone was an American rock band that formed in Seattle, Washington in 1987.* [5] The band was active from 1987 to 1990. [6] *Frontman Andrew Wood's personality and compositions helped to catapult the group to the top of the burgeoning late 1980s/early 1990s Seattle music scene.* [7] *Wood died only days before the scheduled release of the band's debut album, "Apple", thus ending the group's hopes of success.* [8] The album was finally released a few months later.

Q: What was the former band of the member of Mother Love Bone who died just before the release of "Apple"?
A: Malfunkshun
Supporting facts: 1, 2, 4, 6, 7

Figure 3.2: An example of the multi-hop questions in HotpotQA. We also highlight the supporting facts in *blue italics*, which are also part of the dataset.

questions that involve multi-hop reasoning, and to test question answering systems' capability of finding multiple supporting documents to answer the question in the open-domain setting.

Besides requiring multiple documents to answer, we also ask the crowd workers to provide the supporting facts they use to answer the question, which we also provide as part of the dataset (see Figure 3.2 for an example). We have carefully designed a data collection pipeline for HotpotQA, since the collection of high-quality multi-hop questions is non-trivial. We hope that this pipeline also sheds light on future work in this direction. Finally, we also collected a novel type of questions—comparison questions—as part of HotpotQA, in which we require systems to compare two entities on some shared properties to test their understanding of both language and common concepts such as numerical magnitude. We make HotpotQA publicly available at https://HotpotQA.github.io.

3.2 Data Collection

The main goal of our work in this chapter is to collect a diverse and explainable question answering dataset that requires multi-hop reasoning. One way to do so is to define reasoning chains based on a knowledge base (Welbl et al., 2018; Talmor and Berant, 2018). However, the resulting datasets are limited by the incompleteness of entities and relations, and the lack of diversity in the question types. Instead, in this chapter, we focus on text-based question answering in order to diversify the questions and answers. The overall setting is that given some context paragraphs (*e.g.*, a few paragraphs, or the entire Web) and a question, a QA system answers the question by extracting a span of text from the context, similar to Rajpurkar et al. (2016). We additionally ensure that it is necessary to perform multi-hop reasoning to correctly answer the question.

It is non-trivial to collect text-based multi-hop questions. In our pilot studies, we found that simply giving an arbitrary set of paragraphs to crowd workers is counterproductive, because for most paragraph sets, it is difficult to ask a meaningful multi-hop question. To address this challenge, we carefully design a pipeline to collect text-based multi-hop questions.

In the subsections that follow, we will first highlight the key design choices in our pipeline, then detail how we processed and curated Wikipedia data for data collection in HotpotQA, and introduce our data collection interface.

3.2.1 Key Design Choices in HotpotQA

Building a Wikipedia Hyperlink Graph. We use the entire English Wikipedia dump as our corpus.[2] In this corpus, we make two observations: (1) hyper-links in the Wikipedia articles often naturally entail a relation between two (already disambiguated) entities in the context, which could potentially be used to facilitate multi-hop reasoning; (2) the first paragraph of each article often contains much information that could be queried in a meaningful way. Based on these observations, we extract all the hyperlinks from the first paragraphs of all Wikipedia articles. With these hyperlinks, we build a directed graph G, where each edge (a, b) indicates there is a hyperlink from the first paragraph of article a to article b.

Generating Candidate Paragraph Pairs. To generate meaningful pairs of paragraphs for multi-hop question answering with G, we start by considering an example question "when was the singer and songwriter of Radiohead born?" To answer this question, one would need to first reason that the "singer and songwriter of Radiohead" is "Thom Yorke", and then figure out his birthday in the text. We call "Thom Yorke" a *bridge entity* in this example. Given an edge (a, b) in the hyperlink graph G, the entity of b can usually be viewed as a bridge entity that connects a and b. As we observe articles b usually determine the theme of the shared context between a and b, but not all articles b are suitable for collecting multi-hop questions. For example, entities like countries are frequently referred to in Wikipedia, but don't necessarily have much in common with all incoming links. It is also difficult, for instance, for the crowd workers to ask meaningful multi-hop questions about highly technical entities like the IPv4 protocol. To alleviate this issue, we constrain the bridge entities to a set of manually curated pages in Wikipedia (see Section 3.2.3). After curating a set of pages B, we create candidate paragraph pairs by sampling edges (a, b) from the hyperlink graph such that $b \in B$.

[2] `https://dumps.wikimedia.org/` We use the dump for October 1, 2017.

Comparison Questions. In addition to questions collected using bridge entities, we also collect another type of multi-hop questions—comparison questions. The main idea is that comparing two entities from the same category usually results in interesting multi-hop questions, e.g., “Who has played for more NBA teams, Michael Jordan or Kobe Bryant?” To facilitate collecting this type of question, we manually curate 42 lists of similar entities (denoted as L) from Wikipedia.[3] To generate candidate paragraph pairs, we randomly sample two paragraphs from the same list and present them to the crowd worker.

To increase the diversity of multi-hop questions, we also introduce a subset of yes/no questions in comparison questions. This complements the original scope of comparison questions by offering new ways to require systems to reason over both paragraphs. For example, consider the entities Iron Maiden (from the UK) and AC/DC (from Australia). Questions like “Is Iron Maiden or AC/DC from the UK?” are not ideal, because one would deduce the answer is “Iron Maiden” even if one only had access to that article. With yes/no questions, one may ask “Are Iron Maiden and AC/DC from the same country?”, which requires reasoning over both paragraphs.

To the best of our knowledge, text-based comparison questions are a novel type of questions that have not been considered by previous datasets. More importantly, answering these questions usually requires arithmetic comparison, such as comparing ages given birth dates, which presents a new challenge for future model development.

Collecting Supporting Facts. To enhance the explainability of question answering systems, we want them to output a set of *supporting facts* necessary to arrive at the answer, when the answer is generated. To this end, we also collect the sentences that determine the answers from crowd workers. These supporting facts can serve as strong supervision for what sentences to pay attention to. Moreover, we can now test the explainability of a model by comparing the predicted supporting facts to the ground truth ones.

The overall procedure of data collection is illustrated in Algorithm 1. For completeness, we also include further data collection details in the following subsections, including how we preprocess Wikipedia, curate Wikipedia pages, structure bonus for our crowd workers, and what our crowd-sourcing interface looks like.

3.2.2 Data Preprocessing

We downloaded the dump of English Wikipedia of October 1, 2017, and extracted text and hyperlinks with WikiExtractor Attardi (2015).[4] We use Stanford CoreNLP 3.8.0 (Manning et al., 2014) for word and sentence tokenization. We use the resulting sentence boundaries for collection of supporting

[3]This is achieved by manually curating lists from the Wikipedia “List of lists of lists” (`https://wiki.sh/y8qv`). One example is “Highest Mountains on Earth”.

[4]Our modified version is available at `https://github.com/qipeng/wikiextractor`

Algorithm 1 Overall data collection procedure

```
Input: question type ratio $r_1 = 0.75$, yes/no ratio $r_2 = 0.5$
while not finished do
  if random() $< r_1$ then
    Uniformly sample an entity $b \in B$
    Uniformly sample an edge $(a, b)$
    Workers ask a question about paragraphs $a$ and $b$
  else
    Sample a list from $L$, with probabilities weighted by list sizes
    Uniformly sample two entities $(a, b)$ from the list
    if random() $< r_2$ then
      Workers ask a yes/no question to compare $a$ and $b$
    else
      Workers ask a question with a span answer to compare $a$ and $b$
    end if
  end if
  Workers provide the supporting facts
end while
```

facts, and use token boundaries to check whether Turkers are providing answers that cover spans of entire tokens to avoid nonsensical partial-word answers.

3.2.3 Further Data Collection Details

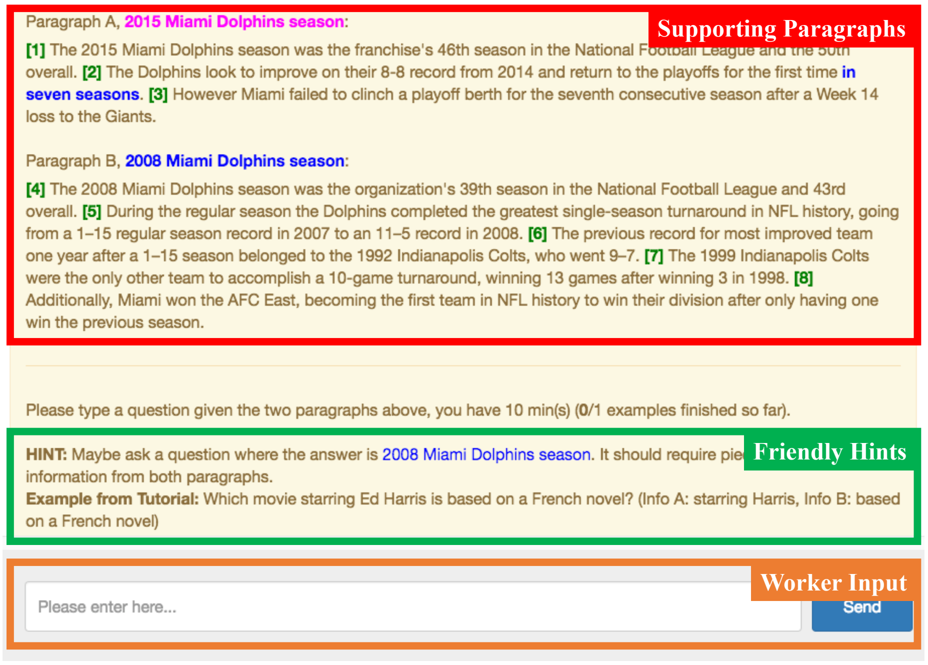
Details on Curating Wikipedia Pages. To make sure the sampled candidate paragraph pairs are intuitive for crowd workers to ask high-quality multi-hop questions about, we manually curate 591 categories from the lists of popular pages by WikiProject.[5] For each category, we sample (a, b) pairs from the graph G where b is in the considered category, and manually check whether a multi-hop question can be asked given the pair (a, b). Those categories with a high probability of permitting multi-hop questions are selected.

Bonus Structures. To incentivize crowd workers to produce higher-quality data more efficiently, we follow Yang et al. (2018b), and employ bonus structures. We mix two settings in our data collection process. In the first setting, we reward the top (in terms of numbers of examples) workers every 200 examples. In the second setting, the workers get bonuses based on their productivity (measured as the number of examples per hour).

3.2.4 Crowd Worker Interface

Our crowd worker interface is based on ParlAI (Miller et al., 2017), an open-source project that facilitates the development of dialog systems and data collection with a dialog interface. We adapt

[5] `https://en.wikipedia.org/wiki/Category:Lists_of_popular_pages_by_WikiProject`



Paragraph A, 2015 Miami Dolphins season:

[1] The 2015 Miami Dolphins season was the franchise's 46th season in the National Football League and the 50th overall. [2] The Dolphins look to improve on their 8-8 record from 2014 and return to the playoffs for the first time in seven seasons. [3] However Miami failed to clinch a playoff berth for the seventh consecutive season after a Week 14 loss to the Giants.

Paragraph B, 2008 Miami Dolphins season:

[4] The 2008 Miami Dolphins season was the organization's 39th season in the National Football League and 43rd overall. [5] During the regular season the Dolphins completed the greatest single-season turnaround in NFL history, going from a 1–15 regular season record in 2007 to an 11–5 record in 2008. [6] The previous record for most improved team one year after a 1–15 season belonged to the 1992 Indianapolis Colts, who went 9–7. [7] The 1999 Indianapolis Colts were the only other team to accomplish a 10-game turnaround, winning 13 games after winning 3 in 1998. [8] Additionally, Miami won the AFC East, becoming the first team in NFL history to win their division after only having one win the previous season.

Please type a question given the two paragraphs above, you have 10 min(s) (0/1 examples finished so far).

HINT: Maybe ask a question where the answer is 2008 Miami Dolphins season. It should require pie information from both paragraphs.
Example from Tutorial: Which movie starring Ed Harris is based on a French novel? (Info A: starring Harris, Info B: based on a French novel)

Please enter here...

Figure 3.3: Screenshot of our worker interface on Amazon Mechanical Turk.

ParlAI for collecting question answer pairs by converting the collection workflow into a system-oriented dialog. This allows us to have more control over the turkers input, as well as provide turkers with in-the-loop feedback or helpful hints to help Turkers finish the task, and therefore speed up the collection process. Specifically, we prompt crowd workers not only with the context that the question should be about, but also sometimes provide them with friendly hints regarding what the question might look like in the specific context to help speed up the collection process.

Please see Figure 3.3 for an example of the worker interface during data collection.

3.3 Processing and Benchmark Settings

We collected 112,779 valid examples in total on Amazon Mechanical Turk[6] using the ParlAI interface (Miller et al., 2017) we described in Section 3.2.4. To isolate potential single-hop questions from the desired multi-hop ones, we first split out a subset of data called *train-easy*. Specifically, we randomly sampled questions (∼3–10 per Turker) from top-contributing turkers, and categorized all their questions into the *train-easy* set if an overwhelming percentage in the sample only required reasoning over one of the paragraphs. We sampled these turkers because they contributed more than 70% of our data. This *train-easy* set contains 18,089 mostly single-hop examples.

We implemented a question answering model based on the current state-of-the-art architectures,

[6] https://www.mturk.com/

Name	Desc.	Usage	# Examples
train-easy	single-hop	training	18,089
train-medium	multi-hop	training	56,814
train-hard	hard multi-hop	training	15,661
dev	hard multi-hop	dev	7,405
test-distractor	hard multi-hop	test	7,405
test-fullwiki	hard multi-hop	test	7,405
Total			112,779

Table 3.1: Data split of HotpotQA. The splits *train-easy*, *train-medium*, and *train-hard* are combined for training. The distractor and full wiki settings use different test sets so that the gold paragraphs in the full wiki test set remain unknown to any models.

which we discuss in detail in Section 3.5.1. Based on this model, we performed a three-fold cross validation on the remaining multi-hop examples. Among these examples, the models were able to correctly answer 60% of the questions with high confidence (determined by thresholding the model loss). These correctly-answered questions (56,814 in total, 60% of the multi-hop examples) are split out and marked as the *train-medium* subset, which will also be used as part of our training set.

After splitting out *train-easy* and *train-medium*, we are left with hard examples. As our ultimate goal is to solve multi-hop question answering, we focus on questions that the latest modeling techniques are not able to answer. Thus we constrain our dev and test sets to be hard examples. Specifically, we randomly divide the hard examples into four subsets, *train-hard*, *dev*, *test-distractor*, and *test-fullwiki*. Statistics about the data split can be found in Table 3.1. In Section 3.5, we will show that combining *train-easy*, *train-medium*, and *train-hard* to train models yields the best performance, so we use the combined set as our default training set. The two test sets *test-distractor* and *test-fullwiki* are used in two different benchmark settings, which we introduce next.

We create two benchmark settings. In the first setting, to challenge the model to find the true supporting facts in the presence of noise, for each example we employ bigram tf-idf (Chen et al., 2017) to retrieve 8 paragraphs from Wikipedia as *distractors*, using the question as the query. We mix them with the 2 gold paragraphs (the ones used to collect the question and answer) to construct the **distractor** setting. The 2 gold paragraphs and the 8 distractors are shuffled before they are fed to the model. In the second setting, we fully test the model's ability to locate relevant facts as well as reasoning about them by requiring it to answer the question given the first paragraphs of all Wikipedia articles without the gold paragraphs specified. This **full wiki** setting truly tests the performance of the systems' ability at multi-hop reasoning in the wild. As we required the crowd workers to use complete entity names in the question, the majority of the questions are unambiguous in the full wiki setting. The two settings present different levels of difficulty, and would require techniques ranging from reading comprehension to information retrieval. As shown in Table 3.1, we use separate test sets for the two settings to avoid leaking information, because the

Set	MAP	Mean Rank	CorAns Rank
train-medium	41.89	288.19	82.76
dev	42.79	304.30	97.93
test	45.92	286.20	74.85

Table 3.2: Retrieval performance comparison on full wiki setting for *train-medium*, *dev* and *test* with 1,000 random samples each. MAP is in %. Mean Rank averages over retrieval ranks of two gold paragraphs. CorAns Rank refers to the rank of the gold paragraph containing the answer.

gold paragraphs are available to a model in the distractor setting, but should not be accessible in the full wiki setting.

Next, we try to gain a deeper understanding of the difference between the different splits of HotpotQA, and document our bigram tf-idf retrieval system for completeness.

3.3.1 Compare *train-medium* Split to Hard Ones

In this section, we try to understand the model's good performance on the *train-medium* split. Manual analysis shows that the ratio of multi-hop questions in *train-medium* is similar to that of the hard examples (93.3% in *train-medium* vs. 92.0% in *dev*), but one of the question types appears more frequently in *train-medium* compared to the hard splits (Type II: 32.0% in *train-medium* vs. 15.0% in *dev*, see Section 3.4 for the definition of Type II questions). These observations demonstrate that given enough training data, existing neural architectures can be trained to answer certain types and certain subsets of the multi-hop questions.

However, *train-medium* remains challenging when not just the gold paragraphs are present. In the fullwiki setting, the retrieval problem on these examples are as difficult as that on their hard cousins. For example, Table 3.2 shows the comparison between the *train-medium* split and hard examples like *dev* and *test* under retrieval metrics in the full wiki setting. As we can see, the performance gap between the *train-medium* split and *dev*/*test* is small, which implies that *train-medium* split has a similar level of difficulty as hard examples under the full wiki setting, where a retrieval model is necessary as the first processing step to find all of the supporting documents to answer the question.

3.3.2 The Inverted Index Filtering Strategy

In the full wiki setting, we adopt an efficient inverted-index-based filtering strategy for preliminary candidate paragraph retrieval. Algorithm 2 provides a detailed description of our strategy, where we set the control threshold $N = 5000$ in our experiments.

Algorithm 2 Inverted Index Filtering Strategy

```
Input: question text $q$, control threshold $N$, ngram-to-Wikidoc inverted index $\mathcal{D}$
Initialize:
Extract unigram + bigram set $r_q$ from $q$
$N_{cand} = +\infty$
$C_{gram} = 0$
while $N_{cands} > N$ do
    $C_{gram} = C_{gram} + 1$
    Set $S_{overlap}$ to be an empty dictionary
    for $w \in r_q$ do
        for $d \in \mathcal{D}[w]$ do
            if $d$ not in $S_{overlap}$ then
                $S_{overlap}[d] = 1$
            else
                $S_{overlap}[d] = S_{overlap}[d] + 1$
            end if
        end for
    end for
    $S_{cand} = \emptyset$
    for $d$ in $S_{overlap}$ do
        if $S_{overlap}[d] \geq C_{gram}$ then
            $S_{cand} = S_{cand} \cup \{d\}$
        end if
    end for
    $N_{cands} = |S_{cand}|$
end while
return $S_{cand}$
```

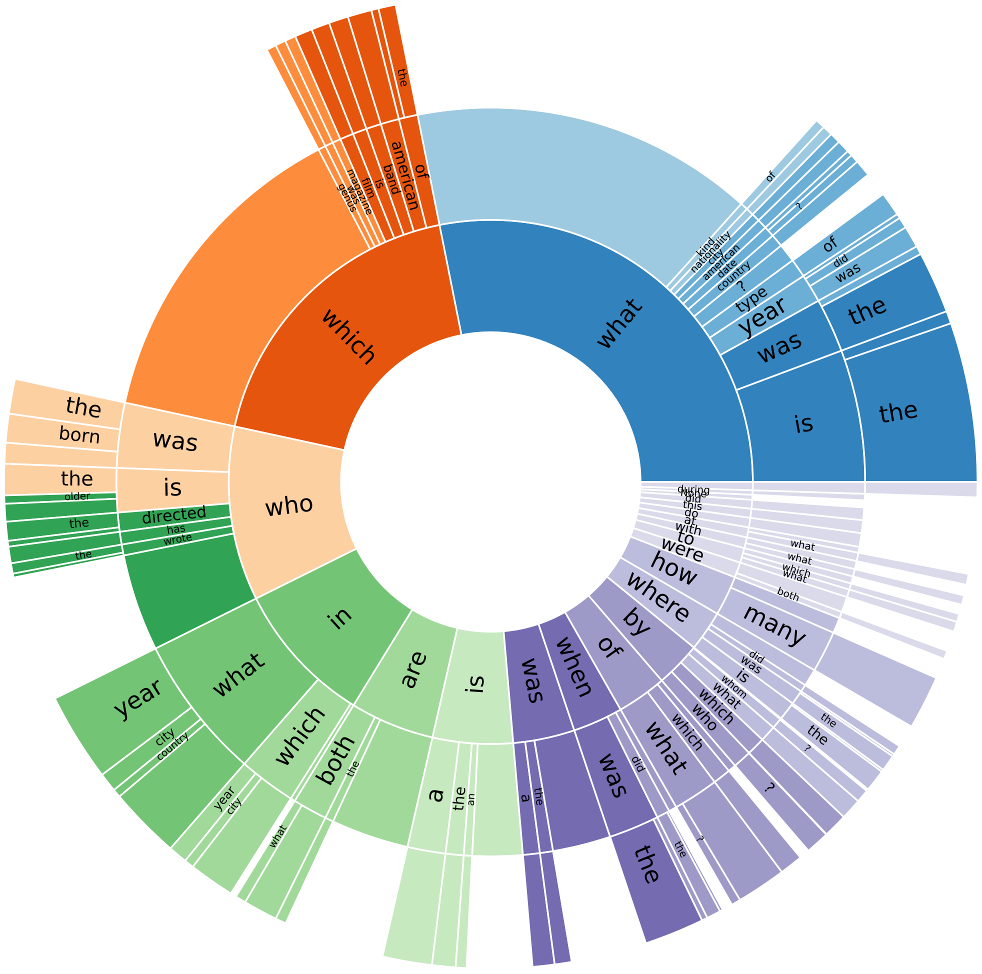


Figure 3.4: Types of questions covered in HotpotQA. Question types are extracted heuristically, starting at question words or prepositions preceding them. Empty colored blocks indicate suffixes that are too rare to show individually. See main text for more details.

3.4 Dataset Analysis

In this section, we analyze the types of questions, types of answers, and types of multi-hop reasoning covered in the dataset.

Question Types. We heuristically identified question types for each collected question. To identify the question type, we first locate the *central question word* (CQW) in the question. Since HotpotQA contains comparison questions and yes/no questions, we consider as *question words* WH-words, copulas ("is", "are"), and auxiliary verbs ("does", "did"). Because questions often involve relative clauses beginning with WH-words (*e.g.*, *"The novel author who wrote Armada has*

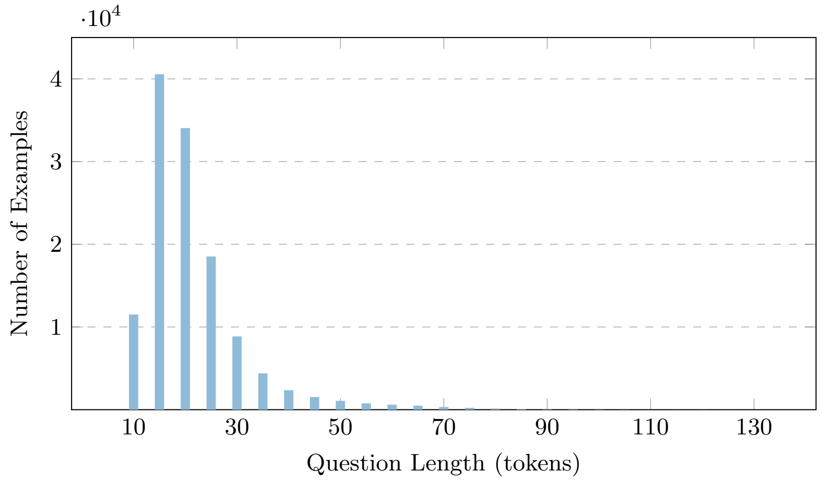


Figure 3.5: Distribution of lengths of questions in HotpotQA.

which novel that will be adapted as a film by Steven Spielberg?"), we define the CQW as the first question word in the question if it can be found in the first three tokens, or the last question word otherwise. Then, we determine question type by extracting words up to 2 tokens away to the right of the CQW, along with the token to the left if it is one of a few common prepositions (*e.g.*, in the cases of "in which" and "by whom").

We visualize the distribution of question types in Figure 3.4, and label the ones shared among more than 250 questions. As is shown, our dataset covers a diverse variety of questions centered around entities, locations, events, dates, and numbers, as well as yes/no questions directed at comparing two entities ("Are both A and B ...?"), to name a few.

To better understand the diversity of the questions in HotpotQA quantitatively, we further visualized the distribution of question lengths in the dataset in Figure 3.5. Besides being diverse in terms of types as is show above, questions also vary greatly in length, indicating different levels of complexity and details covered.

Answer Types. We further sample 100 examples from the HotpotQA dataset, and present the types of answers in Table 3.3. As can be seen, HotpotQA covers a broad range of answer types, which matches our initial analysis of question types. We find that a majority of the questions are about entities in the articles (68%), and a non-negligible amount of questions also ask about various properties like date (9%) and other descriptive properties such as numbers (8%) and adjectives (4%).

Multi-hop Reasoning Types. We also sampled 100 examples from the dev and test sets and manually classified the types of reasoning required to answer each question. Besides comparing two entities, there are three main types of multi-hop reasoning required to answer these questions, which

Answer Type	%	Example(s)
Person	30	King Edward II, Rihanna
Group / Organization	13	Cartoonito, Apalachee
Location	10	Fort Richardson, California
Date	9	10th or even 13th century
Number	8	79.92 million, 17
Artwork	8	Die schweigsame Frau
Yes/No	6	—
Adjective	4	conservative
Event	1	Prix Benois de la Danse
Other proper noun	6	Cold War, Laban Movement Analysis
Common noun	5	comedy, both men and women

Table 3.3: Types of answers in HotpotQA.

we show in Table 3.4 accompanied with examples.

Most of the questions require at least one supporting fact from each paragraph to answer. A majority of sampled questions (42%) require chain reasoning (Type I in the table), where the reader must first identify a bridge entity before the second hop can be answered by filling in the bridge. One strategy to answer these questions would be to decompose them into consecutive single-hop questions. In other question types, the bridge entity could also be used implicitly to help infer properties of other entities related to it. In some questions (Type III), the entity in question shares certain properties with a bridge entity (e.g., they are collocated), and we can infer its properties through the bridge entity. Another type of question involves locating the answer entity by satisfying multiple properties simultaneously (Type II). Here, to answer the question, one could find the set of all entities that satisfy each of the properties mentioned, and take an intersection to arrive at the final answer. Questions comparing two entities (Comparison) also require the system to understand the properties in question about the two entities (e.g., nationality), and sometimes require arithmetic such as counting (as seen in the table) or comparing numerical values ("Who is older, A or B?"). Finally, we find that sometimes the questions require more than two supporting facts to answer (Other). In our analysis, we also find that for all of the examples shown in the table, the supporting facts provided by the Turkers match exactly with the limited context shown here, showing that the supporting facts collected are of high quality.

Aside from the reasoning types mentioned above, we also estimate that about 6% of the sampled questions can be answered with one of the two paragraphs, and 2% of them unanswerable. To obtain a better understanding of the types of reasoning involved in the training set, we also randomly sampled 100 examples from *train-medium* and *train-hard* combined, and the proportions of reasoning types are: Type I 38%, Type II 29%, Comparison 20%, Other 7%, Type III 2%, single-hop 2%, and unanswerable 2%.

Reasoning Type	%	Example(s)
Inferring the ***bridge entity*** to complete the 2nd-hop question (Type I)	42	**Paragraph A:** The 2015 Diamond Head Classic was a college basketball tournament ... ***Buddy Hield*** *was named the tournament's MVP*. **Paragraph B:** ***Chavano Rainier "Buddy" Hield*** is a Bahamian professional basketball player for the **Sacramento Kings** of the NBA... **Q:** Which team does the player named 2015 Diamond Head Classic's MVP play for?
Comparing two entities (Comparison)	27	**Paragraph A:** LostAlone were a British rock band ... consisted of *Steven Battelle, Alan Williamson, and Mark Gibson*... **Paragraph B:** Guster is an American alternative rock band ... Founding members *Adam Gardner, Ryan Miller, and Brian Rosenworcel* began... **Q:** Did LostAlone and Guster have the same number of members? (**yes**)
Locating the **answer entity** by checking multiple properties (Type II)	15	**Paragraph A:** Several *current and former members of the Pittsburgh Pirates* – ... John Milner, **Dave Parker**, and Rod Scurry... **Paragraph B:** **David Gene Parker**, *nicknamed "The Cobra"*, is an American former player in Major League Baseball... **Q:** Which former member of the Pittsburgh Pirates was nicknamed "The Cobra"?
Inferring about the property of an entity in question through a ***bridge entity*** (Type III)	6	**Paragraph A:** *Marine Tactical Air Command Squadron 28* is a United States Marine Corps aviation command and control unit based at ***Marine Corps Air Station Cherry Point***... **Paragraph B:** ***Marine Corps Air Station Cherry Point*** ... is a United States Marine Corps airfield located in **Havelock, North Carolina**, USA ... **Q:** What city is the Marine Air Control Group 28 located in?
Other types of reasoning that require more than two supporting facts (Other)	2	**Paragraph A:** ... the towns of Yodobashi, **Okubo, Totsuka, and Ochiai town** *were merged into Yodobashi ward*. ... ***Yodobashi Camera*** *is a store with its name taken from the town and ward*. **Paragraph B**: ***Yodobashi Camera*** Co., Ltd. is *a major Japanese retail chain specializing in electronics, PCs, cameras and photographic equipment*. **Q:** Aside from Yodobashi, what other towns were merged into the ward which gave the major Japanese retail chain specializing in electronics, PCs, cameras, and photographic equipment it's name?

Table 3.4: Types of multi-hop reasoning required to answer questions in the HotpotQA dev and test sets. We show in ***orange bold italics*** bridge entities if applicable, *blue italics* supporting facts from the paragraphs that connect directly to the question, and **green bold** the answer in the paragraph or following the question. The remaining 8% are single-hop (6%) or unanswerable questions (2%) by our judgement.

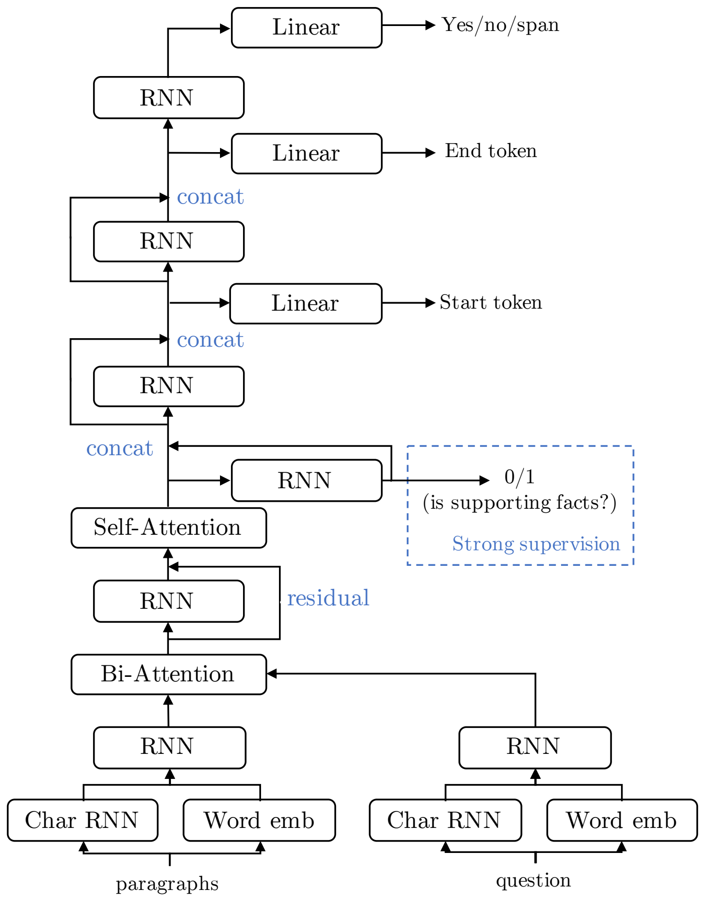


Figure 3.6: The baseline model architecture evaluated on HotpotQA. Strong supervision over supporting facts is used in a multi-task setting.

3.5 Experiments

3.5.1 Model Architecture and Training

To test the performance of leading QA systems on our data, we reimplemented the architecture described in Clark and Gardner (2018) as our baseline model (see Figure 3.6 for an illustration of our model). We note that our implementation without weight averaging achieves performance very close to what the authors reported on SQuAD (about 1 point worse in F_1). Our implemented model subsumes many of the latest technical advances on question answering at the time, including character-level models, self-attention (Wang et al., 2017), and bi-attention (Seo et al., 2017). Combining these three key components is becoming standard practice, and various then state-of-the-art or competitive architectures (Liu et al., 2018; Clark and Gardner, 2018; Wang et al., 2017; Seo et al., 2017; Pan et al., 2017; Salant and Berant, 2018; Xiong et al., 2018) on SQuAD can be viewed as

Set	MAP	Mean Rank	Hits@2	Hits@10
dev	43.93	314.71	39.43	56.06
test	43.21	314.05	38.67	55.88

Table 3.5: Retrieval performance in the full wiki setting. Mean Rank is averaged over the ranks of two gold paragraphs.

similar to our implemented model. To accommodate yes/no questions, we also add a 3-way classifier after the last recurrent layer to produce the probabilities of "yes", "no", and span-based answers. During evaluation time, we first use the 3-way output to determine whether the answer is "yes", "no", or a text span. If it is a text span, we further search for the most probable span from within the context. One of the distinctive features of HotpotQA is the presence of supporting fact sentences as explanation, which we make use of in the baseline model as a source of strong supervision, as we will detail next.

Supporting Facts as Strong Supervision. To evaluate the baseline model's performance in predicting explainable supporting facts, as well as how much they improve QA performance, we additionally design a component to incorporate such strong supervision into our model. For each sentence, we concatenate the output of the self-attention layer at the first and last positions, and use a binary linear classifier to predict the probability that the current sentence is a supporting fact. We minimize a binary cross entropy loss for this classifier. This objective is jointly optimized with the normal question answering objective in a multi-task learning setting, and they share the same low-level representations. With this classifier, the model can also be evaluated on the task of supporting fact prediction to gauge its explainability. Our overall architecture is illustrated in Figure 3.6. Though it is possible to build a pipeline system, in this work we focus on an end-to-end one, which is easier to tune and faster to train.

3.5.2 Results

We evaluate our model in the two benchmark settings. In the full wiki setting, to enable efficient tf-idf retrieval among 5,000,000+ wiki paragraphs, given a question we first return a candidate pool of at most 5,000 paragraphs using an inverted-index-based filtering strategy and then select the top 10 paragraphs in the pool as the final candidates using bigram tf-idf.[7] Here, we choose the number of final candidates as 10 to stay consistent with the distractor setting where candidates are 2 gold paragraphs plus 8 distractors. Retrieval performance of our retrieval system is shown in Table 3.5. After retrieving these 10 paragraphs, we then use the model trained in the distractor setting to evaluate its performance on these final candidate paragraphs.

[7]We have detailed the bigram tf-idf retrieval system and the filtering strategy we employ in Section 3.3.2.

Setting	Split	Answer		Sup Fact		Joint	
		EM	F_1	EM	F_1	EM	F_1
distractor	dev	44.44	58.28	21.95	66.66	11.56	40.86
distractor	test	45.46	58.99	22.24	66.62	12.04	41.37
full wiki	dev	24.68	34.36	5.28	40.98	2.54	17.73
full wiki	test	25.23	34.40	5.07	40.69	2.63	17.85

Table 3.6: Main results: the performance of question answering and supporting fact prediction in the two benchmark settings. We encourage researchers to report these metrics when evaluating their methods.

Setting	Bridge EM	Bridge F_1	Comparison EM	Comparison F_1
distractor	43.41	59.09	48.55	55.05
full wiki	19.76	30.42	43.87	50.70

Table 3.7: Performance breakdown over different question types on the dev set in the distractor setting. "Bridge" denotes questions collected using bridge entities, and "Comparison" denotes comparison questions.

Following previous work (Rajpurkar et al., 2016), we use exact match (EM) and F_1 as two evaluation metrics. Specifically, for span answers, EM assigns an example a score of 1 when the answer exactly matches one of the candidate answers, and F_1 calculates precision and recall at the unigram level. To assess the explainability of the models, we further introduce two sets of metrics involving the supporting facts. The first set focuses on evaluating the supporting facts directly, namely EM and F_1 on the set of supporting fact sentences as compared to the gold set. Here, to calculate F_1 for supporting facts, we treat each supporting sentence as a binary classification task, to aggregate precision and recall. The second set of metrics features joint metrics that combine the evaluation of answer spans and supporting facts as follows. For each example, given its precision and recall on the answer span $(P^{(\mathrm{ans})}, R^{(\mathrm{ans})})$ and the supporting facts $(P^{(\mathrm{sup})}, R^{(\mathrm{sup})})$, respectively, we calculate joint F_1 as

$$\begin{aligned} P^{(\mathrm{joint})} &= P^{(\mathrm{ans})} P^{(\mathrm{sup})}, \\ R^{(\mathrm{joint})} &= R^{(\mathrm{ans})} R^{(\mathrm{sup})}, \\ \text{Joint } \mathrm{F}_1 &= \frac{2P^{(\mathrm{joint})} R^{(\mathrm{joint})}}{P^{(\mathrm{joint})} + R^{(\mathrm{joint})}}. \end{aligned}$$

Joint EM is 1 only if both tasks achieve an exact match and otherwise 0. Intuitively, these metrics penalize systems that perform poorly on either task, and only systems that are predicting the right answer for the right reasons will excel. All metrics are evaluated example-by-example, and then averaged over examples in the evaluation set.

Setting	EM	F_1
our model	44.44	58.28
– sup fact	42.79	56.19
– sup fact, self attention	41.59	55.19
– sup fact, char model	41.66	55.25
– sup fact, train-easy	41.61	55.12
– sup fact, train-easy, train-medium	31.07	43.61
gold only	48.38	63.58
sup fact only	51.95	66.98

Table 3.8: Ablation study of question answering performance on the dev set in the distractor setting. "– sup fact" means removing strong supervision over supporting facts from our model. "– train-easy" and "– train-medium" means discarding the according data splits from training. "gold only" and "sup fact only" refer to using the gold paragraphs or the supporting facts as the only context input to the model.

The performance of our model on the benchmark settings is reported in Table 3.6, where all numbers are obtained with strong supervision over supporting facts. From the distractor setting to the full wiki setting, expanding the scope of the context increases the difficulty of question answering. The performance in the full wiki setting is substantially lower, which poses a challenge to existing techniques on retrieval-based question answering. Overall, model performance in all settings is significantly lower than human performance as we will show in Section 3.5.3, which indicates that more technical advancements are needed in future work.

We also investigate the explainability of our model by measuring supporting fact prediction performance. Our model achieves more than 60% supporting fact prediction F_1 and about 40% joint F_1, which indicates there is room for further improvement in terms of explainability.

In Table 3.7, we break down the performance on different question types. In the distractor setting, comparison questions are more challenging than questions involving bridge entities (as defined in Section 3.2), which indicates that this novel question type might not be well-modeled by existing neural architectures. In the full wiki setting, the performance of bridge entity questions drops significantly while that of comparison questions decreases only marginally. This is because both entities usually appear in the comparison questions, and thus it is much easier for the retrieval system to find both supporting paragraphs for these questions. Combined with the retrieval performance in Table 3.5, we believe that the deterioration in the full wiki setting in Table 3.6 is largely due to the difficulty of retrieving both entities.

We perform an ablation study in the distractor setting, and report the results in Table 3.8. Both self-attention and character-level models contribute notably to the final performance, which is consistent with prior work. This means that techniques targeted at single-hop QA are still somewhat effective in our setting. Moreover, removing strong supervision over supporting facts decreases performance, which demonstrates the effectiveness of our approach and the usefulness of the supporting facts. We establish an estimate of the upper bound of strong supervision by only considering the supporting facts as the oracle context input to our model, which achieves a more than 10% F_1 improvement over not using the supporting facts. Compared with the gain of strong supervision in our model ($\sim$2% in F_1), our proposed method of incorporating supporting facts supervision is most likely suboptimal, and we leave the challenge of better modeling to future work. Finally, we show that combining all data splits (*train-easy*, *train-medium*, and *train-hard*) yields the best performance, which is adopted as the default setting.

3.5.3 Establishing Human Performance

To establish human performance on our dataset, we randomly sampled 1,000 examples from the dev and test sets, and had at least three additional Turkers provide answers and supporting facts for these examples. As a baseline, we treat the answer from the original Turker during data collection as the prediction, and the newly collected answers and supporting facts as references, to evaluate human performance. For each example, we choose the answer and supporting fact reference that maximize the F_1 score to report the final metrics to reduce the effect of ambiguity, as is standard in previous work (Rajpurkar et al., 2016).

As can be seen in Table 3.9, the original crowd worker achieves very high performance in both finding supporting facts, and answering the question correctly. If the baseline model were provided with the correct supporting paragraphs to begin with, it achieves parity with the crowd worker in finding supporting facts, but still falls behind at finding the actual answer by a large margin. When distractor paragraphs are present, the performance gap between the baseline model and the crowd worker on both tasks is enlarged to $\sim$30% for both joint EM and joint F_1.

We further establish the upper bound of human performance in HotpotQA, by taking the maximum EM and F_1 for each example. Here, we use each Turker's answer in turn as the prediction, and evaluate it against all other workers' answers before taking the maximum over all different combinations. As can be seen in Table 3.9, most of the metrics are close to 100%, illustrating that on most examples, at least a subset of Turkers strongly agree with each other, showing high inter-annotator agreement. We also note that crowd workers agree less on supporting facts, which could reflect that this task is inherently more subjective than answering the question.

3.6 Related Work

Various large-scale QA datasets have been constructed in recent years. We categorize them in four categories in our discussion.

Setting	Answer		Sp Fact		Joint	
	EM	F_1	EM	F_1	EM	F_1
gold only	65.87	74.67	59.76	90.41	41.54	68.15
distractor	60.88	68.99	30.99	74.67	20.06	52.37
Human	83.60	91.40	61.50	90.04	52.30	82.55
Human Upperbound	96.80	98.77	87.40	97.56	84.60	96.37

Table 3.9: Comparing baseline model performance with human performance on 1,000 random samples. "Human Upperbound" stands for the upperbound on annotator performance on HotpotQA. For details on how this upperbound is established please refer to the main body.

Single-document datasets. SQuAD (Rajpurkar et al., 2016, 2018) questions that are relatively simple because they usually require no more than one sentence in the paragraph to answer. It is further constrained that the answer must appear as a span of text in the supporting paragraph, which enables evaluating QA systems' performance based on automatic metrics. However, questions collected in this single-paragraph setting are also less friendly to the open-context setting we are interested in, because they often don't contain sufficient information about the context. More recently, Natural Questions Kwiatkowski et al. (2019) addresses some of these challenges in the open-domain by deriving questions from users' Google search queries. However, these datasets are limited to questions that require reasoning about a local context of a paragraph, and cannot help us develop or evaluate QA systems that are capable of more complex text-based reasoning in the open domain.

Multi-document datasets. TriviaQA (Joshi et al., 2017) and SearchQA (Dunn et al., 2017) contain question answer pairs that are accompanied with more than one document as the context. This further challenges QA systems' ability to accommodate longer contexts. However, since the supporting documents are collected after the question answer pairs with information retrieval, the questions are not guaranteed to involve interesting reasoning between multiple documents. Moreover, answers are induced in a distantly supervised manner, which does not guarantee the existence of an instance of the answer span that actually answers the question.

KB-based multi-hop datasets. Recent datasets like QAngaroo (Welbl et al., 2018) and ComplexWebQuestions (Talmor and Berant, 2018) explore different approaches of using pre-existing knowledge bases (KB) with pre-defined logic rules to generate valid QA pairs, to test QA models' capability of performing multi-hop reasoning. In each instance, questions are generated by taking a chain of entities or rules of reasoning in the KB or logical forms, and the answer is generated directly from the knowledge base. The diversity of questions and answers is largely limited by the fixed KB schemas or logical forms, regardless of whether they have been re-stylized to be closer to natural language through crowd-sourcing. Furthermore, these QA datasets could also fall prey to the incompleteness of the KBs – if a factoid is not populated into the knowledge base but is supported by textual evidence, they are at risk of generating questions that seemingly require multi-hop reasoning, but are perfectly answerable with just one document.

Free-form answer-generation datasets. MS MARCO (Nguyen et al., 2016) contains 100 thousand user queries from Bing Search with human generated answers from over 1 million passages from more than 200 thousand documents. On this dataset, systems generate free-form answers and are evaluated by automatic metrics such as ROUGE-L and BLEU-1. However, the reliability of these metrics is questionable because they have been shown to correlate poorly with human judgement (Novikova et al., 2017).

3.7 Conclusion

In this chapter, I have presented HotpotQA, a large-scale question answering dataset aimed at facilitating the development of QA systems capable of performing explainable, multi-hop reasoning over diverse natural language. Aside from multi-hop questions that require systems to infer about entities that are not directly mentioned in the question, HotpotQA also features a new type of factoid comparison questions to test systems' ability to extract and compare various entity properties in text. I believe that this is an important step for us to build QA systems that are capable of aggregating knowledge from multiple sources of supporting facts on-demand without a predefined knowledge schema, so that NLP systems can help us explore and make use of the knowledge contained in large collections of text without always having to go through the time-consuming procedure of preprocessing it into a structured format, or converting it into a sequence of simple text-matching queries.

I have also demonstrated that explainability is not just a nice property to have for the systems and datasets we are building. The supporting fact sentences that we have collected for each question not only allow QA systems to provide them as an explanation at test time, but can also be used as a source of strong supervision, and help QA systems reduce the negative effect of weak supervision and achieve better performance, despite the simple technique we employed to make use of this information. This is also a demonstration of the efficacy of human annotations when performed at the right level of abstraction—the supporting sentence annotation we chose requires minimal additional effort for our crowd workers, but reveals important bits of information to help systems disambiguate between potentially confusing contexts to extract the answer from.

Since the publication of HotpotQA, there has been a lot of community interest in building QA systems that are capable of multi-step reasoning. Many approaches involve building graph structures on top of the entities mentioned in the Wikipedia paragraphs that HotpotQA questions are based on, where graph neural networks are applied to perform multi-step reasoning to arrive at the answer (Qiu et al., 2019; Ye et al., 2019; Fang et al., 2019; Tu et al., 2020). From a different perspective, researchers have also approached answering these questions by decomposing them into simpler, single-hop questions, and answering the resulting series of questions step by step (Min et al., 2019b; Perez et al., 2020).

Researchers have also conducted retrospective studies on HotpotQA, especially its limited-context *distractor* setting. Chen and Durrett (2019) and Min et al. (2019a) have shown separately that there are exploitable artifacts in the design of the distractor setting. One of the salient issues is that since distractor paragraphs are selected via information retrieval with the original question as the query, most of the distractor paragraphs are only confusable with one of the paragraphs for bridge questions, because one of the paragraphs is usually easier to retrieve from the information in the question. Further, in many cases the answer lies in the paragraph that is less straightforward to retrieve, which means systems can more easily locate where the answer is as long as the outlier paragraph is identified for these questions.

These findings have helped redirect some of the community's research focus to the more challenging and practical setting of HotpotQA, the full wiki setting. I have demonstrated that this dataset is challenging for existing QA systems in terms of finding the necessary supporting facts to answer the question, especially when given the challenge of finding these supporting facts from a large text collection like the Wikipedia. What are the main causes behind the poor performance of these systems, and can we build new NLP systems that are capable of handling these complex questions in the open-domain setting?

In the next chapter, I will introduce how I develop an efficient and explainable system that performs complex reasoning in the open domain, by training the system to iterate between retrieving new supporting evidence and reading retrieved content, much like what a human would do when solving these problems.

Chapter 4

Iterative Retrieval and Reading for Multi-step Reasoning

Open-domain question answering (QA) is an important means for natural language processing systems to help us to make use of knowledge in large text corpora and enables diverse queries. As I have covered in the beginning of the previous chapter, text-based question answering systems are much more suitable for on-demand user queries in an ever-changing world compared to knowledge-based question answering systems, which require well-defined knowledge schemas, accurate relation extraction models, and a large amount of preprocessing before the system can serve any user queries on new textual knowledge.

In the meantime, one of the largest weaknesses of open-domain question answering systems, when compared to their knowledge-based counterparts, is that they are typically poor at their capability of performing multi-step reasoning with multiple pieces of text. This is rooted in the fundamental approach that many of these open-domain QA systems employ to look for supporting documents or paragraphs to answer questions. Inspired by the TREC QA competitions,[1] Chen et al. (2017) proposed one of the first neural-network-based open-domain QA systems that operate largely in two stages: retrieval and reading. Specifically, when given a question, the system first looks up a large search index with the question itself as the search query, in hope of finding documents that might contain the answer, and then concatenates the top search results to answer the question with a more accurate reading comprehension model (see Figure 1.1 on page 6). Most follow-up work in this domain focuses on improving different components in the process, including the reading comprehension model (Clark and Gardner, 2018), the retrieval system (Asai et al., 2020), and the reranking mechanism for top retrieval results (Wang et al., 2018).

Despite their improvement upon the original model by Chen et al. (2017), these systems still

[1] `http://trec.nist.gov/data/qamain.html`

adopt the retrieve-and-read processing stages. Therefore, they would usually fall short at answering more complex questions that require multiple supporting facts to answer, like the ones we have presented in the previous chapter. For instance, while these systems can easily handle questions like *"When did Sundar Pichai join Google?"*, they would struggle to answer questions like *"When did Google's current CEO join Google?"*. The reason is intuitive: the first question directly mentions the entity *Sundar Pichai*, on whose Wikipedia page we would expect to find the answer to the question. However, the second question does not lend itself to retrieving information about *Sundar Pichai* as easily, and therefore systems overly reliant on finding the supporting facts to answer the question from a single search query would have a harder time.

In this chapter, I will present a solution to this problem that breaks away from this two-stage approach, which I think is fundamentally ill-equipped at dealing with more complex questions. The approach in this chapter will draw from the idea of how humans solve complex questions like this when faced with them. Instead of typing the entire question into the search bar of one's favorite search engine and hoping that the top search result would contain the answer, which would work especially poorly if the question weren't about famous entities like *Google* or *Sundar Pichai*, a human would probably attempt to decompose the complex question a bit, and query the search engine in an incremental manner. That is, a typical person will probably first attempt to find out whom the *CEO of Google* is through searching on the Web and reading some top search results, then turning to issue a different search query more targeted at either *Sundar Pichai* or *When did Sundar Pichai join Google*.

Similar to what a human would do, the system I will present in this chapter iterates between retrieving more evidence and reading the retrieved information to determine either the answer to the original question, or a follow-up search query to issue for further retrieval. The system is also designed to generate natural language search queries, which not only enables it to leverage efficient, off-the-shelf text-based search engines, but also makes its reasoning steps in the open domain explainable and controllable. Before introducing the approach I take, I will first review related developments in the community, and highlight my contributions to addressing the problem of multi-step reasoning for question answering.

4.1 Background and Contributions

Fueled by the recently proposed large-scale QA datasets such as SQuAD (Rajpurkar et al., 2016, 2018) and TriviaQA (Joshi et al., 2017), much progress has been made in open-domain question answering. Chen et al. (2017) proposed a two-stage approach of retrieving relevant content with the question, then reading the paragraphs returned by the information retrieval (IR) component to arrive at the final answer. This "*retrieve and read*" approach has since been adopted and extended in various open-domain QA systems (Nishida et al., 2018; Kratzwald and Feuerriegel, 2018), but it is

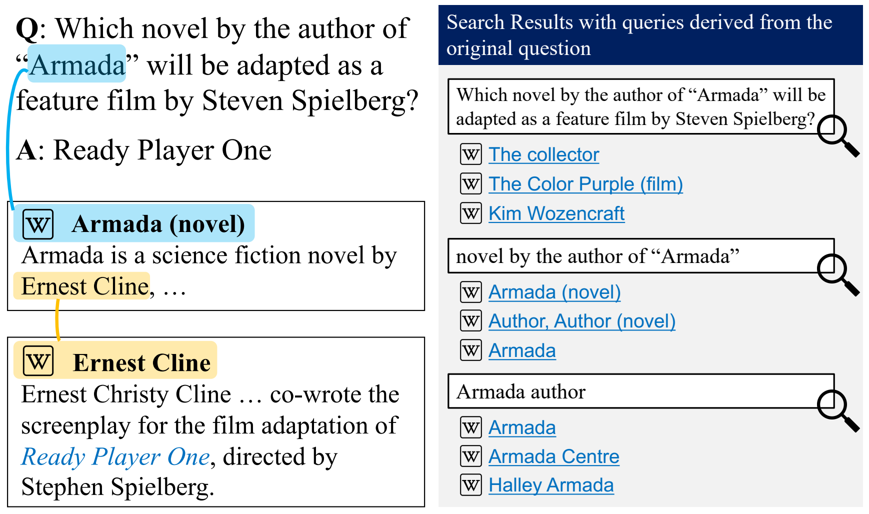

Q: Which novel by the author of "Armada" will be adapted as a feature film by Steven Spielberg?
A: Ready Player One
Armada (novel)
Armada is a science fiction novel by Ernest Cline, …
Ernest Cline
Ernest Christy Cline … co-wrote the screenplay for the film adaptation of *Ready Player One*, directed by Stephen Spielberg.
Search Results with queries derived from the original question
Which novel by the author of "Armada" will be adapted as a feature film by Steven Spielberg?
The collector
The Color Purple (film)
Kim Wozencraft
novel by the author of "Armada"
Armada (novel)
Author, Author (novel)
Armada
Armada author
Armada
Armada Centre
Halley Armada

Figure 4.1: An example of an open-domain multi-hop question from the HotpotQA dev set, where "Ernest Cline" is the missing entity. Note from the search results that it cannot be easily retrieved based on merely the question. (Best viewed in color)

inherently limited to answering questions that do not require multi-hop/multi-step reasoning. This is because for many multi-hop questions, not all the relevant context can be obtained in a single retrieval step (*e.g.*, "Ernest Cline" in Figure 4.1).

More recently, the emergence of multi-hop question answering datasets such as QAngaroo (Welbl et al., 2018) and HotpotQA (Yang et al., 2018a) has sparked interest in multi-hop QA in the research community. Designed to be more challenging than SQuAD-like datasets, they feature questions that require context of more than one document to answer, testing QA systems' abilities to infer the answer in the presence of multiple pieces of evidence and to efficiently find the evidence in a large pool of candidate documents. However, since these datasets are still relatively new, most of the existing research focuses on the few-document setting where a relatively small set of context documents is given, which is guaranteed to contain the "gold" context documents, all those from which the answer comes (De Cao et al., 2019; Zhong et al., 2019).

In this chapter, I present GoldEn (Gold Entity) Retriever. Rather than relying purely on the original question to retrieve passages, the central innovation is that at each step the model also uses IR results from previous hops of reasoning to generate a new natural language query and retrieve new evidence to answer the original question. For the example in Figure 4.1, GoldEn Retriever would first generate a query to retrieve *Armada (novel)* based on the question, then query for *Ernest Cline* based on newly gained knowledge in that article. This allows GoldEn Retriever to leverage off-the-shelf, general-purpose IR systems to scale open-domain multi-hop reasoning to millions of documents efficiently, and to do so in an interpretable manner. Combined with a QA module that extends BiDAF++ (Clark and Gardner, 2018), the final system outperforms the best previously published system on the open-domain (fullwiki) setting of HotpotQA without using powerful pretrained language models like BERT (Devlin et al., 2019).

The main contributions of this chapter are: (a) a novel iterative retrieve-and-read framework capable of multi-hop reasoning in open-domain QA; (b) a natural language query generation approach that guarantees interpretability in the multi-hop evidence gathering process; (c) an efficient training procedure to enable query generation with minimal supervision signal that significantly boosts recall of gold supporting documents in retrieval. For reproducibility, I have made the code and pretrained models for the proposed approach available at `https://github.com/qipeng/golden-retriever`.

4.2 Related Work

Open-domain question answering (QA) Inspired by the series of TREC QA competitions,[2] Chen et al. (2017) were among the first to adapt neural QA models to the open-domain setting. They built a simple inverted index lookup with TF-IDF on the English Wikipedia, and used the question as the query to retrieve top 5 results for a reader model to produce answers with. Recent work on open-domain question answering largely follow this retrieve-and-read approach, and focus on improving the information retrieval component with question answering performance in consideration (Nishida et al., 2018; Kratzwald and Feuerriegel, 2018; Nogueira et al., 2019). However, these one-step retrieve-and-read approaches are fundamentally ill-equipped to address questions that require multi-hop reasoning, especially when necessary evidence is not readily retrievable with the question.

Multi-hop QA datasets QAngaroo (Welbl et al., 2018) and HotpotQA (Yang et al., 2018a) are among the largest-scale multi-hop QA datasets to date. While the former is constructed around a knowledge base and the knowledge schema therein, the latter adopts a free-form question generation process in crowdsourcing and span-based evaluation. Both datasets feature a few-document setting where the gold supporting facts are provided along with a small set of distractors to ease the computational burden. However, researchers have shown that this sometimes results in gameable contexts, and thus does not always test the model's capability of multi-hop reasoning (Chen and Durrett, 2019; Min et al., 2019a). Therefore, in this chapter, I focus on the fullwiki setting of HotpotQA, which features a truly open-domain setting with more diverse questions.

Multi-hop QA systems At a broader level, the need for multi-step searches, query task decomposition, and subtask extraction has been clearly recognized in the IR community (Hassan Awadallah et al., 2014; Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017), but multi-hop QA has only recently been studied closely with the release of large-scale datasets. Much research has focused on enabling multi-hop reasoning in question answering models in the few-document setting, *e.g.*, by modeling

[2] `http://trec.nist.gov/data/qamain.html`

entity graphs (De Cao et al., 2019) or scoring answer candidates against the context (Zhong et al., 2019). These approaches, however, suffer from scalability issues when the number of supporting documents and/or answer candidates grow beyond a few dozen. Ding et al. (2019) apply entity graph modeling to HotpotQA, where they expand a small entity graph starting from the question to arrive at the context for the QA model. However, centered around entity names, this model risks missing purely descriptive clues in the question (*e.g.*, clues like *"adapted as a feature film"*). Das et al. (2019) propose a neural retriever trained with distant supervision to bias towards paragraphs containing answers to the given questions, which is then used in a multi-step reader-reasoner framework. This does not fundamentally address the discoverability issue in open-domain multi-hop QA, however, because usually not all the evidence can be directly retrieved with the question. Besides, the neural retrieval model lacks explainability, which is crucial in real-world applications. Talmor and Berant (2018) instead propose to answer multi-hop questions at scale by decomposing the question into sub-questions and perform iterative retrieval and question answering, which shares very similar motivations as what I propose in this chapter. However, the questions studied in that work are based on logical forms of a fixed schema, which yields additional supervision for question decomposition but limits the diversity of questions. More recently, Min et al. (2019b) apply a similar idea to HotpotQA, but this approach similarly requires additional annotations for decomposition, and the authors did not apply it to iterative retrieval.

4.3 Model

In this section, I formally define the problem of open-domain multi-hop question answering, and motivate the architecture of the proposed GoldEn (Gold Entity) Retriever model. I then detail the query generation components as well as how to derive supervision signal for them, before concluding with the QA component.

4.3.1 Problem Statement

I define the problem of *open-domain multi-hop QA* as one involving a question q, and S relevant (gold) supporting context documents $d_1, \ldots, d_S$ which contain the desired answer a. These S supporting documents usually form a chain or graph of reasoning necessary to arrive at the answer, and they come from a large corpus of documents $\mathcal{D}$ where $|\mathcal{D}| \gg S$. In this chain or graph of reasoning, the supporting documents are usually connected via shared entities or textual similarities (*e.g.*, they describe similar entities or events), but these connections do not necessarily conform to any predefined knowledge schema.

I contrast this to what I call the *few-document setting* of multi-hop QA, where the QA system is presented with a small set of documents $\mathcal{D}_{\text{few-doc}} = \{d_1, \ldots, d_S, d'_1, \ldots, d'_D\}$, where $d'_1, \ldots, d'_D$ comprise a small set of "distractor" documents that test whether the system is able to pick out the correct set of supporting documents in the presence of noise. This setting is suitable for testing QA systems' ability to perform multi-hop reasoning given the gold supporting documents with bounded computational budget, but I argue that it is far from a realistic one. In practice, an open-domain QA system has to locate all gold supporting documents from $\mathcal{D}$ on its own, and as shown in Figure 4.1, this is often difficult for multi-hop questions based on the original question alone, as not all documents comprising the gold context, from which the correct answer is drawn, are easily retrievable given the question.

To address this gold context discoverability issue, I argue that it is necessary to move away from a single-hop retrieve-and-read approach where the original question serves as the search query. In the next section, I introduce GoldEn Retriever, which addresses this problem by iterating between retrieving more documents and reading the context for multiple rounds. Note that although I only consider extractive, or span-based, QA tasks, the problem statement and the proposed method apply to generative QA tasks as well.

4.3.2 Model Overview

Essentially, the challenge of open-domain multi-hop QA lies in the fact that the information need of the user ($q \rightarrow a$) cannot be readily satisfied by any information retrieval (IR) system that models merely the similarity between the question q and the documents. This is because the true information need will only unfold with progressive reasoning and discovery of supporting facts (*e.g.*, in the case of the Armada novel example in Figure 4.1, the name of the author is not specified in the original question and only revealed once we start to find the answer to it). Therefore, one cannot rely solely on a similarity-based IR system for such iterative reasoning, because the potential pool of relevant documents grows exponentially with the number of hops of reasoning.

To this end, I propose GoldEn (Gold Entity) Retriever, which makes use of the gold document[3] information available in the QA dataset at training time to iteratively query for more relevant supporting documents during each hop of reasoning. Instead of relying on the original question as the search query to retrieve all supporting facts, or building computationally expensive search engines that are less interpretable to humans, I propose to leverage text-based IR engines for explainability, and generate different search queries as each reasoning step unfolds. In the very first hop of reasoning, GoldEn Retriever is presented the original question q, from which it generates a search query q_1 that retrieves supporting document d_1.[4] Then for each of the subsequent reasoning steps ($k = 2, \dots, S$), GoldEn Retriever generates a query q_k from the question and the available context, $(q, d_1, \dots, d_{k-1})$. This formulation allows the model to generate queries based on information

[3]In HotpotQA, since all of the supporting documents are drawn from Wikipedia, they usually describe entities. Thus I use "documents" and "entities" interchangeably in this chapter.

[4]For notational simplicity, d_k denotes the supporting document needed to complete the k-th step of reasoning. I also assume that the goal of each IR query is to retrieve one and only one gold supporting document in its top n results.

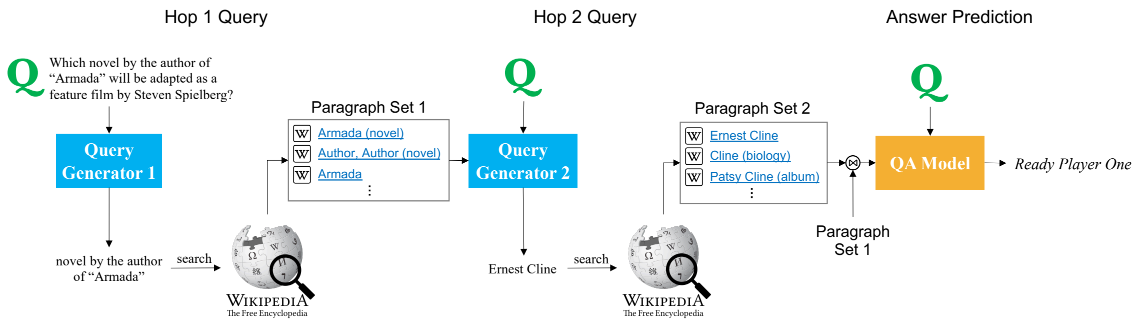


Figure 4.2: Model overview of GoldEn (Gold Entity) Retriever. Given an open-domain multi-hop question, the model iteratively retrieves more context documents, and concatenates all retrieved context for a QA model to answer from.

revealed in the supporting facts (see Figure 4.2, for example).

Note that GoldEn Retriever is much more efficient, scalable, and explainable at retrieving gold documents compared to its neural retrieval counterparts. This is because GoldEn Retriever does not rely on a QA-specific IR engine tuned to a specific dataset, where adding new documents or question types into the index can be extremely inefficient. Further, GoldEn Retriever generates queries in natural language, making it friendly to human interpretation and verification. One core challenge in GoldEn Retriever, however, is to train query generation models in an efficient manner, because the search space for potential queries is enormous and off-the-shelf IR engines are not end-to-end differentiable. I outline my solution to this challenge in the following sections.

4.3.3 Query Generation

For each reasoning step, we need to generate the search query given the original question q and some context of documents we have already retrieved (initially empty). This query generation problem is conceptually similar to the QA task in that they both map a question and some context to a target, only instead of an answer, the target here is a search query that helps retrieve the desired supporting document for the next reasoning step. Therefore, I formulate the query generation process as a question answering task.

To reduce the potentially large space of possible queries, I favor a QA model that extracts text spans from the context over one that generates free-form text as search queries. I therefore employ DrQA's Document Reader model (Chen et al., 2017), which is a relatively light-weight recurrent neural network QA model that has demonstrated success in few-document QA. I adapt it to query generation as follows.

For each reasoning step $k = 1, \ldots, S$, given a question q and some *retrieval context* C_k which *ideally* contains the gold supporting documents $d_1, \ldots, d_{k-1}$, I aim to generate a search query q_k that helps us retrieve d_k for the next reasoning step. A Document Reader model is trained to select a span from C_k as the query

$$q_k = G_k(q, C_k),$$

where G_k is the query generator at step k. This query is then used to search for supporting documents, which are concatenated with the current retrieval context to update it

$$C_{k+1} = C_k \bowtie \mathrm{IR}_n(q_k)$$

where $\mathrm{IR}_n(q_k)$ is the top n documents retrieved from the search engine using q_k, and $\bowtie$ is a concatenation operator that combines the current retrieval context with newly retrieved documents. Here, $C_1 = q$, because in the first hop we can only select a span from the original question as a search query.[5] At the end of the retrieval steps, I provide q as question along with C_S as context to the final few-document QA component detailed in Section 4.3.5 to obtain the final answer to the original question. Note that in practice this iterative retrieve and read scheme can be executed for more than or fewer than S steps, where S is the number of gold documents necessary to answer each question. This quantity also does not have to be fixed, and instead could be dynamically determined by the model—the proposed method in this chapter will apply regardless. For simplicity, however, in this chapter I consider the special case where the system iterates exactly S times, where $S = 2$ for HotpotQA.

To train the query generators, I follow the steps above to construct the retrieval contexts, but during training time, when d_k is not part of the IR result, I replace the lowest ranking document with d_k before concatenating it with C_k to make sure the downstream models have access to necessary context to predict the correct search query or answer from.

4.3.4 Deriving Supervision Signal for Query Generation

When deriving supervision signal to train the query generators, the potential search space is enormous for each step of reasoning even if we constrain ourselves to predicting spans from the context. This is aggravated by multiple hops of reasoning required by the question. Because the context of later steps of retrieval depend on queries of earlier steps in the process, the search space for queries grows exponentially in the number of reasoning steps we would like to consider. One solution to this issue is to train the query generators with reinforcement learning (RL) techniques (*e.g.*, REINFORCE (Sutton et al., 2000)), where (Nogueira and Cho, 2017) and (Buck et al., 2018) are examples of one-step query generation with RL. However, it is computationally inefficient, and has high variance especially for the second reasoning step and forward, again because the context depends greatly on what queries have been chosen previously and their search results.

Instead, I propose to leverage the limited supervision we have about the gold supporting documents $d_1, \ldots, d_S$ to narrow down the search space. The key insight I base my approach on is that at

[5]In the query result, the title of each document is delimited with special tokens `<t>` and `</t>` before concatenation.

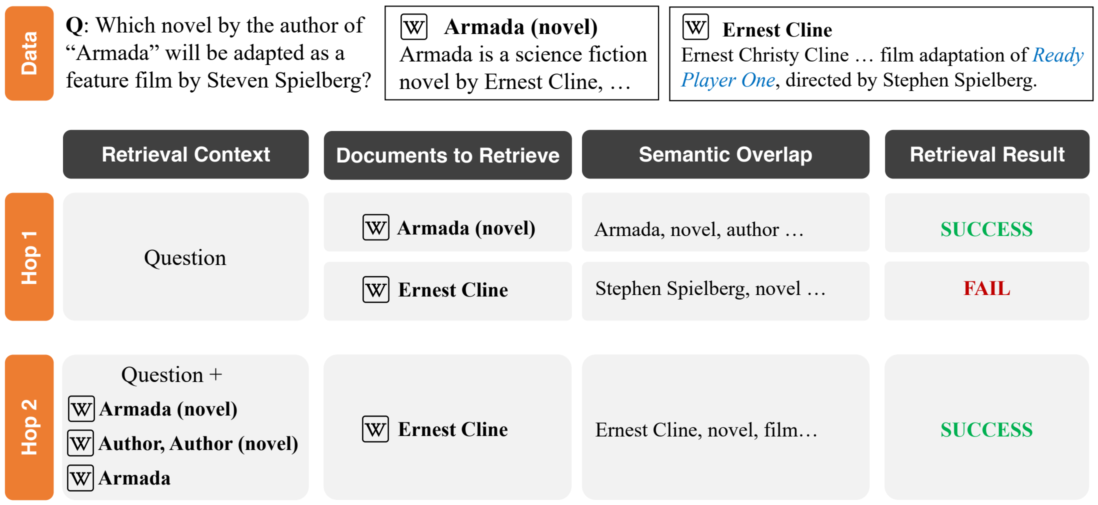

Data
Q: Which novel by the author of "Armada" will be adapted as a feature film by Steven Spielberg?
W Armada (novel)
Armada is a science fiction novel by Ernest Cline, …
W Ernest Cline
Ernest Christy Cline … film adaptation of Ready Player One, directed by Stephen Spielberg.
Retrieval Context
Documents to Retrieve
Semantic Overlap
Retrieval Result
Hop 1
Question
W Armada (novel)
Armada, novel, author …
SUCCESS
W Ernest Cline
Stephen Spielberg, novel …
FAIL
Hop 2
Question +
W Armada (novel)
W Author, Author (novel)
W Armada
W Ernest Cline
Ernest Cline, novel, film…
SUCCESS

Figure 4.3: Illustration of the oracle query generation process for the question in Figure 4.1. Here, besides the process of looking for semantic overlap between the retrieval context and the documents to retrieve, I also illustrate how this process can be used to determine which document is a natural next step in the reasoning chain.

any step of open-domain multi-hop reasoning, there is some *semantic overlap* between the retrieval context and the next document(s) we wish to retrieve. Take the *Armada* question in Figure 4.1 for an example. When the retrieval context contains only the question at the first step of reasoning, this overlap is the novel itself between the question and the Armada novel page. After the retrieval context has been expanded with retrieved documents that contains the novel's page, this overlap becomes the name of the author, *Ernest Cline*, which occurs in both the novel's page and the author's page. In practice, when the exact reasoning path is not given, we simply need to enumerate all of the documents that have yet to be retrieved to obtain a "topological sort" of the chain of reasoning. In the *Armada* example, this is manifested in the fact that we will not be able to find any meaningful overlap that helps retrieve *Ernest Cline* at the first step of reasoning (see Figure 4.3 for an illustrated example). Finding this semantic overlap between the retrieval context and the desired documents not only reveals the chain of reasoning naturally, but also allows us to use it as the search query for retrieval.

Because off-the-shelf IR systems generally optimize for shallow lexical similarity between query and candidate documents in favor of efficiency, a good proxy for this overlap is locating spans of text that have high lexical overlap with the intended supporting documents. To this end, I propose a simple yet effective solution, employing several heuristics to generate candidate queries: computing the longest common string/sequence between the current retrieval context and the title/text of the intended paragraph ignoring stop words, then taking the contiguous span of text that corresponds to this overlap in the retrieval context. This allows us to not only make use of entity names, but

Question	Hop 1 Oracle	Hop 2 Oracle
What government position was held by the woman who portrayed Corliss Archer in the film Kiss and Tell?	Corliss Archer in the film Kiss and Tell	Shirley Temple
Scott Parkin has been a vocal critic of Exxonmobil and another corporation that has operations in how many countries?	Scott Parkin	Halliburton
Are Giuseppe Verdi and Ambroise Thomas both Opera composers?	Giuseppe Verdi	Ambroise Thomas

Table 4.1: Example oracle queries on the HotpotQA dev set.

also textual descriptions that better lead to the gold entities. It is also more generally applicable than question decomposition approaches (Talmor and Berant, 2018; Min et al., 2019b), and does not require additional annotation for decomposition.

Applying various heuristics results in a handful of candidate queries for each document, and I use the IR engine (detailed next) to rank them based on recall of the intended supporting document to choose one as the final oracle query I train my query generators to predict. This allows us to train the query generators in a fully supervised manner efficiently. Some examples of oracle queries on the HotpotQA dev set can be found in Table 4.1. I refer the reader to Section 4.3.6 for more technical details about my heuristics and how the oracle queries are derived.

Note that despite the fact that I assume an order of logical discovery $(d_1, \ldots, d_S)$ in the description of this section, the method I propose actually does not require this information to be available. Based on the same insight about strong semantic overlaps, we should be able to determine, from a collection of unordered gold supporting documents, which ones can be plausibly retrieved given the knowledge in the documents already retrieved. The only computational price one has to pay during training is exhausting the $O(S)$ candidates at each step of reasoning for $O(S)$ steps, which results in a "topological sort" of the gold paragraphs in polynomial time of the number of reasoning steps, rather than the exponential time complexity that would be required if we were to enumerate all possible query candidates and reasoning paths in an brute-force attempt.

Oracle Query vs Single-hop Query I evaluate the oracle query against the single-hop query, i.e., querying with the original question, on the HotpotQA dev set. Specifically, I compare the recall of gold paragraphs, because the greater the recall, the fewer documents I need to pass into the expensive neural multi-hop QA component.

I index the English Wikipedia dump with introductory paragraphs provided by the HotpotQA authors[6] with Elasticsearch 6.7 (Gormley and Tong, 2015), where I index the titles and document text in separate fields with bigram indexing enabled. This results in an index with 5,233,329 total

[6]`https://hotpotqa.github.io/wiki-readme.html`

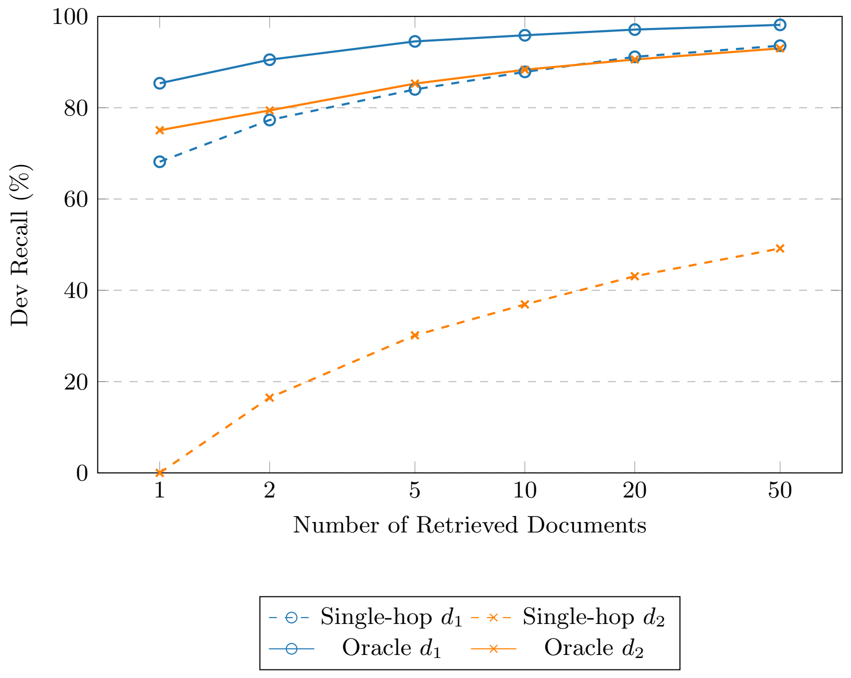


Figure 4.4: Recall comparison between single-hop queries and GoldEn Retriever oracle queries for both supporting paragraphs on the HotpotQA dev set. Note that the oracle queries are much more effective than the original question (single-hop query) at retrieving target paragraphs in both hops.

documents. At retrieval time, I boost the scores of any search result whose title matches the search query better – this results in a better recall for entities with common names (*e.g.*, “*Armada*” the novel). For more details about how the IR engine is set up and the effect of score boosting, please refer to Section 4.4.2.

In Figure 4.4, I compare the recall of the two gold paragraphs required for each question in HotpotQA at various number of documents retrieved (R@n) for the single-hop query and the multi-hop queries generated from the oracle. Note that the oracle queries are much more effective at retrieving the gold paragraphs than the original question in both hops. For instance, if I combine R@5 of both oracles (which effectively retrieves 10 documents from two queries) and compare that to R@10 for the single-hop query, the oracle queries improve recall for d_1 by 6.68%, and that for d_2 by a significant margin of 49.09%.[7] This means that the final QA model will need to consider far fewer documents to arrive at a decent set of supporting facts that lead to the answer.

[7]Since HotpotQA does not provide the logical order its gold entities should be discovered, I simply call the document d_1 which is more easily retrievable with the queries, and the other as d_2.

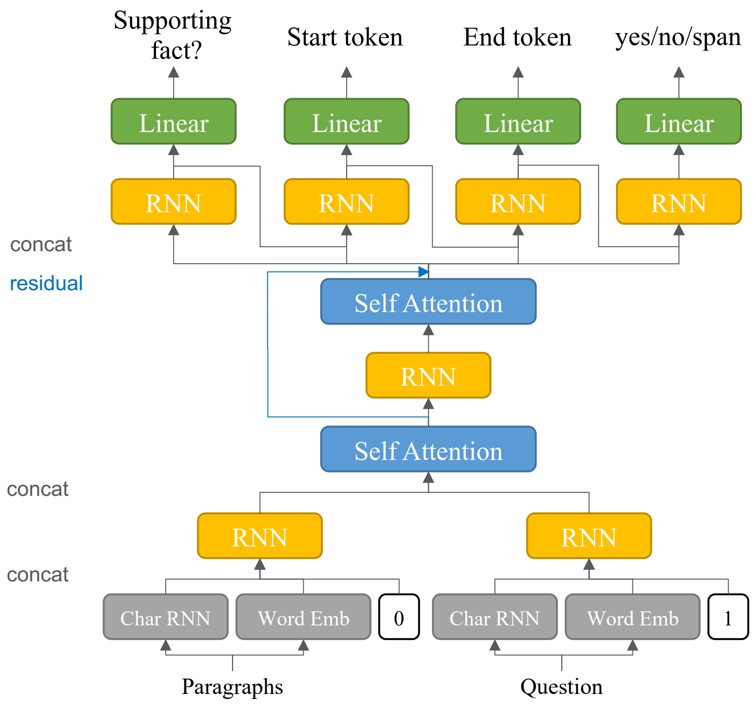


Figure 4.5: Question answering component in GoldEn Retriever. (Best viewed in color)

4.3.5 Question Answering Component

The final QA component of GoldEn Retriever is based on the baseline model presented in (Yang et al., 2018a), which is in turn based on BiDAF++ (Clark and Gardner, 2018). I make two major changes to this model. Yang et al. (2018a) concatenated all context paragraphs into one long string to predict span begin and end offsets for the answer, which is potentially sensitive to the order in which these paragraphs are presented to the model. I instead process them separately with shared encoder RNN parameters to obtain paragraph order-insensitive representations for each paragraph. Span offset scores are predicted from each paragraph independently before finally aggregated and normalized with a global softmax operation to produce probabilities over spans. The second change is that I replace all attention mechanisms in the original model with self attention layers over the concatenated question and context. To differentiate context paragraph representations from question representations in this self-attention mechanism, I indicate question and context tokens by concatenating a 0/1 feature at the input layer. Figure 4.5 illustrates the QA model architecture.

4.3.6 Heuristics for Oracle Query Generation

I mainly employ three heuristics to find the semantic overlap between the retrieval context and the desired documents: longest common subsequence (LCS), longest common substring (LCSubStr), and overlap merging which generalizes the two. Specifically, the overlap merging heuristic looks for contiguous spans in the retrieval context that have high rates of overlapping tokens with the desired document, determined by the total number of overlapping tokens divided by the total number of tokens considered in the span.

In all heuristics, I ignore stop words and lowercase the rest in computing the spans to capture more meaningful overlaps, and finally take the span in the retrieval context that all the overlapping words are contained in. For instance, if the retrieval context contains "*the* GoldEn *Retriever model on* HotpotQA" and the desired document contains "GoldEn *Retriever on the* HotpotQA *dataset*", I will identify the overlapping terms as "GoldEn", "*Retriever*", and "HotpotQA", and return the span "GoldEn *Retriever model on* HotpotQA" as the resulting candidate query.

To generate candidates for the oracle query, I apply the heuristics between combinations of {cleaned question, cleaned question without punctuation} × {cleaned document title, cleaned paragraph}, where cleaning entails stop word removal and lowercasing.

This process results in many oracle query candidates, from which I select only one to train the query generators with. To gauge which oracle query candidate is going to be more effective during retrieval and help the query generator learn to generate effective queries without having to experiment against a search engine, I launch these queries against Elasticsearch to determine the rank of the desired paragraph. If multiple candidate queries are able to place the desired paragraph in the top 5 results, I further rank the candidate queries by other metrics (*e.g.*, length of the query) to arrive at the final oracle query to train the query generators.

4.4 Experiments

4.4.1 Data and Setup

I evaluate my models in the fullwiki setting of HotpotQA (Yang et al., 2018a). As I have introduced in the previous chapter, HotpotQA is a question answering dataset collected on the English Wikipedia, containing about 113k crowd-sourced questions that are constructed to require the introduction paragraphs of two Wikipedia articles to answer. Each question in the dataset comes with the two gold paragraphs, as well as a list of sentences in these paragraphs that crowdworkers identify as supporting facts necessary to answer the question. A diverse range of reasoning strategies are featured in HotpotQA, including questions involving missing entities in the question (our *Armada* example), intersection questions (*What satisfies property A and property B?*), and comparison questions, where two entities are compared by a common attribute, among others. In the few-document *distractor* setting, the QA models are given ten paragraphs in which the gold paragraphs are guaranteed to be found; in the open-domain *fullwiki* setting, which I focus on in this chapter, the models are only given the question and the entire Wikipedia. Models are evaluated on their answer accuracy and explainability, where the former is measured as overlap between the predicted and gold answers with exact match (EM) and unigram F_1, and the latter concerns how well the predicted supporting fact sentences match human annotation (Supporting Fact EM/F_1). A joint metric is also reported on this dataset, which encourages systems to perform well on both tasks simultaneously.

I use the Stanford CoreNLP toolkit (Manning et al., 2014) to preprocess Wikipedia, as well as to generate POS/NER features for the query generators, following (Yang et al., 2018a) and (Chen et al., 2017). I always detokenize a generated search query before sending it to Elasticsearch, which has its own preprocessing pipeline. Since all questions in HotpotQA require exactly two supporting documents, I fix the number of retrieval steps of GoldEn Retriever to $S = 2$. During training and evaluation, I set the number of retrieved documents added to the retrieval context to 5 for each retrieval step, so that the total number of paragraphs our final QA model considers is 10, for a fair comparison to (Yang et al., 2018a).

Before diving into details, I will briefly review hyperparameter and training details in the subsections that follow, for reproducibility.

4.4.2 Elasticsearch Setup

Setting Up the Index

I start from the Wikipedia dump file containing the introductory paragraphs used in HotpotQA that Yang et al. (2018a) provide,[8] and add the fields corresponding to Wikipedia page titles and the introductory paragraphs (text) into the index.

For the title, I use Elasticsearch's `simple` analyzer which performs basic tokenization and lowercasing of the content. For the text, I join all the sentences and use the `standard` analyzer which further allows for removal of punctuation and stop words. For both fields, I index an auxiliary field with bigrams using the `shingle` filter,[9] and perform basic `asciifolding` to map non ASCII characters to a similar ASCII character (*e.g.*, "é"→ "e").

At search time, I launch a `multi_match` query against all fields with the same query, which performs a full-text query employing the BM25 ranking function (Robertson et al., 1995) with default parameters ($k_1 = 1.2, b = 0.75$) against all fields in the index, and returns the score of the best field for ranking by default. To promote documents whose title match the search query, I boost the search score of all title-related fields by 1.25 in this query.

Reranking Search Results

In Wikipedia, it is common that pages or entity names share rare words that are important to search engines, and a naive full-text search IR system will not be able to pick the one that matches the query the best. For instance, if one set up Elasticsearch according to the instructions above and searched for "*George W. Bush*", one would be surprised to see that the actual page is not even in

[8] `https://hotpotqa.github.io/wiki-readme.html`
[9] `https://www.elastic.co/guide/en/elasticsearch/reference/6.7/analysis-shingle-tokenfilter.html`

IR System	R@10 for d_1	R@10 for d_2
Final system	87.85	36.91
w/o Title Boosting	86.85	32.64
w/o Reranking	86.32	34.77
w/o Both	84.67	29.55

Table 4.2: IR performance (recall in percentages) of various Elasticsearch setups on the HotpotQA dev set using the original question.

the top-10 search results, which contains entities such as "*George W. Bush Childhood Home*" and "*Bibliography of George W. Bush*".

To this end, I propose to rerank these query results with a simple but effective heuristic that alleviates this issue. I would first retrieve at least 50 candidate documents for each query for consideration, and boost the query scores of documents whose title exactly matches the search query, or is a substring of the search query. Specifically, I multiply the document score by a heuristic constant between 1.05 and 1.5, depending on how well the document title matches the query, before reranking all search results. This results in a significant improvement in these cases. For the query "*George W. Bush*", the page for the former US president is ranked at the top after reranking. In Table 4.2, I also provide results from the single-hop query to show the improvement from title score boosting introduced from the previous section and reranking.

4.4.3 Training Details

Query Generators

Once the oracle queries are generated, I train our query generators to emulate them on the training set, and choose the best model on the dev set in terms of F_1 in selecting the oracle query span. I experiment with hyperparameters such as learning rate, training epochs, batch size, number of word embeddings to finetune, among others, and report the final hyperparameters for both query generators (G_1 and G_2) in Table 4.3.

Question Answering Model

The final question answering component is trained with the paragraphs produced by the oracle queries (5 from each hop, 10 in total), with d_1 and d_2 inserted to replace the lowest ranking paragraph in each hop if they are not in the set already.

I develop the model based on the baseline model of Yang et al. (2018a), and reuse the same default hyperparameters whenever possible. The main differences in the hyperparameters are: I optimize the QA model with Adam (with default hyperparameters) (Kingma and Ba, 2015) instead of stochastic gradient descent with a larger batch size of 64; I anneal the learning rate by 0.5 with a

Hyperparameter	Values
Learning rate	*5×10^{-4}*, $\mathbf{1 \times 10^{-3}}$
Finetune embeddings	0, 200, **500**, *1000*
Epoch	**25**, *40*
Batch size	***32***, 64, 128
Hidden size	64, ***128***, 256, 512, 768
Max sequence length	15, *20*, 30, **50**, 100
Dropout rate	0.3, 0.35, ***0.4***, 0.45

Table 4.3: Hyperparameter settings for the query generators. The final hyperparameters for the Hop 1 query generator are shown in **bold**, and those for the Hop 2 query generator are shown in *underlined italic*.

patience of 3 instead of 1, that is, I multiply the learning rate by 0.5 after three consecutive failures to improve dev F_1; I clip the gradient down to a maximum ℓ_2 norm of 5; I apply a 10% dropout to the model, for which I have increased the hidden size to 128; and use 10 as the coefficient by which I multiply the supporting facts loss, before mixing it with the span prediction loss. I configure the model to read 10 context paragraphs, and limit each paragraph to at most 400 tokens including the title.

4.5 Results

In this section, I cover the main results from the experiments with GoldEn Retriever on the HotpotQA dataset. I will begin by focusing on the end-to-end performance of the system on the question answering task, before moving on to analyzing the performance of the query generation models.

4.5.1 End-to-end Question Answering

I compare the end-to-end performance of GoldEn Retriever against several QA systems on the HotpotQA dataset: (1) the baseline presented in (Yang et al., 2018a), (2) CogQA (Ding et al., 2019), the top-performing previously published system, and (3) other high-ranking systems on the leaderboard. As shown in Table 4.4, GoldEn Retriever is much better at locating the correct supporting facts from Wikipedia compared to CogQA, as well as most of the top-ranking systems. However, the QA performance is handicapped because I do not make use of large pretrained contextualization models such as BERT (Devlin et al., 2019) that these systems use. More recent question answering systems have benefitted greatly from the adoption of these large pretrained language models to contextualize word meanings, which also helps them better find fuzzy matching patterns to predict the correct answer span. I expect a boost in QA performance from adopting these more powerful question answering models, especially ones that are tailored to perform few-document

System	**Answer**		**Sup Fact**		**Joint**	
	EM	F_1	EM	F_1	EM	F_1
Baseline (Yang et al., 2018a)	25.23	34.40	5.07	40.69	2.63	17.85
GRN + BERT	29.87	39.14	13.16	49.67	8.26	25.84
MUPPET (Feldman and El-Yaniv, 2019)	30.61	40.26	16.65	47.33	10.85	27.01
CogQA (Ding et al., 2019)	37.12	48.87	22.82	57.69	12.42	34.92
PR-Bert	43.33	53.79	21.90	59.63	14.50	39.11
Entity-centric BERT Pipeline	41.82	53.09	26.26	57.29	17.01	39.18
SemanticRetrievalMRS (Nie et al., 2019)	45.32	57.34	38.67	70.83	25.14	47.60
GoldEn Retriever	37.92	48.58	30.69	64.24	18.04	39.13

Table 4.4: End-to-end QA performance of baselines and the GoldEn Retriever model on the HotpotQA fullwiki test set. Among systems that were not published at the time of submission of this paper, "SemanticRetrievalMRS" was submitted to the official HotpotQA leaderboard on May 15th (thus contemporaneous with GoldEn Retriever), while "Entity-centric BERT Pipeline" and "PR-Bert" were submitted after the original GoldEn Retriever paper was submitted.

multi-hop reasoning.

To understand the contribution of GoldEn Retriever's iterative retrieval process compared to that of the IR engine, I compare the performance of GoldEn Retriever against two baseline systems on the dev set: one that retrieves 10 supporting paragraphs from Elasticsearch with the original question, and one that uses the IR engine presented in HotpotQA to retrieve 10 supporting paragraphs. For the latter, I use the fullwiki test input file orignally released with the dataset, which contains the top-10 IR output from that retrieval system with the question as the query. In all cases, I use the QA component in GoldEn Retriever for the final question answering step. As shown in Table 4.5, replacing the hand-engineered IR engine in (Yang et al., 2018a) with Elasticsearch does result in some gains in recall of the gold documents, but that does not translate to a significant improvement in QA performance. Further inspection reveals that despite Elasticsearch improving overall recall of gold documents, it is only able to retrieve both gold documents for 36.91% of the dev set questions, in comparison to 28.21% from the IR engine in (Yang et al., 2018a). In contrast, GoldEn Retriever improves this percentage to 61.01%, almost doubling the recall over the single-hop baseline, providing the QA component a much better set of context documents to predict answers from.

Lastly, I perform an ablation study in which I replace the proposed query generator models with the query oracles I derived and observe the effect on end-to-end performance. As can be seen in Table 4.6, replacing G_1 with the oracle only slightly improves end-to-end performance, but further substituting G_2 with the oracle yields a significant improvement. This illustrates that the performance loss is largely attributed to G_2 rather than G_1, because G_2 solves a harder span selection problem from a longer retrieval context. In the next section, I examine the query generation models more closely by evaluating their performance without the QA component.

Setting	**Ans F$_1$**	**Sup F$_1$**	**R@10***
GoldEn Retriever	49.79	64.58	75.46
Single-hop query	38.19	54.82	62.38
HotpotQA IR	36.34	46.78	55.71

Table 4.5: Question answering and IR performance amongst different IR settings on the dev set. I observe that although improving the IR engine is helpful, most of the performance gain results from the iterative retrieve-and-read strategy of GoldEn Retriever. (*: for GoldEn Retriever, the 10 paragraphs are combined from both hops, 5 from each hop.)

System	**Ans F$_1$**	**Sup F$_1$**	**Joint F$_1$**
GoldEn Retriever	49.79	64.58	40.21
w/ Hop 1 oracle	52.53	68.06	42.68
w/ Hop 1 & 2 oracles	62.32	77.00	52.18

Table 4.6: Pipeline ablative analysis of GoldEn Retriever end-to-end QA performance by replacing each query generator with a query oracle.

4.5.2 Analysis of Query Generation

To evaluate the query generators, I begin by determining how well they emulate the oracles. I evaluate these models using Exact Match (EM) and F$_1$ on the span prediction task, as well as compare their queries' retrieval performance against the oracle queries. As can be seen in Table 4.7, the performance of G_2 is worse than that of G_1 in general, confirming my findings on the end-to-end pipeline. The retrieval performance of Hop 1 generated queries outperforms single-hop R@5 by about 2% (the latter is 84.01%), which is presumably because the predicted queries removed noise from the question. For the Hop 2 query generator, I also find the oracle queries noisier for its training, which exacerbates the deterioration in its performance.

When I combine these query generators into a pipeline, the generated queries perform only slightly better on d_1 when a total of 10 documents are retrieved (89.91% vs 87.85%), but are significantly more effective for d_2 (61.01% vs 36.91%). If we further zoom in on the retrieval performance on non-comparison questions for which finding the two entities involved is less trivial, we can see that the recall on d_2 improves from 27.88% to 53.23%, almost doubling the number of questions

Model	**Span**		**R@5**
	EM	F$_1$	
G_1	51.40	78.75	85.86
G_2	52.29	63.07	64.83

Table 4.7: Span prediction and IR performance of the query generator models for Hop 1 (G_1) and Hop 2 (G_2) evaluated separately on the HotpotQA dev set.

	Question	Predicted q_1	Predicted q_2
(1)	What video game character did the voice actress in the animated film Alpha and Omega voice?	voice actress in the animated film Alpha and Omega *(animated film Alpha and Omega voice)*	Hayden Panettiere
(2)	What song was created by the group consisting of Jeffrey Jey, Maurizio Lobina and Gabry Ponte and released on 15 January 1999?	Jeffrey Jey *(group consisting of Jeffrey Jey, Maurizio Lobina and Gabry Ponte)*	Gabry Ponte and released on 15 January 1999 *("Blue (Da Ba Dee)")*
(3)	Yau Ma Tei North is a district of a city with how many citizens?	Yau Ma Tei North	Yau Tsim Mong District of Hong Kong *(Hong Kong)*
(4)	What company started the urban complex development that included the highrise building, The Harmon?	highrise building, The Harmon	CityCenter

Table 4.8: Examples of predicted queries from the query generators on the HotpotQA dev set. The oracle query is displayed in blue in parentheses if it differs from the predicted one.

for which we have the complete gold context to answer. I note that the IR performance I report on the full pipeline is different to that when I evaluate the query generators separately. I attribute this difference to the fact that the generated queries sometimes retrieve both gold documents in one step.

To better understand model behavior, I also randomly sampled some examples from the dev set to compare the oracle queries and the predicted queries. Aside from exact matches, I find that the predicted queries are usually small variations of the oracle ones. In some cases, the model selects spans that are more natural and informative (Example (1) in Table 4.8). When the predicted query differ a bit more from the oracle query, the model is usually overly biased towards shorter entity spans and misses out on informative information (Example (2)). When there are multiple entities in the retrieval context, the model sometimes selects the wrong entity, which suggests that a more powerful query generator might be desirable (Example (3)). Despite these issues, we can see that these natural language queries make the reasoning process more explainable, and easier for a human to verify or intervene as needed.

Limitations Although I have demonstrated that generating search queries with span selection works in most cases, it also limits the kinds of queries we can generate, and in some cases leads to undesired behavior. One common issue is that the entity of interest has a name shared by too many Wikipedia pages (*e.g.*, "*House Rules*" the 2003 TV series). This sometimes results in the inclusion of extra terms in the oracle query to expand it (*e.g.*, Example (4) specifies that "*The Harmon*" is a highrise building). I argue, though, these are due to the simplifying choice of span selection for query generation and fixed number of query steps, rather than the inherit limitation of the overall approach I take in this chapter.

In some cases, the span oracle makes use of too much information from the gold entities and reveals information from a list of entities that would have otherwise been opaque unless one knew the answer already (Example (2), where a human would likely query for "*Eiffel 65 song released 15 January 1999*" because "*Blue*" is not the only song mentioned in d_1). This problem will likely be resolved if counterfactual reasoning were designed as part of the query generation process of the oracle.

Outside of HotpotQA, when the methodology of GoldEn Retriever is applied to less entity-centric multi-step question answering tasks like multi-step common sense question answering (*e.g.*, WorldTree (Jansen et al., 2018)), the problem of query generation and information retrieval may be more challenging. This is because these questions tend to revolve around words and entities that are more common in the text collection, which is not what typical information retrieval systems are good at. The graph approaches I have mentioned in Section 4.6 or reranking approaches that can efficiently process a larger amount of search results will likely be helpful here, together with QA systems that are capable of exploring more than one reasoning paths to answer a question.

Aside from the retrieval component, the GoldEn Retriever system presented in this system is also limited by the capabilities of the span-based reading comprehension model it uses in the types of multi-hop questions it can answer. Two salient examples of practically useful multi-hop questions that such reading comprehension systems cannot accommodate are aggregation questions, where the answer requires arithmetic computations given the facts stated in the text, and enumeration questions, where the answer is a list of entities or text spans that are potentially disjoint or in different documents.

4.6 Conclusion

In this chapter, I presented GoldEn (Gold Entity) Retriever, an open-domain multi-hop question answering system for scalable multi-hop reasoning. Through iterative reasoning and retrieval, GoldEn Retriever greatly improves the recall of gold supporting facts, thus providing the question answering model a much better set of context documents to produce an answer from, and demonstrates competitive performance to the state of the art on HotpotQA. Designed to generate natural languages queries for each step of reasoning, GoldEn Retriever is more explainable to humans compared to previous neural retrieval approaches and affords better understanding and verification of model behavior. This explainable design of the model not only allows it to make use of a computationally efficient information retrieval system, but also serves as an important motivation for how we can derive supervision signal from the limited training data available and train the various components in the system to work together. As a result, this formulation allows NLP systems to help us gain a deeper access to textual knowledge on-demand, with almost no additional cost to preprocess or store data once the knowledge is available in textual form.

GoldEn Retriever's model design also has several advantages when compared to previous and more recent work in open-domain QA that leverages trained neural network retrieval systems (Das et al., 2019; Lee et al., 2019; Feldman and El-Yaniv, 2019; Dhingra et al., 2020; Karpukhin et al., 2020) that I would like to highlight. I have already mentioned that leveraging text-based retrieval systems makes the entire system more explainable and controllable to a user, which is important. Moreover, it is also more adaptive to any generic-purpose off-the-shelf retrieval system. On the one hand, this means that whenever a better text-based retrieval system is made available, it is much easier for GoldEn retriever to switch to it with minimal amount of training. The search queries and results are also highly transparent to human intervention and progress monitoring. On the other hand, this also has the advantage of not requiring task-specific training or finetuning of a retrieval system at a large scale, which may not later generalize to unseen scenarios, or not be as amenable to updating with new documents that describe new textual knowledge.

Up until this point in this dissertation, I have focused on building NLP systems that are capable of complex reasoning over textual knowledge to help us answer questions. These systems expand our capabilities of making use of textual knowledge by freeing us from having to curate the relevant facts manually and/or performing reasoning and aggregation on our own. However, these systems are based on a crucial underlying assumption, which is that the user is able to express their information need in a single question.

This might not always be the case in reality. Aside from our conscious adaptation to the capabilities of our tools such as search engines, where we decompose our questions into simpler requests, there are also commonly cases that we cannot determine what the right question is to arrive at the answer we want. In these cases, it is important that an NLP system either learns to infer our intent by simulating the behavior of someone with an information need, or ask meaningful clarification questions to help us narrow the search space. Both would require the system to be able to ask questions, to which the answer is not already presented in a readily available context of text, unlike in the typical reading comprehension settings that I have introduced this far.

In the next chapter, I will present how we can build NLP systems that ask meaningful questions to gather information that is not readily presented, by defining the notion of informativeness of questions and optimizing it.

Chapter 5

Reasoning about Informativeness in Conversations

In previous chapters, I have presented various approaches that enable NLP systems to help us solve problems with textual knowledge, which involve the systems unpacking our complex request into multiple steps of reasoning, and gathering relevant information to provide an answer to it. However, this still leaves some information needs underserved, which might still have their answers in the vast amount of knowledge available to us in textual form. Specifically, we might often find ourselves in a situation where we do not have all the necessary information to articulate our information need, and would therefore need to iterate and collect more information until we can actually ask the right question. One example is a student learning the methodology from a knowledgeable teacher while establishing a common background; another is debugging complex issues in programming. A more everyday example might be the purchase of a camera – one would often have to learn about what features are important in the decision-making process before finally figuring out what one's criteria and needs are.

In these scenarios, it is highly desirable that our NLP systems be interactive, and able to actively reason about our underlying intent to accommodate our information needs. To achieve this goal, two kinds of skills are important: asking questions to directly gather information from us about what our underlying intent might be, and reasoning about why we asked the questions we did in an interactive setting to infer what our intent might have been through simulation. Both of these skills revolve around a common theme, *i.e.*, asking questions in an interactive setting (*e.g.*, a conversation) about something that one might not already know the answer to, or even what the answer might be. Besides acquiring information, the capability of asking questions without knowing the answer and or accessing the informational context can also help NLP systems forward simulate what follow-up questions a user might ask given different answers to previous questions in the conversation. This can potentially help NLP systems reason about the effectiveness of different answer candidates, and optimize communciation efficiency in serving our information needs.

In this chapter, I study the problem of question asking (or question generation) in a conversational setting, to help shed light on these important issues. Specifically, I focus on conversational settings where there is information asymmetry, as well as a need/pressure to communicate this information through conversation. Moreover, I study information needs that are complex, diverse, and naturally occurring, as opposed to more "scripted" and "form-filling" ones (*e.g.*, the ones in ATIS (Hemphill et al., 1993), which concerns booking air travel from various cities to different destinations) which are much better understood.[1]

I study the problem of generating meaningful, inquisitive questions in an information-seeking conversation between two agents, where one agent possesses knowledge that is encoded in textual format but is limited to sharing it with the other. For simplicity, I will refer to the more knowledgeable agent as the teacher, and to the other as the student. The goal of the student is to ask relevant questions that elicit answers from the teacher to enrich its own knowledge. I focus on an open-domain and open-ended setting, by which I mean that the topic the agents are allowed to talk about are not predefined by any specific knowledge schema ahead of time, and by the latter I mean the task is purely *curiosity-driven* for the student—rather than optimizing for any specific end goal, the main driving force for communication is gathering more knowledge about a specific topic. This is a challenging topic that has rarely been explored in previous work.

I will first review the background and related work around question generation, show how previous approaches largely fall short at solving the problem we have at hand, and present my approach towards the problem by defining the informativeness of inquisitive questions and optimizing it. When this informativeness measure is combined with a system that assesses these questions for relevance and specificity, the joint supervision signal trains a system that generates questions in open-domain conversations, which are not just interesting but also acquire new knowledge from a more knowledgeable agent.

5.1 Background

Conversations are a primary means to seek and communicate information between humans, where asking the right question is an important skill and prerequisite for effective exchange of knowledge (Browne and Keeley, 2014). Learning to ask questions in conversations can help computer systems not only acquire new knowledge, but also engage human interlocutors by making them feel heard (Huang et al., 2017).

[1] Here I use the word "scripted" in the sense as it was introduced by Schank and Abelson (1977). Rather than a comment on whether each individual example has a prepared script to follow at the time of collection, this is more related to the nature of such tasks, where conversations could be said to loosely follow a high-level "script" of what typically takes place after what.

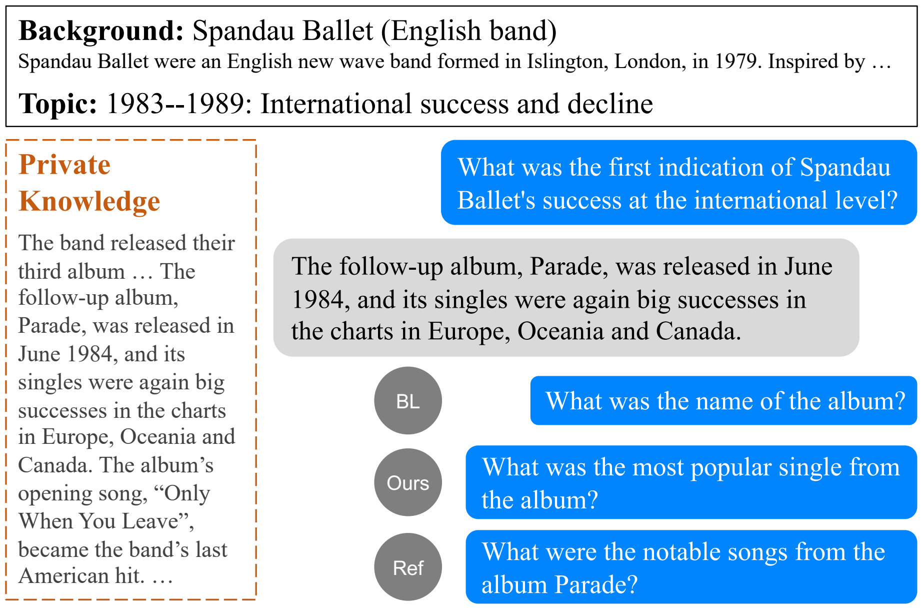

Background: Spandau Ballet (English band)
Spandau Ballet were an English new wave band formed in Islington, London, in 1979. Inspired by …
Topic: 1983--1989: International success and decline
Private Knowledge
The band released their third album … The follow-up album, Parade, was released in June 1984, and its singles were again big successes in the charts in Europe, Oceania and Canada. The album's opening song, "Only When You Leave", became the band's last American hit. …
What was the first indication of Spandau Ballet's success at the international level?
The follow-up album, Parade, was released in June 1984, and its singles were again big successes in the charts in Europe, Oceania and Canada.
BL
What was the name of the album?
Ours
What was the most popular single from the album?
Ref
What were the notable songs from the album Parade?

Figure 5.1: An example for asking questions in a conversation to acquire information. In this communication setting, the question asker has access to the background and topic, but no access to the private textual knowledge that contains the answer. In this example, the baseline non-pragmatic question generator (BL) generates an uninformative question (one that has already been answered), while the pragmatic system I propose (Ours) and humans (Ref) actively seek new information.

Previous work on question generation often falls into three classes: generating questions according to a discrete schema or end goal (Bordes et al., 2017; Zhang et al., 2018b), transforming the answer statement into a question (Mitkov and Ha, 2003; Rus et al., 2010; Heilman and Smith, 2010), or generating questions with data-driven systems by conditioning on the context where the answer comes from (Du et al., 2017; Zhou et al., 2017). Despite their successful adaptation to conversations to predict the question that elicits the observed answer (Gao et al., 2019; Pan et al., 2019; Nakanishi et al., 2019), existing question generation techniques are not suitable for modeling communication of knowledge in open-domain conversations, because the crucial problem of *what to communicate* has already been assumed to be addressed by conditioning on the schema of information need or the context that contains the answer.

I instead study the problem of question generation in a more realistic setting, *i.e.*, in *open-domain information-seeking conversations* where the question asker cannot access the answering context. This is an important step towards practical natural language processing (NLP) systems that can reason about the state of mind of agents they interact with purely through natural language interactions, so that they can generate more helpful responses. In this chapter, I build a question generator that reasons pragmatically about what information the answerer can provide, and generates questions to gather new information in a conversation (see Figure 5.1 for an example).

I identify several key challenges in this task: (1) generating informative questions without access to potential answers; (2) evaluating generated questions beyond comparing them to the reference question, because multiple questions can reveal unseen information despite being very different to each other; (3) navigating a large search space of potential questions to improve informativeness by reasoning about the other agent's knowledge, which is more complex than limited reference games in previous work on computational pragmatics.

To address these issues, I first develop a baseline question generation model that generates questions in a conversation without conditioning on the unseen knowledge. I then propose automatic metrics to quantify how much new information questions reveal, as well as how specific they are to the conversation. Next, I use reinforcement learning to optimize our question generator on these metrics. In experiments on the QuAC dataset (Choi et al., 2018), I show that the proposed method substantially improves the specificity and informativeness of the generated questions as evaluated by our automatic metrics. These results are corroborated by blinded human evaluation, where questions generated by our system are also of higher overall quality than those by the baseline system as judged by humans. To recap, the main contributions I make in this chapter are:

- To the best of my knowledge, this work represents the first attempt at studying question generation to seek information in open-domain communication, which involves challenging NLP problems, *e.g.*, evaluation of open-ended language generation and pragmatic reasoning;

- To address these problems, I propose automatic metrics to quantify the informativeness and specificity of questions. Specifically, the informativeness metric evaluates how much new information can be elicited from the answerer for each potential question, and the specificity metric discourages generic questions that are informative but not interesting (*e.g.*, *"What else?"*). These metrics are essential for efficient iterative system development;

- I show that optimizing the proposed metrics via reinforcement learning leads to a system that behaves pragmatically and has improved communication efficiency, as also verified by human evaluation. This represents a practical method for pragmatic reasoning in an open-domain communication setting.

5.2 Related Work

Question Generation. Question generation has long been studied in the education and psychology communities as a means to assess and promote reading comprehension in humans (Davey and McBride, 1986). In natural language processing, question generation has been explored to improve the systems in various natural language processing tasks, *e.g.*, the quality of question answering systems (Duan et al., 2017) as well as information retrieval in an open-domain question answering system (Nogueira et al., 2019).

Some of the first question generation systems were rule-based (Mitkov and Ha, 2003; Rus et al., 2010; Heilman and Smith, 2010), where syntactic and lexical patterns are used to transform statements into fill-in-the-blank questions and question sentences. More recently, large-scale question answering datasets, *e.g.*, SQuAD (Rajpurkar et al., 2016, 2018), have kindled research interest in data-driven approaches. Du et al. (2017) and Zhou et al. (2017) apply sequence-to-sequence (seq2seq) models to generate SQuAD questions from Wikipedia sentences containing the answers.

The release of large conversational question answering datasets such as QuAC (Choi et al., 2018) and CoQA (Reddy et al., 2019) enabled Gao et al. (2019), Pan et al. (2019) and Nakanishi et al. (2019) to extend previous neural seq2seq question generators by conditioning them on the conversation history and the context that contains the answer, while Scialom and Staiano (2019) remove answers to the reference question to generate curiosity-driven questions from the rest of the context.

Despite their success, most existing approaches to question generation are limited to either *reading comprehension* settings where what can be answered is known *a priori*, or *goal-oriented settings* where the schema of knowledge is limited (Bordes et al., 2017; Zhang et al., 2018b). This prevents them from being applied to an *open-domain communication* setting, where the purpose of questions is to acquire information that is unknown ahead of time. That is, situations in which questions are generated without knowing the answer or seeing the informational text.

Evaluating System-generated Questions. Automatic evaluation of system-generated text has long been an important topic in NLP. Traditional n-gram overlap-based approaches (Papineni et al., 2002; Lin, 2004) are computationally efficient, but have been shown to correlate poorly with human judgement of quality (Novikova et al., 2017). More recently, Zhang et al. (2020) leverage large pretrained language models (BERT, Devlin et al., 2019) to relax the limitation of exact n-gram overlap. Hashimoto et al. (2019) combine human judgement with system-reported likelihood of generated text to make population-level estimates of quality and diversity. However, most existing metrics either evaluate generated text against very few references, or provide only relative ranking for multiple systems at a population level rather than reliable feedback for each example. This renders them inapplicable to generating informative questions in a conversation, where multiple questions can be equally informative and relevant in a given scenario, and per-example feedback is necessary.

Pragmatic Reasoning for Informativeness. Pragmatic reasoning is tightly related to informativeness and efficiency in communication. Starting from the cooperative maxims for conversational pragmatic reasoning (Grice, 1975), Frank and Goodman (2012) developed a computational framework that has been applied to reference games with images (Andreas and Klein, 2016) and colors (Monroe et al., 2017), as well as generating descriptions for images (Cohn-Gordon et al., 2019). Decision-theoretic principles (van Rooy, 2003) have also been applied to quantify the informativeness of community questions (Rao and Daumé III, 2018). These approaches usually assume that either the list of *referents* (images, colors, or answers) or the space of utterances (descriptions or questions) is enumerable or can be directly sampled from, or both. More crucially, the speaker agent usually has complete access to this information to readily gauge the effect of different utterances. I instead study a more realistic information-seeking setting, where the questioner cannot access the answers, let alone aggregate them for pragmatic reasoning, and where these simplifying assumptions will not hold.

5.3 Method

In this section, I outline the problem setup for the communication problem I set out to address, present a baseline system, and lay out our approach to extending it to reason pragmatically to acquire information more efficiently.

5.3.1 Problem Setup

I consider a communication game between two agents, a teacher and a student (see Figure 5.1 for an example). The two agents share a common topic of discussion $\mathcal{T}$ (*Background* and *Topic* in the figure), as well as a common goal for the student to acquire some knowledge $\mathcal{K}$ on this topic that only the teacher has direct access to (*Private Knowledge* in the figure). I consider the scenario where the agents can only communicate to each other by engaging in a conversation, where the conversation history $\mathcal{H}$ is shared between the agents. I further constrain the conversation to one where the student asks questions about the shared topic, and the teacher provides answers based on $\mathcal{K}$. Note that this setup is very similar to that of the "Game of Interrogation" by Groenendijk (1999), except we relax the definition, using natural language instead of focusing on predicate logic, as I will detail in the sections that follow.

Despite these constraints, this *information-seeking conversation* setting is still a good testbed to study various interesting and challenging problems in natural language processing, including quantification of information flow in natural language, and its evaluation, as well as pragmatic reasoning about the other agent's state of mind to ask the right question or provide an informative answer.

In this chapter, I am interested in building a model of the student (question asker) in this scenario. Specifically, I investigate how to enable the student to reason pragmatically about which questions to ask to efficiently acquire knowledge, given only the topic $\mathcal{T}$ and the conversation history $\mathcal{H}$. This setting of *information-seeking conversations* involves many interesting and challenging problems in natural language processing:

- **Quantifying textual information.** We need to be able to quantify how much knowledge the student has acquired from $\mathcal{K}$.

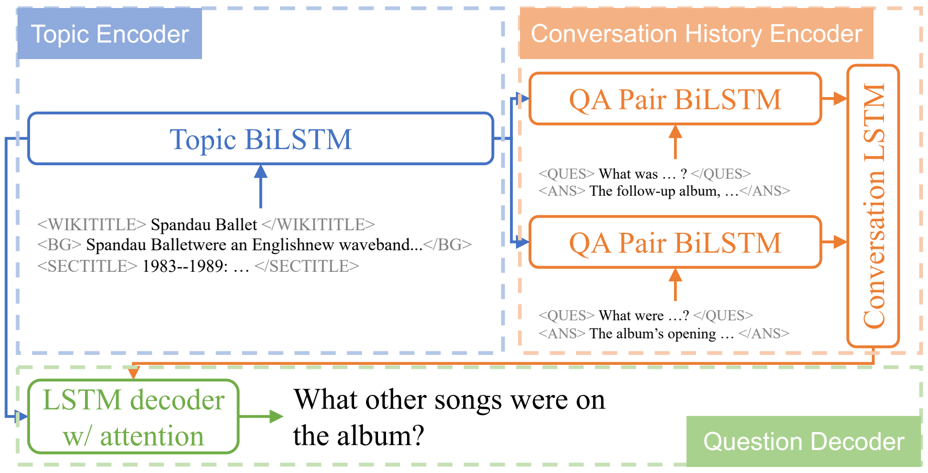

Topic Encoder
Topic BiLSTM
<WIKITITLE> Spandau Ballet </WIKITITLE>
<BG> Spandau Balletwere an Englishnew waveband...</BG>
<SECTITLE> 1983--1989: … </SECTITLE>
Conversation History Encoder
QA Pair BiLSTM
<QUES> What was … ? </QUES>
<ANS> The follow-up album, …</ANS>
QA Pair BiLSTM
<QUES> What were …? </QUES>
<ANS> The album's opening … </ANS>
Conversation LSTM
LSTM decoder w/ attention
What other songs were on the album?
Question Decoder

Figure 5.2: Architecture of the question generation model, which takes only the shared topic and conversation history to generate the next question in the conversation.

- **Evaluating language generation when a single reference is insufficient.** At any state in the conversation, there is usually more than one valid question, some more effective and more appropriate than others. To address this problem, we need to come up with evaluation metrics and objective functions accordingly, rather than relying on the similarity between generated questions and the single reference that is available in existing datasets.

- **Pragmatic reasoning with partial information and a large search space.** In order to train computational agents capable of pragmatic reasoning, previous work typically takes the approach of either limiting the space of referents, or the space of possible utterances, or both. However, the former is infeasible in a communication setting as the student doesn't have access to $\mathcal{K}$ beyond what is already revealed in the conversation, and the latter is also impractical for natural conversations that cover a diverse set of topics.

I address these challenges by proposing two automatic reward functions that evaluate the informativeness and specificity of questions, and optimizing them with reinforcement learning.

5.3.2 Generating Questions in Conversations

Before we delve into the proposed approaches for training a question generator model to be pragmatic, an introduction of the model itself is due.

For the purposes of this chapter, I assume that the shared topic $\mathcal{T}$, the shared conversation history $\mathcal{H}$, and the teacher's knowledge $\mathcal{K}$ (which the student has no access to) are all made available to agents in natural language. Since we consider information-seeking conversations only, the conversation history is grouped into pairs of questions and answers: $\mathcal{H} = [(q_1, a_1), (q_2, a_2), \ldots, (q_{|\mathcal{H}|}, a_{|\mathcal{H}|})]$.

Student: Who discovered the theory of relativity?

Teacher: Albert Einstein

Student: What is it about?

Teacher: Special relativity applies to all physical phenomena in the absence of gravity. General relativity explains the law of gravitation and its relation to other forces of nature.

Student: What is the main contribution of special relativity?

Teacher: Special relativity is a theory of the structure of spacetime.

Figure 5.3: An example of a communication game where a teacher teaches a student about the theory of relativity. Here we use the Wikipedia page on the theory of relativity as a source of knowledge for the teacher.

To generate conversational questions, I build a sequence-to-sequence model that encodes the information available to the student and decodes it into the next question in the conversation (see Figure 5.2). Specifically, I first model the shared topic $\mathcal{T}$ with a bi-directional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997), and use the resulting topic representation $h_{\mathcal{T}}$ in the conversation encoder. Then I obtain a representation of the conversation with hierarchical LSTM encoders: I first encode each pair of question and answer with $h_{\mathcal{T}}$ using a BiLSTM, then feed these pair representations into a *unidirectional* LSTM in the direction that the conversation unfolds. To generate the question, I apply an LSTM decoder with attention both on the topic and the conversation history (Bahdanau et al., 2015). This allows us to efficiently batch computation for each conversation by sharing these representations across different turns. I include detailed description of the model in Section 5.3.6.

As a baseline, I train this model to minimize the negative log likelihood (NLL) of questions observed in the training set:

$$\ell_{\text{NLL}} = -\frac{1}{N_{\text{p}}}\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{H}^{(i)}|}\log P_\theta(q_j^{(i)}|\mathcal{H}_{<j}^{(i)}, \mathcal{T}), \tag{5.1}$$

where θ stands for model parameters, N is the number total conversations in the training dataset, $\mathcal{H}^{(i)}$ the conversation history of the i-th conversation in the dataset, and $N_{\text{p}} = \sum_{i=1}^{N}|\mathcal{H}^{(i)}|$ is the total number of question-answer pairs in the training dataset. Intuitively, this trains the model to mimic the observed questions in the dataset, but does not provide guarantees or assessment of how well generated questions are actually able to acquire information from the teacher agent.

5.3.3 Evaluating Informativeness through Question Answering

In order to train the question generation model to generate pragmatically apt questions that reveal new information from $\mathcal{K}$, we need to be able to quantify informativeness in communication first.

However, informativeness is difficult to quantify in an open-domain dialogue, and sometimes even subjective. Whether a question is informative to the asker depends a lot on the asker's background knowledge, subjective experience, and common ground between the agents. For instance, in the example in Figure 5.3, *Who discovered the theory of relativity?* is informative for someone who is new to this concept, but probably much less so to a PhD student in physics.

In this chapter, I sidestep the issues of subjectivity and prior experiences, and focus on providing an objective metric for how much new information is revealed by a question. Since questions do not reveal information directly, but rather rely on the answers to them to introduce new facts into the conversation, I begin by defining the *informativeness* of an answer a once it is provided. Specifically, I am interested in characterizing how much new information an answer a reveals about $\mathcal{K}$ beyond what is already provided in the conversation history $\mathcal{H}_{<j}$ up until this point in the conversation. Theoretical quantities like mutual information might seem appealing in this context given their strong grounding in information theory. However, applying them would potentially require us to fully specify the state space the world can be in for an open-domain conversation, as well as estimating the probability distribution over potential configurations, neither of which is trivial, if feasible. Therefore, I turn to more practical quantities in defining the informativeness of an answer a given the conversation history $\mathcal{H}_{<j}$ by leveraging the observation that, the more new information an answer reveals about $\mathcal{K}$, the more likely it involves words that have not already been mentioned in $\mathcal{H}_{<j}$. Therefore, making use of the unigram precision function $\mathrm{Prec}(a, a')$ between the predicted answer a and an answer a' that is already provided in the conversation history $\mathcal{H}_{<j}$, I define the informativeness of the predicted answer as follows

$$I_{\mathrm{ans}}(a; \mathcal{H}_{<j}) := 1 - \max_{1 \le k < j} \mathrm{Prec}(a, a_k), \tag{5.2}$$

Intuitively, the more a overlaps with *any* of the previously revealed answers, the less new information it contains. This metric of informativeness has the advantages of objectivity and ease of automatic evaluation. Also note that the choice of unigram precision is here not one of necessity but simplicity and practicality. It is in principle interchangeable with more sophisticated models of fuzzy text overlap (*e.g.*, BERTScore (Zhang et al., 2020)).

I use this definition of answer informativeness to define the utility of potential questions. Specifically, I define the *informativeness* of a question as the amount of new information it can immediately reveal through its answer

$$I(q; \mathcal{C}_{<j}) := I_{\mathrm{ans}}(\mathrm{QA}(q, \mathcal{C}_{<j}), \mathcal{H}_{<j}), \tag{5.3}$$

where $\mathcal{C}_{<j} = (\mathcal{H}_{<j}, \mathcal{T}, \mathcal{K})$ is the complete context available to the teacher up until the question is raised, $\mathrm{QA}(q, \mathcal{C}_{<j})$ is a pretrained conversational question answering (QA) model that answers the question q from the knowledge source $\mathcal{K}$ given this context. This is equivalent to using a point

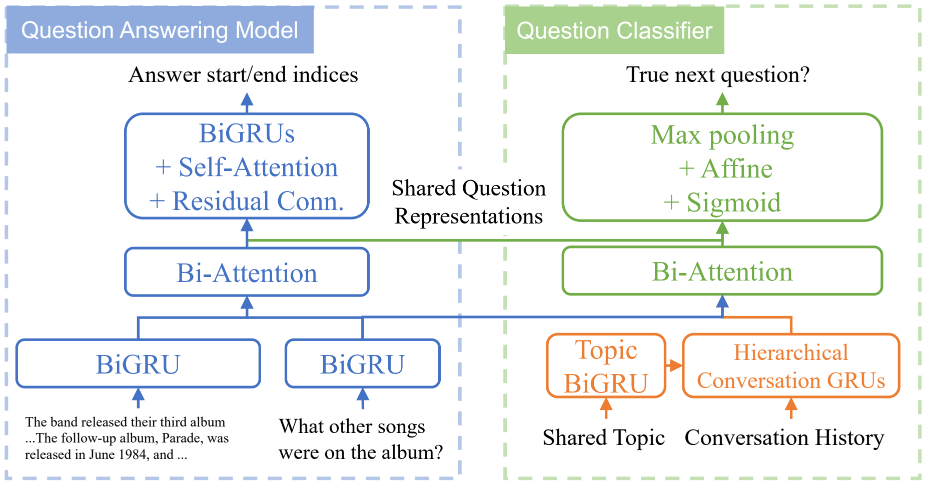


Figure 5.4: Architecture of the model to evaluate how informative and specific generated questions are.

estimate for $P(a|q, \mathcal{C}_{<j})$ to evaluate q's expected utility, which is practical for pragmatic reasoning at scale by avoiding the need for aggregating over a large set of candidate answers for each question. In contrast, previous work on pragmatics often require probabilistic normalization in the space of speaker utterances (questions) and listener actions (answers), which is intractable in our setting.

This definition of informativeness is also *explainable*: it is easy for a human to inspect the answer provided by the QA model and compare it to previous ones to understand how much new information has been revealed. Note that this definition itself also doesn't rely on any specific QA model, although more accurate QA models could result in more accurate estimates of informativeness. For simplicity, I use a bidirectional attention flow model (Seo et al., 2017) with self-attention (Clark and Gardner, 2018) as adapted for conversational QA by Choi et al. (2018) (see Figure 5.4).

5.3.4 Evaluating Question Specificity

Now that we have a metric to evaluate informativess, can we maximize it and obtain a good model for generating pragmatic conversational questions? It turns out that there are two issues with naïvely optimizing this value: generated questions could be overly generic or disruptive of the conversation flow while still acquiring new information. For instance, questions like *What else?* almost always reveal new information without being specifically related to the conversation at hand. On the other hand, in the example in Figure 5.1, *Did they go on tour for their 1983 album?* seems more disruptive (topic-changing) as the next question in the conversation than the candidate questions in the figure, although it does involve potentially new information.

To address this, I take a similar approach to previous work by selecting negative examples to target these issues and training a classifier to distinguish them from questions that were actually part of the conversation (Lowe et al., 2017; Rao and Daumé III, 2018). Once this classifier is trained, I can make use of the score it assigns different candidate questions to evaluate how specific each is to the current conversation history. Specifically, I select two kinds of negative questions to train the classifier: frequent questions from the training set (frequency>1) and random questions other than the observed one from the same conversation. I train a model (with shared parameters with the QA model, see Figure 5.4) to assign a probability that a question is the true next question (positive) given the conversation history, and define this quantity as the *specificity* S of the question q

$$S(q; \mathcal{H}_{<j}, \mathcal{T}) := P_\xi(q \text{ is positive} | \mathcal{H}_{<j}, \mathcal{T}), \tag{5.4}$$

where ξ is the parameters of the classifier optimized with binary cross entropy loss. Once this classifier is trained jointly with the QA model, I can use this specificity reward to bias the model towards generating questions that are not only informative, but also specific to the given conversation history.

Conceptually, the idea of "specificity" as is defined in this context is related to a few separate but connected concepts in natural language processing, namely discourse coherence, relevance, and reducing genericness in natural language generation. The coherence and relevance of a piece of text in a discourse is highly correlated with the perceived quality of the generated text, especially when considering whether the content presented is on the same topic as its surrounding context, and whether it is presented in a "logical" order. Since the proposed specificity classifier contrasts the true follow-up question against randomly sampled questions from other conversations, as well as questions in the conversation presented out of order, it is trained to discern questions that are less relevant and coherent in the current discourse, and thus discourage the question generator from generating questions with these issues. Previous work has approached generating coherent utterances in conversations through learning similar distributed representations (Baheti et al., 2018; Xu et al., 2018; Zhang et al., 2018a), while we make use of a discriminative classifier. More recently, the idea of discerning discourse consistency has also been applied to large pretrained language models, where it is similarly manifested as a classification task that determines whether sentence pairs or groups follow each other in the order specified in the potentially shuffled training data (Devlin et al., 2019; Iter et al., 2020). The authors have shown that this additional task of discourse coherence sometimes yields performance gains when the pretrained models are finetuned on downstream tasks.

On the other hand, since negative examples for the proposed specificity classifier also come from other conversations are frequent questions in the training set, it also discourages the question generator from generating overly generic questions. Previous work has approached the genericness issue in conversational natural language generation by proposing training objectives that maximize the utility of the generated utterance estimated with adversarial networks (Rao and Daumé III, 2019), specificity estimates that are (Ko et al., 2019a,b), or the mutual information between the generated turn and previous ones (Li et al., 2016). Specifically, aside from maximizing the probability of predicting the next utterance from the conversational history, an auxiliary model is often trained th predict relevant values (utility, specificity, and the likelihood of previous turns, respectively) from the generated utterance, associated with the conversation history. Then, this quantity is either used to finetune the generative model, or rerank the generated outputs. As a result, the model is less prone to generate generic utterances like “I don’t know” or “ I’m fine”, but instead outputs utterances that are more specific to the conversational history provided. What I have proposed here can be viewed as a generalization of these approaches, where the objective to be optimized at the time of generation is implicitly specified via a parameterized model by choosing negative examples for contrast.

5.3.5 Generating Informative and Specific Questions

Given the informativeness metric and specificity reward, we can improve upon these by maximizing the following reward function that blends the two in a weighted sum

$$R(q;\mathcal{C}_{<j}) = \lambda_1 I(q;\mathcal{C}_{<j}) + (1-\lambda_1) S(q;\mathcal{H}_{<j},\mathcal{T}). \tag{5.5}$$

Since this quantity can only be evaluated once a complete question has been generated, the non-differentiability of the decoding process prevents us from directly optimizing it with respect to θ using gradient-based optimization. However, we can still estimate the gradient of the expected reward of generated questions, $\mathbb{E}_{q\sim P_\theta}[R(q)]$ using REINFORCE (Williams, 1992), a reinforcement learning technique. For an example q, the gradient estimate is the gradient of the following loss function

$$\ell_R(q) = -\big[R(\hat{q}) - b(q)\big] \log P_\theta(\hat{q}) \tag{5.6}$$

where $\hat{q}$ is a sample from P_θ and I dropped the dependency on $\mathcal{C}_{<j}$ for notational clarity. $b(q)$ is called the *baseline function*, which, if chosen carefully, reduces the variance of this gradient estimate and results in faster convergence. I apply a technique called self-critical sequence training (Rennie et al., 2017), which selects $b(q) = R(q_\text{G})$, the reward obtained by the greedily decoded sequence, q_G, from the question generator.

To ensure that the generator maximizes the desired reward function without losing fluency in generated questions, I combine ℓ_R with negative log likelihood during model finetuning (Paulus et al., 2018). We finetune a pretrained question generator (trained with ℓ_NLL) using the following objective

$$\ell = \lambda_2 \ell_R + (1-\lambda_2)\ell_\text{NLL}. \tag{5.7}$$

Here, $\ell_\mathcal{R} = \frac{1}{N_\text{p}}\sum_{i=1}^{N}\sum_{j=1}^{M_i} \ell_\mathcal{R}(q_j^{(i)})$. I choose $\lambda_1 = 0.5$ and $\lambda_2 = 0.98$ in my experiments, which were chosen by tuning the model on the dev set.

5.3.6 Model Details

In this section, I include the details of the question generation model and the informativeness/specificity model I used in the experiments.

Question Generator

For the input to the encoder and decoder models in the question generator, I tokenize them with the spaCy toolkit,[2] and initialize word representations with 100-dimensional GloVe vectors (Pennington et al., 2014). As shown in Figure 5.2, I also introduce special XML-like symbols to delimit different parts of the input to various models. The representations of these special symbols are randomly initialized, and finetuned with those of the top 1000 most frequent words in the training set during training.

For the topic containing the title of the Wikipedia page and the background on the entity after concatenating them with special symbols, I feed them into a topic BiLSTM model and obtain the topic representation with a multi-layer perceptron (MLP) attention mechanism, using the concatenated final state from each direction of the BiLSTM as the key

$$h_{\mathcal{T}} = \mathrm{BiLSTM}_{\mathcal{T}}(x_{\mathcal{T}}), \tag{5.8}$$

$$h_{\mathcal{T}}^{\mathrm{attn}} = \mathrm{MLPSelfAttn}(h_{\mathcal{T}}). \tag{5.9}$$

I use this representation to initialize the BiLSTM we use to encode each pair of turns in the conversation that contains a question and its corresponding answer

$$h_{\mathcal{H}_j}^0 = \mathrm{BiLSTM}_{\mathcal{H},\mathrm{pair}}(x_{\mathcal{H}_j}, h_{\mathcal{T}}^{\mathrm{attn}}), \tag{5.10}$$

which I in turn use as the input to our unidirectional LSTM model to obtain the representation of the entire conversation up until a certain turn

$$h_{\mathcal{H}} = \mathrm{BiLSTM}_{\mathcal{H},\mathrm{conv}}([h_{\mathcal{H}_1}^0, \cdots, h_{\mathcal{H}_{|\mathcal{H}|}}^0]). \tag{5.11}$$

Note that for modeling simplicity, I use the section title as the 0th "turn" for each conversation. I similarly obtain a summary representation of the conversation with MLP self-attention

$$h_{\mathcal{H}}^{\mathrm{attn}} = \mathrm{MLPSelfAttn}(h_{\mathcal{H}}), \tag{5.12}$$

and concatenate it with $h_{\mathcal{T}}^{\mathrm{attn}}$ to initialize the decoder.

[2]`https://spacy.io/`

Split	Orig.	# Entities	# Dialogues	# QA pairs
Train	Train	2727	10572	76338
Dev	Train	264	995	7230
Test	Dev	721	1000	7354

Table 5.1: Data split of the QuAC dataset used in our experiments.

To represent the input words in the decoder, I use the same embedding matrix as the encoder. I also employ weight tying between the input embeddings and the output weights for word prediction to reduce parameter budget (Inan et al., 2017; Press and Wolf, 2017). For each word in the decoder input, I concatenate its embedding with $h_{\mathcal{T}}^{\text{attn}}$ for topical context. I provide the decoder access through attention to all of the representations of encoded tokens, *i.e.*, $[h_{\mathcal{T}}^{\text{attn}}, h_{\mathcal{H}_0}^0, \cdots, h_{\mathcal{H}_{|\mathcal{H}|}}^0]$. Finally, the weighted average of encoder representations is combined with the decoder LSTM's representation of the decoded sequence to yield a probabilistic distribution over words in the vocabulary.

Informativeness/Specificity Model

For informativeness, I follow closely the open implementation of BiDAF++ for QuAC that is available in AllenNLP (Gardner et al., 2018). For each word, I concatenate its word representations with character representations derived from a convolutional neural network from its character spelling. I replace the ELMo embeddings with GloVe ones for computational efficiency, which results in a relatively small drop in QA performance compared to AllenNLP's implementation (by about 2–3 F_1 on the official dev set). Note that following Choi et al. (2018), I use gated recurrent units (GRUs; Cho et al., 2014) in this part of the model.

For the specificity model, I first encode the topic and conversation history in a similar fashion as I did for the encoder in the question generator. Then, this representation is combined with the question representation from the BiGRU encoder in the QA model via a bidirectional attention (bi-attention) mechanism. The resulting representation is combined with the question representation from the bi-attention in the QA model, and max pooled over time, before an affine transform is applied to convert the representation into a score.

5.4 Experiments

Data. For experiments, I use the QuAC dataset presented by Choi et al. (2018). Although other similar datasets share some common characteristics, some crucial differences render them inapplicable for the experiments. For instance, CoQA (Reddy et al., 2019) gives both agents access to the context, while Wizard of Wikipedia (Dinan et al., 2019) doesn't assign the student agent clear goals of acquiring new information.

System	dev				test			
	PPLX	ROUGE-L	INFO	SPEC	PPLX	ROUGE-L	INFO	SPEC
Reference	–	–	0.639	0.825	–	–	0.635	0.821
Baseline	**8.46**	25.5	0.653	0.761	**8.08**	24.3	0.664	0.779
Our Model	9.24	**25.6**	**0.740**	**0.834**	8.79	**24.9**	**0.752**	**0.835**

Table 5.2: Evaluation of the baseline system, our pragmatically finetuned system, and the reference questions on conventional metrics as well as our proposed metric and reward functions.

Since QuAC's test set is held private for fair evaluation, I repurpose the original dev set as my test set. I randomly split the training set into training and development partitions, ensuring that the Wikipedia entities discussed in conversations do not overlap between these partitions. The goal of the split is to obtain a development set that is roughly as large as the repurposed test set. The statistics of our data split can be found in Table 5.1.

Training. I follow the recipe available in AllenNLP (Gardner et al., 2018) to train the QA model on QuAC, and make sure that it obtains performance on par with that reported by Choi et al. (2018) on the official dev set (with multiple answer references).[3] I use the Adam optimizer (Kingma and Ba, 2015) with default hyperparameters to train and finetune our question generator, and anneal the learning rate by 0.5 whenever dev performance doesn't improve for more than 3 consecutive epochs (patience=3). When training finishes, the specificity classifier achieves approximately 75% F_1 on the dev set when the true next question, sampled frequent questions and random questions from the same conversation have a balanced ratio of 1:1:1. For unanswerable questions in QuAC, I revise Equation (5.3) and set informativeness to zero if the predicted answer is `CANNOTANSWER`, as the answer does not reveal new information about the hidden knowledge $\mathcal{K}$.

5.5 Results

5.5.1 Metric-based Evaluation

For the baseline model and my model finetuned for informativeness and specificity, I generate predictions with greedy decoding for simplicity. I evaluate them on conventionally used metrics such as perplexity (PPLX) of the reference question and the F_1 score of the ROUGE-L metric (ROUGE-L) (Lin, 2004) between the predicted questions and the reference. The former helps verify the overall quality of our model, while the latter helps us compare single-reference metrics to our proposed ones. I also report the informativeness metric (INFO) and specificity reward (SPEC) for these models, and compare them to the reference questions on these measures on both the dev and test sets.

[3]However, in practice, I remove the ELMo component, which greatly speeds up computation at the cost of only losing 2–3 F_1 in answer prediction.

	Win	Tie	Lose	p-value
Ours vs. Baseline				
Overall	30.0%	48.0%	22.0%	0.108
INFO	21.0%	68.5%	10.5%	0.015
SPEC	26.0%	53.5%	20.5%	0.171
Ours vs. Human Reference				
Overall	16.0%	25.0%	59.0%	$< 10^{-6}$
INFO	9.0%	72.0%	19.0%	0.011
SPEC	13.0%	27.5%	59.5%	$< 10^{-6}$

Table 5.3: Human evaluation comparing questions the proposed system generated to those from the baseline, as well as the original reference questions in QuAC. I perform a bootstrap test with 10^6 samples for the difference between pairs of systems and report the p-values here.

As shown in Table 5.2, the baseline model and our pragmatically finetuned model achieve comparable performance when evaluated against the reference question using n-gram overlap metrics (ROUGE-L), and the perplexity of the reference question is only slightly worse. As expected, these metrics tell us nothing about how well the model is going to fare in actual communication, because perplexity doesn't evaluate the usefulness of generated questions, and ROUGE-L can barely tell these systems apart.

We can also see in Table 5.2 that the finetuned model improves upon the baseline model on both informativeness and specificity. Further, I notice that despite their high specificity, the reference questions are only about as informative as our baseline questions on average, which is a bit surprising at first sight. Further analysis reveals that about 12.6% of dev questions and 15.7% of test ones are considered unanswerable by crowd workers, which is a byproduct of the information-asymmetric setting adopted when the data was collected. As a result, many reference questions could be considered uninformative by our definition, since they might cause the QA model to abstain from answering.

5.5.2 Human Evaluation

Although the results in Table 5.2 show that my model sees substantial improvements on the proposed informativeness and specificity metrics, it remains unclear whether these improvements correlate well with human judgement of quality, which is critical in the application of the resulting system. To study this, I conduct a comparative human evaluation.

I randomly selected 100 turns from the test set, and asked two NLP PhD students to evaluate the reference questions, as well as those generated by the baseline model and our model. These questions are evaluated on their overall quality, informativeness, and specificity. System identity is never revealed to the annotators, and the order of the systems is shuffled for each comparison.

For each of these three metrics, the annotators are asked to rank the candidate questions on each metric, and ties are allowed in each of them. Prior to annotation, both annotators were educated to follow the same guidelines to encourage high agreement. I include the guidelines in Figure 5.5 for reference.

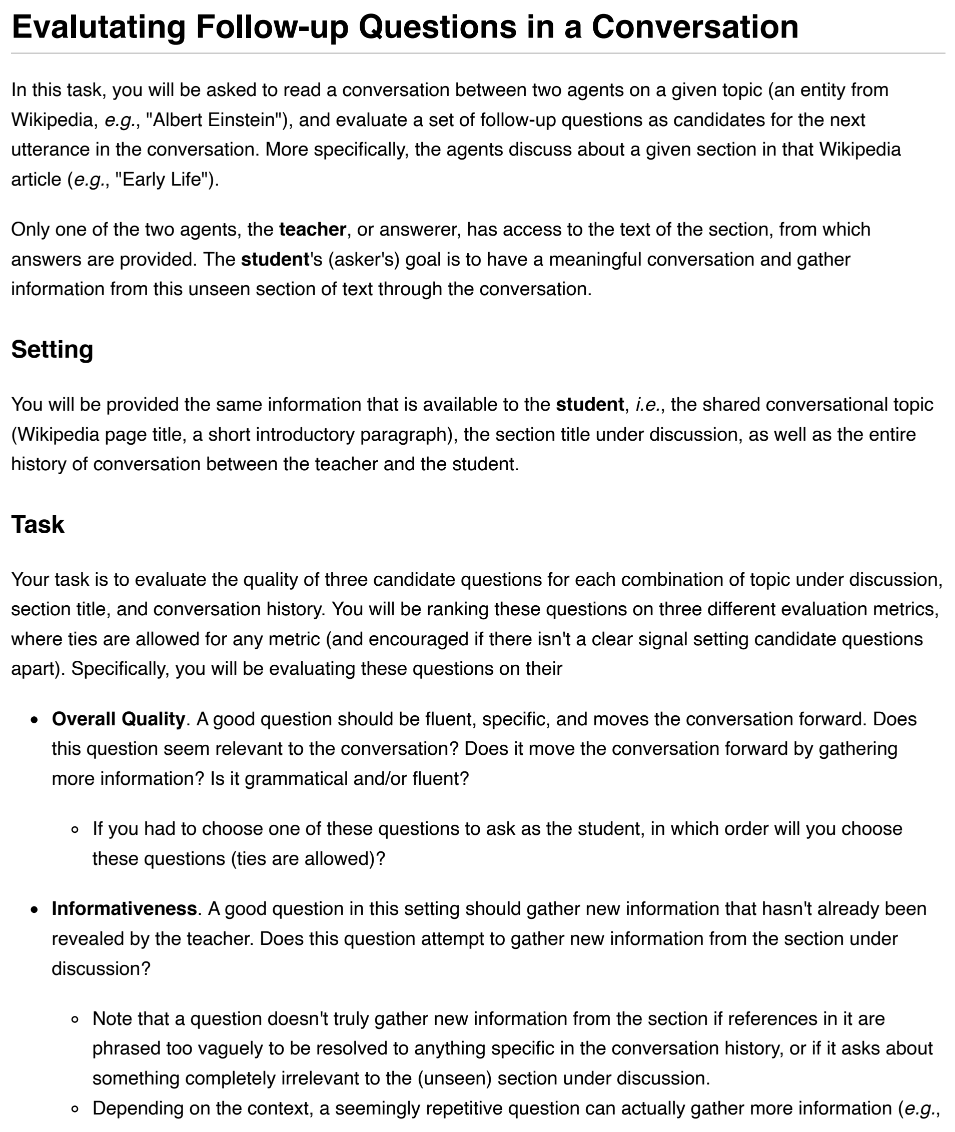
As shown in Table 5.3, despite there being many ties, human annotators favor my system over the baseline on informativeness (89.5% of our questions are considered equally or more informative) with a statistically significant margin. My system is also favored on overall quality (78.0%) and specificity (79.5%) to a slight extent, although not statistically significantly. This difference is partly due to the inherent nature of these annotation tasks: while the annotators agree on 77.3% of the pair-wise judgements regarding informativeness, agreement decreases to 71.7% for overall quality and 70.3% for specificity since they are more subjective. It is encouraging, however, that my system is also considered equally or more informative than the human reference 81% of the time. What negatively affects human's perception of the overall quality of questions our system generates is largely attributable to the over-genericness of these questions compared to the references.

5.6 Analysis

I further analyze concrete examples of generated questions in conversations to understand the behavior of my informativeness and specificity metrics.

Case Study. To sanity check whether my informativeness metric and specificity reward match human intuition, we manually inspect a few examples from the test set. Figure 5.6 represents a case where all the questions our system generated are considered equal to or more informative than the reference and baseline generated questions by our metric. As shown in the example, the baseline system is prone to generating topical but uninformative questions (BL_2 and BL_3). My system finetuned on the reward function is more pragmatic and asks about relevant questions that can likely be answered from the unseen paragraph $\mathcal{K}$. My informativeness metric also correctly identifies that both Ours_3 and Ref_3 are good questions that reveal new information about $\mathcal{K}$, although there is very little overlap between the two. On the other hand, the specificity reward successfully identifies that BL_3 and Ref_4 are the least specific questions of their respective turn, where the former is disconnected from the most recent topic under discussion (the song), the latter is phrased in an overly generic way.

I also demonstrate some clear failure cases. In Figure 5.7, we see that our informativeness and specificity measures make judgements a human will unlikely make, as the topic implies $\mathcal{K}$ is unlikely to contain information about Moyet's first album/recording. In fact, the QA model fails to recognize that these questions ($\text{BL}_{1,2}$, $\text{Ours}_{1,2,3}$, Ref_1) are unanswerable, and instead assigns them high informativeness. The specificity model, on the other hand, fails to recognize near paraphrases



Evalutating Follow-up Questions in a Conversation

In this task, you will be asked to read a conversation between two agents on a given topic (an entity from Wikipedia, *e.g.*, "Albert Einstein"), and evaluate a set of follow-up questions as candidates for the next utterance in the conversation. More specifically, the agents discuss about a given section in that Wikipedia article (*e.g.*, "Early Life").

Only one of the two agents, the **teacher**, or answerer, has access to the text of the section, from which answers are provided. The **student**'s (asker's) goal is to have a meaningful conversation and gather information from this unseen section of text through the conversation.

Setting

You will be provided the same information that is available to the **student**, *i.e.*, the shared conversational topic (Wikipedia page title, a short introductory paragraph), the section title under discussion, as well as the entire history of conversation between the teacher and the student.

Task

Your task is to evaluate the quality of three candidate questions for each combination of topic under discussion, section title, and conversation history. You will be ranking these questions on three different evaluation metrics, where ties are allowed for any metric (and encouraged if there isn't a clear signal setting candidate questions apart). Specifically, you will be evaluating these questions on their

- **Overall Quality**. A good question should be fluent, specific, and moves the conversation forward. Does this question seem relevant to the conversation? Does it move the conversation forward by gathering more information? Is it grammatical and/or fluent?
    - If you had to choose one of these questions to ask as the student, in which order will you choose these questions (ties are allowed)?
- **Informativeness**. A good question in this setting should gather new information that hasn't already been revealed by the teacher. Does this question attempt to gather new information from the section under discussion?
    - Note that a question doesn't truly gather new information from the section if references in it are phrased too vaguely to be resolved to anything specific in the conversation history, or if it asks about something completely irrelevant to the (unseen) section under discussion.
    - Depending on the context, a seemingly repetitive question can actually gather more information (*e.g.*,

Figure 5.5: Human evaluation guidelines to compare system-generated questions with the human reference.

Background: Spandau Ballet (English band) **Topic:** 1983–1989: International success and decline
Candidate Questions **BL$_1$:** What happened in 1983? **Ours$_1$:** What happened in 1983? **Ref$_1$:** What was the first indication of Spandau Ballet's success at the international level?
Ans$_1$: The follow-up album, *Parade*, was released in June 1984, and its singles were again big successes in the charts in Europe, Oceania and Canada.
BL$_2$: What was the name of the album? **Ours$_2$:** What was the most popular single from the album? **Ref$_2$:** What were the notable songs from the album Parade?
Ans$_2$: The album's opening song, *"Only When You Leave"*.
BL$_3$: What was the name of the album that was released? **Ours$_3$:** What other songs were on the album? **Ref$_3$:** How did the opening song do on the charts?
Ans$_3$: Became the band's last American hit.
BL$_4$: What was the last album that they released? **Ours$_4$:** What other songs were on the album? **Ref$_4$:** Are there any other interesting aspects about this article?

Figure 5.6: A success example where the proposed automatic metrics align well with human judgement of informativeness and specificity, when comparing questions generated by the baseline (BL), the proposed system (Ours), and the original human-written reference (Ref).

(BL$_1$ vs Ours$_1$) and a question that was likely just answered (BL$_3$). A positive finding in this example is that the informativeness metric is well-aligned with pragmatic behavior in the fourth turn—had Moyet won the Grammy, the previous answer (A$_3$) would have mentioned it instead of just her nomination.

I include in Figures 5.8 and 5.9 the contexts that contain the answer for these examples for the reader's reference, where gold answers in the case study are highlighted in the paragraphs. Following Choi et al. (2018), I concatenate an artificial `CANNOTANSWER` token to the end of the paragraph for the question answering model to abstain from answering the question.

Explainable Informativeness. As stated in Section 5.3.3, my definition of informativeness is explainable to humans—I demonstrate this with concrete examples. For instance, in the example in Figure 5.6, although the question *What happened in 1983?* is phrased rather vaguely, the QA model is able to identify its correct answer from the paragraph *The band released their third album, True, in March 1983*, which offers new information. Similarly, the QA model correctly identifies that the question our model generated on the second turn (Ours$_2$) has the same answer as the human reference (Ref$_2$), which introduces a new entity into the conversation. BL$_2$ and BL$_3$ are deemed uninformative in this case since the QA model offered the same answer about the album *True* again. Although this answer is about an incorrect entity in this context (the album *True* instead of *Parade*,

Background: Alison Moyet (English singer)
Topic: 1990s: Further recordings and hiatus
Candidate Questions
BL$_1$: What was the first album released?
Ours$_1$: What was her first recording?
Ref$_1$: What did she do in 1990?
INFO: *Ref = BL = Ours* SPEC: *Ref ≈ Ours > BL*
Ans$_1$: `CANNOTANSWER`
BL$_2$: What was her first album?
Ours$_2$: What was her first album?
Ref$_2$: What else did she record in the 1990's?
INFO: *Ref = BL = Ours* SPEC: *BL=Ours>Ref*
Ans$_2$: Hoodoo.
BL$_3$: What was the name of the album?
Ours$_3$: What was her first album?
Ref$_3$: Did he reach bill board charts?
INFO: *BL = Ref > Ours* SPEC: *Ours>BL>Ref*
Ans$_3$: Yes, and Moyet was nominated for a Grammy for the single
BL$_4$: What was the song called?
Ours$_4$: What other songs did she release?
Ref$_4$: Did she receive an award?
INFO: *BL = Ours > Ref* SPEC: *BL≈Ours≈Ref*

Figure 5.7: A failure case of the proposed informativeness metric and specificity reward, when comparing reference questions (Ref), baseline generated questions (BL), and those generated by the proposed model (Ours).

The band released their third album, True, in March 1983. Produced by Tony Swain and Steve Jolley, the album featured a slicker pop sound. It was at this point that Steve Norman began playing saxophone for the band. Preceded by the title track which reached number one in various countries, the album also reached number onein the UK. Their next single, "Gold", reached number 2. [The follow-up album, Parade, was released in June 1984, and its singles were again big successes in the charts in Europe, Oceania and Canada.]$_{\text{Ans}_1}$ [The album's opening song, "Only When You Leave"]$_{\text{Ans}_2}$, [became the band's last American hit.]$_{\text{Ans}_3}$ At the end of 1984, the band performed on the Band Aid charity single and in 1985 performed at Wembley Stadium as part of Live Aid. During this same year, Spandau Ballet achieved platinum status with the compilation The Singles Collection, which kept the focus on the band between studio albums and celebrated its five years of success. However, the album was released by Chrysalis Records without the band's approval and the band instigated legal action against the label. In 1986, Spandau Ballet signed to CBS Records and released the album Through the Barricades, in which the band moved away from the pop and soul influences of True and Parade and more toward rock. Though the first single, "Fight for Ourselves" peaked at 15 in the UK, the title track and the album both reached the Top 10 in the UK and Europe. After a hiatus from recording, the band released their next album, Heart Like a Sky, in September 1989. The album and its singles were unsuccessful in the UK, and the album itself was not released in the United States. It did, however, do well in Italy (where its singles "Raw" and "Be Free with Your Love" reached the Top 10) and also in Belgium, Germany and the Netherlands. CANNOTANSWER

Figure 5.8: Private context that contains the answers to questions in our case study example in Figure 5.6.

Following a period of personal and career evaluation, [Hoodoo]$_{\text{Ans}_2}$ was released in 1991. The album sold respectably in the UK, [and Moyet was nominated for a Grammy for the single "It Won't Be Long".]$_{\text{Ans}_3}$ However, the release of Hoodoo marked the beginning of an eight-year fight for Moyet to secure complete control of her artisticdirection. Like many similar artists (including Aimee Mann and the late Kirsty MacColl), Moyet was reluctant to record a radio-friendly "pop" album simply for the sake of creating chart hits. Moyet's next album, Essex (1994), was also a source of controversy for her; in order for the album to be released, her label(now Sony) insisted that certain Essex tracks be re-recorded and re-produced, and that there be additional material remixed to create a more' commercial' package. The video for the single "Whispering Your Name" again featured Dawn French. Following the release of Essex, Sony released a greatest hits compilation of Moyet's work. Singles entered the UK charts at No. 1 and, following a UK tour, was re-issued as a double CD set which included "Live (No Overdubs)", a bonus live CD. Upon re-issue, Singles charted again, this time in the Top 20. Due to prolonged litigation with Sony, Moyet did not record or release a new studio album for over eight years after the release of Essex. During this time, however, she recorded vocals for Tricky, Sylk-130, Ocean Colour Scene, The Lightning Seeds, and King Britt, and was featured on the British leg of the Lilith Fair tour. 2001 saw the release of The Essential Alison Moyet CD, and in 2002 The Essential Alison Moyet DVD. In 1995, she sang back-up vocals with Sinead O'Connor for one of Dusty Springfield's last television appearances, singing "Where Is a Woman to Go ?" on the music show Later With Jools Holland. [CANNOTANSWER]$_{\text{Ans}_1}$

Figure 5.9: Private context that contains the answers to questions in our case study example in Figure 5.7.

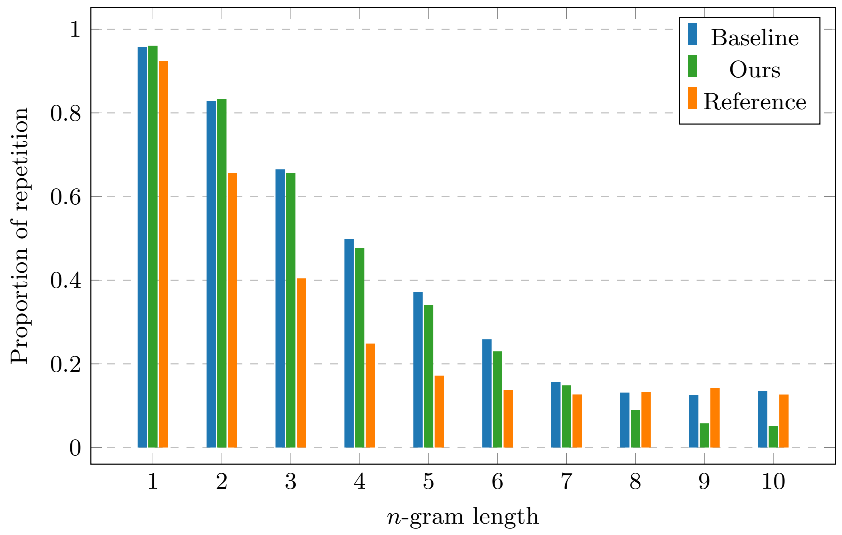


Figure 5.10: Proportion of repeated n-grams in questions from the conversation history. As can be seen from the plot, the proposed pragmatic system reduces the amount of n-grams repeated from previous questions especially for longer n-grams.

which is the focus of discussion), the large amount of overlap between this answer and Ans_1 is still sufficient to regard these questions as less informative.

In the example in Figure 5.7, as we have mentioned previously, the QA model is poorly calibrated for when it should abstain from answering questions. Specifically, for all the "first album" questions, the QA model answers *Hoodoo was released in 1991*, which is factually incorrect. It does correctly identify that Ref_4 is unanswerable from the paragraph, however. My model pragmatically avoids generating this uninformative question, because the answerer/paragraph should not have just mentioned her nomination had she won the award.

I note that this informativeness metric does have an exploitable flaw—it does not prevent the questioner from asking vague, open-ended questions (*e.g.*, *What else do you know?*) to acquire knowledge. In fact, we find this strategy is also adopted by QuAC's crowd workers. However, with the specificity reward to penalize genericness, this issue is alleviated in the questions my system generates. I will show next that my system repeats n-grams from previous questions less frequently.

I examine the outputs of our model to assess whether finetuning on the specificity reward results in more specific questions rather than generic and repetitive ones. To measure this, I compute the n-gram overlap between generated questions and all questions in the conversation history for all systems. The lower this repetition is, the more likely the system is bringing up new entities or topics in its questions, and thus more specific to the given conversation history. As can be seen in Figure 5.10, our system improves upon the baseline system by reducing this repetition noticeably in longer n-grams ($n \geq 3$). When n is very large ($n \geq 8$), the proposed pragmatic system is less repetitive even compared to the human reference, which often contains long and repetitive questions like *Are there any other interesting aspects about this article?* as a generic inquiry for more information.

5.7 Conclusion

In this chapter, I presented a system that generates inquisitive questions in information-seeking conversations. This is achieved by first defining new automatic metrics that evaluate the informativeness and specificity given the conversation history, then optimizing these as reward functions with reinforcement learning. After the question generation model is optimized for these rewards, it is able to generate pragmatically relevant and specific questions to acquire new information about an unseen source of textual knowledge through conversing with another agent. The informativeness metric proposed is also explainable, which allows us to analyze model output, and identify cases where the model genuinely behaved as expected, where it made mistakes in prediction but somehow produced the desired behavior by accident, or where it made irrecoverable mistakes that derailed the model on generating informative questions. This helps us identify areas of improvement more easily in a complex system, and make solid progress on improving the informativeness of these systems more quickly, so that they can maximize the efficiency at which they communicate with us. The proposed method presents a practical if shallow implementation of pragmatics in an open-domain communication setting beyond simple reference games, which extends the latter approaches by providing a path forward towards learning a general pragmatic agent that doesn't require complete knowledge of the state of the world. This presents a practical solution to improve the efficiency of communication when a computer system is gathering information in a natural language conversational interface.

Successful modeling of the information acquisition behavior in this setting not only solves the problem of gathering information, but more importantly, it is also important for improving the communication efficiency when NLP systems serve information to users. That is, it helps pave the way forward for question answering systems to provide more informative answers that are closer to what humans are able to provide in natural communications, and therefore communicate more information in fewer exchanges (one example is the "weekend plan" dialogue in Figure 1.5 on page 14). More specifically, this work provides us with a tool to "unroll" or simulate forward the interaction between our question answering system and an agent that is acquiring information, which helps us understand whether followup questions from the agent indicate trivial follow-up requests (as illustrated in Figure 1.5) or confusion. This allows question answering systems to formulate the answers they provide to help the question asker avoid having to ask a series of trivial follow-up questions to gather the same amount of information, or being confused about the lack of context in the answer and not knowing how to follow up. As a result, NLP systems will be able to communicate the knowledge encoded in text collections more efficiently to humans, to help us quickly gather the information we need.

I hope that this work brings the community's attention to the important problem of natural language communication under information asymmetry in an open-domain setting, which is an important means through which future NLP systems can both interact naturally with humans and learning from us through interaction.

Chapter 6

Conclusion and Future Directions

This dissertation has focused on natural language processing methods and systems that help broaden our human access to knowledge that is encoded in text through the form of asking and answering questions. The end goal is to leverage the scalability of computer systems to process large amounts of text, and surface answers to our information needs from the knowledge therein.

To this end, I have presented my work that addresses problems falling under two broad categories: one focuses on enhancing the language understanding capabilities of these systems, so that we can delegate the task of information gathering and aggregation to these systems; the other aims at enabling these systems to interact with us by asking questions to work out more complex information needs that require unpacking, and eventually affording us more flexibility in how we can access textual knowledge through them.

In my work, I have also demonstrated how building NLP systems to be explainable and controllable can not only help build trust with human users and establish robust human-computer collaborations as a result, but these properties also serve as an important source of inspiration for methods that drive our systems towards better statistical, computational, and communication efficiency in various applications through the reduction of intermediate representations of data and prediction to human-understandable forms to leverage human-inspired forms of inductive bias and supervision.

Looking into the future, I am most excited about two directions of research in natural language processing, one is building *explainable NLP systems* that allow us better understanding of how computers are carrying out the complex reasoning we delegate to them, and the other is further improving the *communication efficiency of NLP systems* in natural language interactions via pragmatic reasoning.

Explainable Natural Language Processing. I would like to begin by acknowledging that explainability is never a binary property of a system, but rather more like a continuum. This is partly because the evaluation of explainability is nuanced and could be viewed as a continuous scale, as I have presented in Chapter 1. More importantly, as NLP systems for complex tasks are often themselves made up of many different components, there is almost always an opportunity to "open up the black box" further and expose more explanations of system decisions and intermediate representations to help us understand whether it is doing the right thing at each step. One example from my own work is that the final reading comprehension component from GoldEn Retriever is largely treated as a black box, and little effort is taken to make it more explainable aside from making sure it predicts the supporting sentences that is required for evaluation on the HotpotQA dataset. When these explanations cannot be easily extracted from existing text and must be generated from NLP systems, one important challenge is to make sure that these explanations/justifications are consistent with the model's actual predictions. On the flipside, I must also acknowledge that not everything can or should be explained or justified, and consistent explanations can still sometimes hide systematic bias (*e.g.*, for loan decisions). But in general, I believe that exposing critical decisions or steps of these models in human-understandable formats would still be desirable in helping us gather tangible information about system behavior and bias indicators, if not in helping us control their behavior to address these issues.

Efficient Communication in NLP. I am excited about a future where humans and computers can collaborate toward a common objective and solve problems better and more efficiently than either working on these problems alone. More importantly, I believe that this is best achieved when computers can communicate efficiently with us in our language and means communication rather than just rigid and logical computer code, where natural language processing systems will play a crucial role. Such efficient communication usually requires each agent to be able to reason about others' goals and mental state pragmatically by observing their communication or lack thereof, which most humans are capable of subconsciously but computers are still incompetent at. In the nearer term, I hope my work in this dissertation on asking informative questions can help computer systems simulate the information gathering behavior of a less knowledgeable agent, and as a result drive the development of NLP systems that go beyond giving a correct answer when prompted with a question. Specifically, I hope that question answering systems in the near future can pragmatically paraphrase answers to reduce the potential confusion caused in the question asker, by forward simulating with different answers and picking ones that elicit non-trivial and non-generic follow-up questions. Such systems will be able to provide more information beyond verbatim answers to help us properly ground the answer in the shared background knowledge, and allow us to continue the carrying out the conversation and learn from the large collections of texts that support the knowledge of these NLP systems. Despite foreseeable challenges along the way, I hope this will provide the answer to more natural and efficient human-computer interaction and collaboration in the future when natural language is part of the interface.

Bibliography

Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT).*

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Tejasvi Chaganty, Jason Bolton, Melvin Jose Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D Manning. 2015. Bootstrapped self training for knowledge base population. In *TAC*.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations.*

Giusepppe Attardi. 2015. Wikiextractor. `https://github.com/attardi/wikiextractor`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations.*

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.*

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268.*

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*.

M. Neil Browne and Stuart M. Keeley. 2014. *Asking the Right Questions*, 11th edition. Pearson.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *Proceedings of the International Conference on Learning Representations*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 724–731.

Arun Tejasvi Chaganty, Ashwin Paranjape, Jason Bolton, Matthew Lamm, Jinhao Lei, Abigail See, Kevin Clark, Yuhao Zhang, Peng Qi, and Christopher D Manning. 2017. Stanford at TAC KBP 2017: Building a trilingual relational knowledge graph. In *Text Analysis Conference (TAC)*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Reuben Cohn-Gordon, Noah Goodman, and Christopher Potts. 2019. An incremental iterated response model of pragmatics. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 81–90.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 423–429, Barcelona, Spain. URL: https://www.aclweb.org/anthology/P04-1054, doi:10.3115/1218955.1219009.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*.

Beth Davey and Susan McBride. 1986. Effects of question-generation training on reading comprehension. *Journal of Educational Psychology*, 78(4):256.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable reasoning over a virtual knowledge base. In *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SJxstlHFPH.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations (ICLR)*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association of Computational Linguistics*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Association of Computational Linguistics*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL-HLT*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

Andrei Dulceanu, Thang Le Dinh, Walter Chang, Trung Bui, Doo Soon Kim, Manh Chien Vu, and Seokhwan Kim. 2018. PhotoshopQuiA: A corpus of non-factoid questions and answers for why-question answering. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A new Q&A dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2019. Hierarchical graph network for multi-hop question answering. *arXiv preprint arXiv:1911.03631*.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association of Computational Linguistics*.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Yifan Gao, Piji Li, Irwin King, and Michael R. Lyu. 2019. Interconnected question generation with coreference alignment and conversation flow modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4853–4862.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine*. O'Reilly Media, Inc.

Herbert P Grice. 1975. Logic and conversation. In *Syntax and Semantics*, volume 3, pages 41–58. Academic Press.

Jeroen Groenendijk. 1999. The logic of interrogation. In *Semantic and Linguistics Theory*.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W Bruce Croft. 2020. Antique: A non-factoid question answering benchmark. In *European Conference on Information Retrieval*, pages 166–173. Springer.

Tatsunori Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1689–1701.

Ahmed Hassan Awadallah, Ryen W White, Patrick Pantel, Susan T Dumais, and Yi-Min Wang. 2014. Supporting complex search tasks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 829–838. ACM.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Michael Heilman and Noah A Smith. 2010. Good question! Statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1993. The ATIS spoken language systems pilot corpus. `https://catalog.ldc.upenn.edu/LDC93S4B/`. Linguistic Data Consortium.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).*

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. 2019. FlowQA: Grasping flow in history for conversational machine comprehension. In *International Conference on Learning Representations (ICLR).*

Karen Huang, Michael Yeomans, Alison Wood Brooks, Julia Minson, and Francesca Gino. 2017. It doesn't hurt to ask: Question-asking increases liking. *Journal of personality and social psychology*, 113(3):430.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations.*

Dan Iter, Kelvin Guu, Larry Lansing, and Dan Jurafsky. 2020. Pretraining with contrastive sentence objectives improves discourse performance of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.*

Peter Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton Morrison. 2018. WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).*

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).*

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Annual Meeting of the Association of Computational Linguistics (ACL).*

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).*

Dan Jurafsky and James H. Martin. 2019. *Speech and Language Processing*, third edition. Unpublished draft.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 Interactive poster and demonstration sessions*. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wentau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR 2017)*.

Wei-Jen Ko, Greg Durrett, and Junyi Jessy Li. 2019a. Domain agnostic real-valued specificity prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6610–6617.

Wei-Jen Ko, Greg Durrett, and Junyi Jessy Li. 2019b. Linguistically-informed specificity and semantic plausibility for dialogue generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3456–3466.

Bernhard Kratzwald and Stefan Feuerriegel. 2018. Adaptive document retrieval for deep question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Empirical Methods for Natural Language Processing (EMNLP)*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*.

Rishabh Mehrotra, Prasanta Bhattacharya, and Emine Yilmaz. 2016. Deconstructing complex search tasks: A bayesian nonparametric approach for extracting sub-tasks. In *Proceedings of the 2016*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 599–605.

Rishabh Mehrotra and Emine Yilmaz. 2017. Extracting hierarchies of search tasks & subtasks via a bayesian nonparametric approach. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294. ACM.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *EMNLP*.

Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. 2017. ParlAI: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the Annual Conference of the Association of Computational Linguistics*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the Annual Conference of the Association of Computational Linguistics*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*, pages 17–22.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Will Monroe, Robert X. D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5:325–338.

Mao Nakanishi, Tetsunori Kobayashi, and Yoshihiko Hayashi. 2019. Towards answer-unaware conversational question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 63–71.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS).*

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2018. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 647–656. ACM.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. In *LREC*.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375.*

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Boyuan Pan, Hao Li, Ziyu Yao, Deng Cai, and Huan Sun. 2019. Reinforced dynamic reasoning for conversational question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2114–2124.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098.*

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, volume 14, pages 1532–1543.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. *arXiv preprint arXiv:2002.09758*.

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

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. Answering complex open-domain questions through iterative query generation. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Peng Qi, Yuhao Zhang, and Christopher D Manning. 2020. Stay hungry, stay focused: Generating informative and specific questions in information-seeking conversations. *arXiv preprint arXiv:2004.14530*.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. *Proceedings of the 15th Conference of the European Association for Computational Linguistics (EACL 2017)*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2737–2746.

Sudha Rao and Hal Daumé III. 2019. Answer-based Adversarial Training for Generating Clarification Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Bryan Rink and Sanda Harabagiu. 2010. UTD: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics.

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. NIST.

Robert van Rooy. 2003. Questioning to resolve decision problems. *Linguistics and Philosophy*, 26(6):727–763.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*.

Shimi Salant and Jonathan Berant. 2018. Contextualized word representations for reading comprehension. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4974–4983.

Roger C Schank and Robert P Abelson. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures.* Psychology Press.

Thomas Scialom and Jacopo Staiano. 2019. Ask to learn: A study on curiosity-driven question generation. *arXiv preprint arXiv:1911.03350.*

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR).*

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In *NAACL-HLT.*

Robert F Simmons, Sheldon Klein, and Keren McConlogue. 1964. Indexing and dependency logic for answering english questions. *American Documentation*, 15(3):196–204.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL 2015).*

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics.*

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.*

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *AAAI*, pages 9073–9080.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. r^3: Reinforced reader-ranker for open-domain question answering. In *AAAI Conference on Artificial Intelligence*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming 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)*, volume 1, pages 189–198.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Wikipedia contributors. 2020a. Homo sapiens — Wikipedia, the free encyclopedia. [Online; accessed 16-July-2020]. URL: `https://en.wikipedia.org/w/index.php?title=Homo_sapiens&oldid=966296007`.

Wikipedia contributors. 2020b. Massachusetts — Wikipedia, the free encyclopedia. [Online; accessed 26-August-2020]. URL: `https://en.wikipedia.org/w/index.php?title=Massachusetts&oldid=974918392`.

Wikipedia contributors. 2020c. New york (state) — Wikipedia, the free encyclopedia. [Online; accessed 26-August-2020]. URL: `https://en.wikipedia.org/w/index.php?title=New_York_(state)&oldid=974297034`.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. 2019. Simplifying graph convolutional networks. In *International Conference on Machine Learning*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. DCN+: Mixed objective and deep residual coattention for question answering. In *Proceedings of the International Conference on Learning Representations*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Xinnuo Xu, Ondřej Dušek, Ioannis Konstas, and Verena Rieser. 2018. Better conversations by modeling, filtering, and optimizing for coherence and diversity. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1785–1794.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018a. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhilin Yang, Saizheng Zhang, Jack Urbanek, Will Feng, Alexander H Miller, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018b. Mastering the dungeon: Grounded language learning by mechanical turker descent. In *Proceedings of the International Conference on Learning Representations*.

Deming Ye, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, and Maosong Sun. 2019. Multi-paragraph reasoning with knowledge-enhanced graph neural network. *arXiv preprint arXiv:1911.02170*.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2014)*, pages 2335–2344.

Hainan Zhang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2018a. Reinforcing coherence for sequence to sequence model in dialogue generation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4567–4573.

Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, Jianfeng Lu, and Anton Van Den Hengel. 2018b. Goal-oriented visual question generation via intermediate rewards. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 186–201.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*.

Yuhao Zhang, Arun Tejasvi Chaganty, Ashwin Paranjape, Danqi Chen, Jason Bolton, Peng Qi, and Christopher D. Manning. 2016. Stanford at tac kbp 2016: Sealing pipeline leaks and understanding chinese. In *Text Analysis Conference*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018c. Graph convolution over pruned dependency trees improves relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. 2019. Coarse-grain fine-grain coattention network for multi-evidence question answering. In *Proceedings of the International Conference on Learning Representations*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, page 207.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671.