# NEURAL READING COMPREHENSION AND BEYOND

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

Danqi Chen
December 2018



This dissertation is online at: http://purl.stanford.edu/gd576xb1833

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Christopher Manning, Primary Adviser**

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Dan Jurafsky**

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Percy Liang**

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Luke Zettlemoyer**

Approved for the Stanford University Committee on Graduate Studies.

**Patricia J. Gumport, Vice Provost for Graduate Education**

*This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.*

# Abstract

Teaching machines to understand human language documents is one of the most elusive and long-standing challenges in Artificial Intelligence. This thesis tackles the problem of reading comprehension: how to build computer systems to read a passage of text and answer comprehension questions. On the one hand, we think that reading comprehension is an important task for evaluating how well computer systems understand human language. On the other hand, if we can build high-performing reading comprehension systems, they would be a crucial technology for applications such as question answering and dialogue systems.

In this thesis, we focus on neural reading comprehension: a class of reading comprehension models built on top of deep neural networks. Compared to traditional sparse, hand-designed feature-based models, these end-to-end neural models have proven to be more effective in learning rich linguistic phenomena and improved performance on all the modern reading comprehension benchmarks by a large margin.

This thesis consists of two parts. In the first part, we aim to cover the essence of neural reading comprehension and present our efforts at building effective neural reading comprehension models, and more importantly, understanding what neural reading comprehension models have actually learned, and what depth of language understanding is needed to solve current tasks. We also summarize recent advances and discuss future directions and open questions in this field.

In the second part of this thesis, we investigate how we can build practical applications based on the recent success of neural reading comprehension. In particular, we pioneered two new research directions: 1) how we can combine information retrieval techniques with neural reading comprehension to tackle large-scale open-domain question answering; and

2) how we can build conversational question answering systems from current single-turn, span-based reading comprehension models. We implemented these ideas in the DrQA and CoQA projects and we demonstrate the effectiveness of these approaches. We believe that they hold great promise for future language technologies.

# Acknowledgments

The past six years at Stanford have been an unforgettable and invaluable experience to me. When I first started my PhD in 2012, I could barely speak fluent English (I was required to take five English courses at Stanford), knew little about this country and had never heard of the term "natural language processing". It is unbelievable that over the following years I have actually been doing research about language and training computer systems to understand human languages (English in most cases), as well as training myself to speak and write in English. At the same time, 2012 is the year that deep neural networks (also called deep learning) started to take off and dominate almost all the AI applications we are seeing today. I witnessed how fast Artificial Intelligence has been developing from the beginning of the journey and feel quite excited —– and occasionally panicked —– to be a part of this trend. I would not have been able to make this journey without the help and support of many, many people and I feel deeply indebted to them.

First and foremost, my greatest thanks go to my advisor Christopher Manning. I really didn't know Chris when I first came to Stanford — only after a couple of years that I worked with him and learned about NLP, did I realize how privileged I am to get to work with one of the most brilliant minds in our field. He always has a very insightful, high-level view about the field while he is also uncommonly detail oriented and understands the nature of the problems very well. More importantly, Chris is an extremely kind, caring and supportive advisor that I could not have asked for more. He is like an older friend of mine (if he doesn't mind me saying so) and I can talk with him about everything. He always believes in me even though I am not always that confident about myself. I am forever grateful to him and I have already started to miss him.

I would like to thank Dan Jurafsky and Percy Liang — the other two giants of the

Stanford NLP group — for being on my thesis committee and for a lot of guidance and help throughout my PhD studies. Dan is an extremely charming, enthusiastic and knowledgeable person and I always feel my passion getting ignited after talking to him. Percy is a superman and a role model for all the NLP PhD students (at least myself). I never understand how one can accomplish so many things at the same time and a big part of this dissertation is built on top of his research. I want to thank Chris, Dan and Percy, for setting up the Stanford NLP Group, my home at Stanford, and I will always be proud to be a part of this family.

It is also my great honor to have Luke Zettlemoyer on my thesis committee. The work presented in this dissertation is very relevant to his research and I learned a lot from his papers. I look forward to working with him in the near future. I also would like to thank Yinyu Ye for his time chairing my thesis defense.

During my PhD, I have done two wonderful internships at Microsoft Research and Facebook AI Research. I thank my mentors Kristina Toutanova, Antoine Bordes and Jason Weston when I worked at these places. My internship project at Facebook eventually leads to the DrQA project and a part of this dissertation. I also would like to thank Microsoft and Facebook for providing me with fellowships.

Collaboration is a big lesson that I learned, and also a fun part of graduate school. I thank my fellow collaborators: Gabor Angeli, Jason Bolton, Arun Chaganty, Adam Fisch, Jon Gauthier, Shayne Longpre, Jesse Mu, Siva Reddy, Richard Socher, Yuhao Zhang, Victor Zhong, and others. In particular, Richard — with him I finished my first paper in graduate school. He had very clear sense about how to define an impactful research project while I had little experience at the time. Adam and Siva — with them I finished the DrQA and CoQA projects respectively. Not only am I proud of these two projects, but also I greatly enjoyed the collaborations. We have become good friends afterwards. The KBP team, especially Yuhao, Gabor and Arun — I enjoyed the teamwork during those two summers. Jon, Victor, Shayne and Jesse, the younger people that I got to work with, although I wish I could have done a better job. I also want to thank the two teaching teams (7 and 25 people respectively) for the NLP class that I've worked on and that was a very unique and rewarding experience for me.

I thank the whole Stanford NLP Group, especially Sida Wang, Will Monroe, Angel

Chang, Gabor Angeli, Siva Reddy, Arun Chaganty, Yuhao Zhang, Peng Qi, Jacob Steinhardt, Jiwei Li, He He, Robin Jia and Ziang Xie, who gave me a lot of support at various times. I am even not sure if there could be another research group in the world better than our group (I hope I can create a similar one in the future). The NLP retreat, NLP BBQ and those paper swap nights were among my most vivid memories in graduate school.

Outside of the NLP group, I have been extremely lucky to be surrounded by many great friends. Just to name a few (and forgive me for not being able to list all of them): Yanting Zhao, my close friend for many years, who keeps pulling me out from my stressful PhD life, and I share a lot of joyous moments with her. Xueqing Liu, my classmate and roommate in college who started her PhD at UIUC in the same year and she is the person that I can keep talking to and exchanging my feelings and thoughts with, especially on those bad days. Tao Lei, a brilliant NLP PhD and my algorithms "teacher" in high school and I keep learning from him and getting inspired from every discussion. Thanh-Vy Hua, my mentor and "elder sister" who always makes sure that I am still on the right track of my life and taught me many meta-skills to survive this journey (even though we have only met 3 times in the real world). Everyone in the "cǎo yú" group, I am so happy that I have spent many Friday evenings with you.

During the past year, I visited a great number of U.S. universities seeking an academic job position. There are so many people I want to thank for assistance along the way —— I either received great help and advice from them, or I felt extremely welcomed during my visit —— including Sanjeev Arora, Yoav Artzi, Regina Barzilay, Chris Callison-Burch, Kai-Wei Chang, Kyunghyun Cho, William Cohen, Michael Collins, Chris Dyer, Jacob Eisenstein, Julia Hirschberg, Julia Hockenmaier, Tengyu Ma, Andrew McCallum, Kathy McKeown, Rada Mihalcea, Tom Mitchell, Ray Mooney, Karthik Narasimhan, Graham Neubig, Christos Papadimitriou, Nanyun Peng, Drago Radev, Sasha Rush, Fei Sha, Yulia Tsvetkov, Luke Zettlemoyer and many others. These people are really a big part of the reasons that I love our research community so much, therefore I want to follow their paths and dedicate myself to an academic career. I hope to continue to contribute to our research community in the future.

A special thanks to Andrew Chi-Chih Yao for creating the Special Pilot CS Class where I did my undergraduate studies. I am super proud of being a part of the "Yao class" family.

I also thank Weizhu Chen, Qiang Yang and Haixun Wang, with them I received my very first research experience. With their support, I was very fortunate to have the opportunity to come to Stanford for my PhD.

I thank my parents: Zhi Chen and Hongmei Wang. Like most Chinese students in my generation, I am the only child of my family and I have a very close relationship with them — even if they are living 16 (or 15) hours ahead of me and I can only spare 2–3 weeks staying with them every year. My parents made me who I am today and I never know how to pay them back. I hope that they are at least a little proud of me for what I have been through so far.

Lastly, I would like to thank Huacheng for his love and support (we got married 4 months before this dissertation was submitted). I was fifteen when I first met Huacheng and we have been experiencing almost everything together since then: from high-school programming competitions, to our wonderful college time at Tsinghua University and we both made it to the Stanford CS PhD program in 2012. For over ten years in the past, he is not only my partner, my classmate, my best friend, but also the person I admire most, for his modesty, intelligence, concentration and hard work. Without him, I would not have come to Stanford. Without him, I would also not have taken the job at Princeton. I thank him for everything he has done for me.

To my parents and Huacheng, for their unconditional love.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Teaching machines to understand human language documents is one of the most elusive and long-standing challenges in Artificial Intelligence. Before we proceed, we must ask what it means to understand human language? Figure 1.1 demonstrates a children's story from the MCTEST dataset (Richardson et al., 2013), with simple vocabulary and grammar. To process such a passage of text, the NLP community has put decades of effort into solving different tasks for various aspects of text understanding, including:

(a) **part-of-speech tagging**. It requires our machines to understand that, for example, in the first sentence *Alyssa got to the beach after a long trip.*, *Alyssa* is a proper noun, *beach* and *trip* are common nouns, *got* is a verb in its past tense, *long* is an adjective, *after* is a preposition.

(b) **named entity recognition**. Our machines also should understand that *Alyssa*, *Ellen*, *Kristen* are the names of people in the story while *Charlotte*, *Atlanta* and *Miami* are the names of locations.

(c) **syntactic parsing**. To understand the meaning of each single sentence, our machines also need to understand the relationship between words, or the syntactical (grammatical) structure. Taking the first sentence *Alyssa got to the beach after a long trip.* as an example again, the machines should understand that *Alyssa* is the subject, and *beach* is

**Alyssa** got to the beach after a long trip. She's from Charlotte. She traveled from Atlanta. She's now in Miami. She went to Miami to visit some friends. But she wanted some time to herself at the beach, so she went there first. After going swimming and laying out, she went to her friend **Ellen**'s house. **Ellen** greeted **Alyssa** and they both had some lemonade to drink. **Alyssa** called her friends **Kristen** and **Rachel** to meet at **Ellen**'s house. The girls traded stories and caught up on their lives. It was a happy time for everyone. The girls went to a restaurant for dinner. The restaurant had a special on catfish. **Alyssa** enjoyed the restaurant's special. **Ellen** ordered a salad. **Kristen** had soup. **Rachel** had a steak. After eating, the ladies went back to **Ellen**'s house to have fun. They had lots of fun. They stayed the night because they were tired. **Alyssa** was happy to spend time with her friends again.

(a) **Question:** What city is Alyssa in?
**Answer**: Miami

(b) **Question**: What did Alyssa eat at the restaurant?
**Answer**: catfish

(c) **Question**: How many friends does Alyssa have in this story?
**Answer**: 3

Figure 1.1: A sample story and comprehension questions from the MCTEST dataset (Richardson et al., 2013).

the object of the verb *got*, while *after a long trip* as a whole is a prepositional phrase which describes a temporal relationship with the verb.

(d) **coreference resolution**. Furthermore, our machines even need to understand the interplay between sentences. For example, the mention *She* in the sentence *She's now in Miami* refers to *Alyssa* mentioned in the first sentence, while the mention *The girls* refers to *Alyssa, Ellen, Kristen and Rachel* in the previous sentences.

Is there a comprehensive evaluation that can test all these aspects and probe even deeper levels of understanding? We argue that the task of **reading comprehension** — answering comprehension questions over a passage of text — is a proper and important way to approach that. Just as we use reading comprehension tests to measure how well a human has understood a piece of text, we believe that it can play the same role for evaluating how well computer systems understand human language.

Let's take a closer look at some comprehension questions posed on the same passage (Figure 1.1):

(a) To answer the first question *What city is Alyssa in?*, our machines need to pick out the sentence *She's now in Miami.*, and resolve the *coreference resolution* problem that *She* refers to *Alyssa*, and then finally give the correct answer *Miami*.

(b) For the second question *What did Alyssa eat at the restaurant?*, they need to first locate the two sentences *The restaurant had a special on catfish.* and *Alyssa enjoyed the restaurant's special.* and understand the *special* that *Alyssa enjoyed* in the second sentence refers back to the first sentence. Based on the fact that *catfish* modifies *special*, the answer is hence *catfish*.

(c) The last question is especially challenging. To arrive at the correct answer, the machines have to keep track of all the names of people mentioned in the text and their relationships, perform some arithmetic reasoning (counting), and finally give the answer *3*.

As we can see, our computer systems have to understand many different aspects of text to answer these questions correctly. Since questions can be designed to query the aspects that we care about, *reading comprehension could be the most suitable task for evaluating language understanding*. This is a central theme of this thesis.

In this thesis, we study the problem of reading comprehension: how can we build computer systems to read a passage and answer these comprehension questions? In particular, we focus on **neural reading comprehension**, a class of reading comprehension models built using deep neural networks, which have been proven much more effective than non-neural, feature-based models.

The field of reading comprehension has a long history — as early as the 1970s, researchers already recognized that it is an important way to test the language understanding capabilities of computer programs (Lehnert, 1977). However, the field has been neglected for decades and only recently, it has received a great deal of attention and rapid progress has been made (see Figure 2.1 as an example), including our efforts that we will detail in this thesis. The recent success of reading comprehension can be attributed to two reasons:

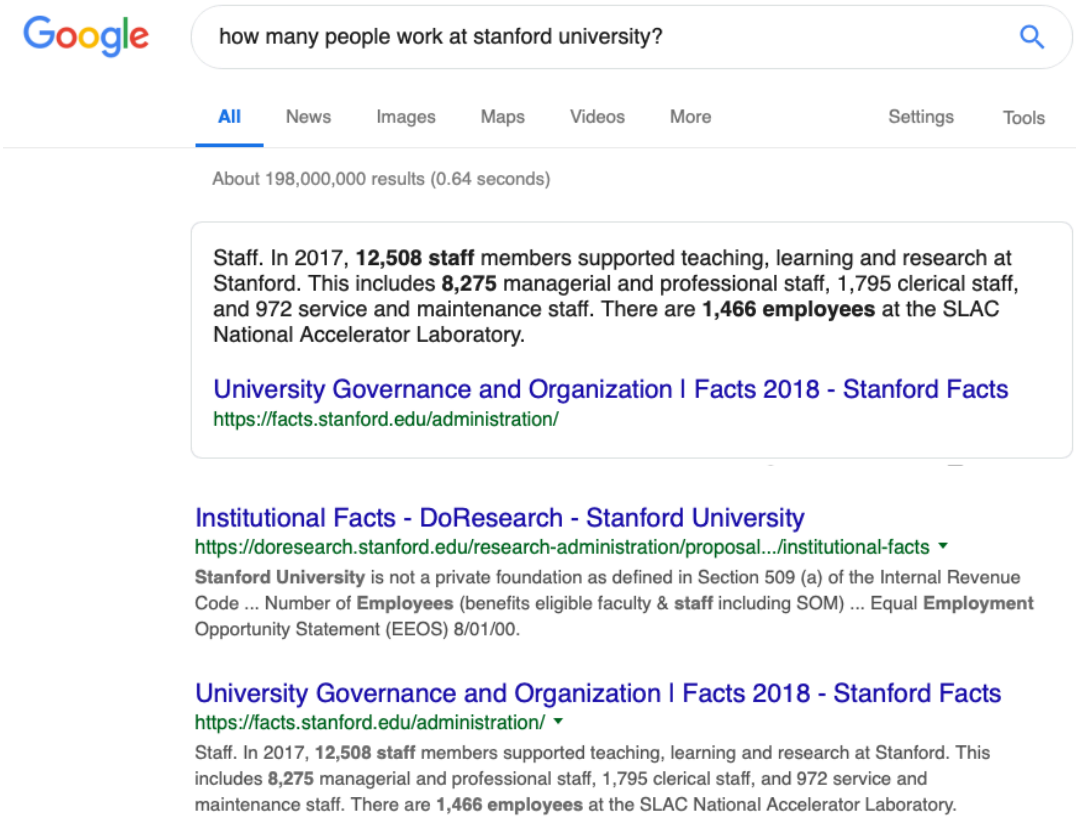



Figure 1.2: A search result on GOOGLE. It not only returns a list of search documents but gives more precise answers within the documents.

1) the creation of large-scale supervised datasets in the form of (passage, question, answer) triples; 2) the development of neural reading comprehension models.

In this thesis, we will cover the essence of modern neural reading comprehension: the formulation of the problem, the building blocks and key ingredients of these systems, and understanding of where current neural reading comprehension systems can excel and where they still lag behind.

The second central theme of the thesis is that we deeply believe that, if we can build high-performing reading comprehension systems, *they would be a crucial technology for applications such as question answering and dialogue systems*. Indeed, these language technologies are already very relevant to our daily lives. For example, today if we enter a search query into GOOGLE "How many people work at Stanford University?" (Figure 1.2),

GOOGLE not only returns a list of search documents, but also attempts to read these Web documents and finally highlight the most plausible answers and display them at the top of the search results. We believe this is exactly where reading comprehension can help and thus can facilitate more intelligent search engines. Additionally, with the development of digital personal assistants such as Amazon's ALEXA, Apple's SIRI, GOOGLE ASSISTANT or Microsoft's CORTANA, more and more users engage with these devices by having conversations and asking informational questions.[1] We believe that building machines which are able to read and comprehend text will also greatly improve the capabilities of these personal assistants.

Therefore, in this thesis, we are also interested in how we can build practical applications from the recent success of neural reading comprehension. We explore two research directions which employ neural reading comprehension as a key component:

**Open-domain question answering** combines the challenges from both information retrieval and reading comprehension and aims to answer general questions from either the Web or a large encyclopedia (e.g., Wikipedia).

**Conversational question answering** combines the challenges from dialogue and reading comprehension, and tackles the problem of multi-turn question answering over a passage of text, like how users would engage with conversational agents. Figure 1.3 demonstrates an example from our COQA dataset (Reddy et al., 2019). In this example, a person can ask a series of interconnected questions based on the content of a CNN article.

## 1.2 Thesis Outline

Following the two central themes that we just discussed, this thesis consists of two parts — PART I NEURAL READING COMPREHENSION: FOUNDATIONS and PART II NEURAL READING COMPREHENSION: APPLICATIONS.

[1] A recent study https://www.stonetemple.com/digital-personal-assistants-study/ reported that asking general questions is indeed the number one use for such digital personal assistants.

Fort Lauderdale, Florida (CNN) – Just taking a sip of water or walking to the bathroom is excruciatingly painful for 15-year-old Michael Brewer, who was burned over 65 percent of his body after being set on fire, allegedly by a group of teenagers.
"It hurts my heart to see him in pain, but it enlightens at the same time to know my son is strong enough to make it through on a daily basis," his mother, Valerie Brewer, told CNN on Wednesday.
Brewer and her husband, Michael Brewer, Sr., spoke to CNN's Tony Harris, a day after a 13-year-old boy who witnessed last month's attack publicly read a written statement:
"I want to express my deepest sympathy to Mikey and his family," Jeremy Jarvis said."I will pray for Mikey to grow stronger every day and for Mikey's speedy recovery."
Jarvis' older brother has been charged in the October 12 attack in Deerfield Beach, Florida. When asked about the teen's statement, Valerie Brewer – who knows the Jarvis family – said she "can't focus on that."
"I would really like to stay away from that because that brings negative energy to me and I don't need that right now," she said.
Her son remains in guarded condition at the University of Miami's Jackson Memorial Hospital Burn Center. He suffered second- and third-degree burns over about two-thirds of his body, according to the hospital's associate director, Dr. Carl Schulman.
The teen faces a lifelong recovery from his injuries, Schulman told CNN's Harris.

$Q_1$: What is the subject of the story?
$A_1$: Michael Brewer

$Q_2$: What happened to him?
$A_2$: He was burned

$Q_3$: How badly?
$A_3$: Over 65% of his body

$Q_4$: Do we know who caused the burns?
$A_4$: Yes

Figure 1.3: A conversation from CoQA based on an CNN article.

PART I focuses on the task of reading comprehension, with an emphasis on close reading of a short paragraph so that computer systems are able to answer comprehension questions.

In Chapter 2, we first give an overview of the history and recent development of the field of reading comprehension. Next we formally define the problem formulation and its

main categories. We then briefly discuss the differences of reading comprehension and general question answering. Finally, we argue that the recent success of neural reading comprehension is driven by both large-scale datasets and neural models.

In Chapter 3, we present the family of neural reading comprehension models. We begin with describing non-neural, feature-based classifiers, and discuss how they differ from the end-to-end neural approaches. We then introduce a neural approach that we proposed named THE STANFORD ATTENTIVE READER and we describe its basic building blocks and extensions. We present experimental results on two representative reading comprehension datasets: CNN/DAILY MAIL and SQUAD, and more importantly, we conduct an in-depth analysis of the neural models to understand better what these models have actually learned. Finally, we summarize recent advances of neural reading comprehension models in different aspects. This chapter is based on our works (Chen et al., 2016) and (Chen et al., 2017).

In Chapter 4, we discuss future work and open questions in this field. We first examine error cases of existing models despite their high accuracies on the current benchmarks. We then discuss future directions, in terms of both the datasets and the models. Finally, we review several important research questions in this field, which still remain as open questions and yet to be answered in the future.

PART II views reading comprehension as an important building block for practical applications such as question answering systems and conversational agents. Detailedly,

In Chapter 5, we address the problem of open domain question answering as an application of reading comprehension. We discuss how we can combine a high-performing neural reading comprehension system and effective information retrieval techniques, to build a new generation of open-domain question answering systems. We describe a system we built named DRQA: its key components and how we create training data for it, and we then present a comprehensive evaluation on multiple question answering benchmarks. We discuss its current limitations and the future work in the end. This chapter is based on our work (Chen et al., 2017).

In Chapter 6, we study the problem of conversational question answering, where a machine has to understand a text passage and answer a series of questions that appear in a conversation. We first briefly review the literature on dialogue and argue that conversational question answering is the key to building information-seeking dialogue agents. We introduce CoQA: a novel dataset for building **Co**nversational **Q**uestion **A**nswering systems, comprising 127k questions with answers, obtained from 8k conversations about text passages. We analyze the dataset in depth and build several competitive models on top of conversational and neural reading comprehension models and present the experimental results. We finally discuss the future work in this area. This chapter is based on our work (Reddy et al., 2019).

We will finally conclude in Chapter 7.

## 1.3 Contributions

The contributions of this thesis are summarized as follows:

- We were among the first to research neural reading comprehension. In particular, we proposed the STANFORD ATTENTIVE READER model, which has demonstrated superior performance on various modern reading comprehension tasks.
- We made the effort to understand better what neural reading comprehension models have actually learned, and what depth of language understanding is needed to solve current tasks. We concluded that neural models are better at learning lexical matches and paraphrases compared to conventional feature-based classifiers, while the reasoning capabilities of existing systems are still rather limited.
- We pioneered the research direction of employing neural reading comprehension as a core component of open domain question answering, and examined how to generalize the model for this case. In particular, we implemented this idea in the DRQA system, a large-scale, factoid question answering system over English Wikipedia.
- Finally, we set out to tackle the conversational question answering problem, in which computer systems need to answer comprehension questions in a dialogue context, so

each question needs to be understood with its conversation history. To tackle this, we proposed the CoQA challenge and also built neural reading comprehension models adapted to this problem. We believe that this is a first but important step to building conversational QA agents.

# Part I

# Neural Reading Comprehension: Foundations

# Chapter 2

# An Overview of Reading Comprehension

> When a person understands a story, he can demonstrate his understanding by answering questions about the story. Since questions can be devised to query any aspect of text comprehension, the ability to answer questions is the strongest possible demonstration of understanding. If a computer is said to understand a story, we must demand of the computer the same demonstration of understanding that we require of people. Until such demands are met, we have no way of evaluating text understanding programs.
>
> Wendy Lehnert, 1977

In this chapter, we aim to provide readers with an overview of reading comprehension. We begin with the history of reading comprehension (Section 2.1), from the early systems developed in the 1970s, to the attempts to build machine learning models for this task, to the more recent resurgence of neural (deep learning) approaches. This field has been completely reshaped by neural reading comprehension, and the progress is very exciting.

We then formally define the reading comprehension task as a supervised learning problem in Section 2.2 and describe four different categories based on the answer type. We end by discussing their evaluation metrics.

Next we discuss briefly how reading comprehension differs from question answering,

especially in their final goals (Section 2.3). Finally, we discuss that how the interplay of large-scale datasets and neural models contributes to the development of modern reading comprehension in Section 2.4.

## 2.1 History

### 2.1.1 Early Systems

The history of building automated reading comprehension systems dates back to over forty years ago. In the 1970s, researchers already recognized the importance of reading comprehension as an appropriate way of testing the language understanding abilities of computer programs.

One of the most notable early works is the QUALM system detailed in Lehnert (1977). Built on top of the framework of scripts and plans as devices for modeling human story comprehension (Schank and Abelson, 1977), Lehnert (1977) devised a theory of question answering and focused on pragmatic issues and the importance of the context of the story in responding to questions. This early work set a strong vision for language understanding, but the actual systems built at that time were very small and limited to hand-coded scripts, and difficult to generalize to broader domains.

Due to the complexity of the problem, this line of research was mostly neglected in the 1980s and 1990s.[1] In the late 1990s, there was some small revival of interest, following the creation of a reading comprehension dataset by Hirschman et al. (1999) and a subsequent Workshop on Reading Comprehension Tests as Evaluation for Computer-based Understanding Systems at ANLP/NAACL 2000. The dataset consists of 60 stories for development and 60 stories for testing of 3rd to 6th grade material, followed by short-answer *who*, *what*, *when*, *where* and *why* questions. It only requires systems to return a sentence which contains the right answer. The systems developed at this stage were mostly rule-based bag-of-words approaches with shallow linguistic processing such as stemming, semantic class identification and pronoun resolution in the DEEP READ system (Hirschman

[1]There has been a large body of work in story comprehension developed within the psychology community, see (Kintsch, 1998).

et al., 1999), or manually generated rules based on lexical and semantic correspondence in the QUARC system (Riloff and Thelen, 2000) or their combinations (Charniak et al., 2000). These systems achieved $30\%$–$40\%$ accuracy on retrieving the correct sentence.

### 2.1.2 Machine Learning Approaches

Between 2013 and 2015, there were remarkable efforts of formulating reading comprehension as a *supervised learning* problem: researchers collected human-labeled training examples in the form of (passage, question, answer) triples, with the hope that we can train statistical models which learn to map a passage and question pair into their corresponding answer: $f : (\text{passage}, \text{question}) \longrightarrow \text{answer}$.

Two notable datasets during this period are MCTEST (Richardson et al., 2013) and PROCESSBANK (Berant et al., 2014). MCTEST collects 660 fictional stories, with 4 multiple choice questions per story (each question comes with 4 hypothetical answers and one of them is correct) (Table 2.1 (b)). PROCESSBANK is designed to answer binary-choice questions in a paragraph describing a biological process and requires an understanding of the relations between entities and events in the process. The dataset comprises 585 questions spread over the 200 paragraphs.

In the original MCTEST paper, Richardson et al. (2013) proposed several rule-based baselines without leveraging any training data. One is a heuristic sliding window approach, which measures the weighted word overlap/distance information between words in the question, the answer and the sliding window; the other is to run an off-the-shelf textual entailment system by converting each question-answer pair into a statement. This dataset later inspired a strand of machine learning models (Sachan et al., 2015; Narasimhan and Barzilay, 2015; Wang et al., 2015). These models were mostly built on top of a simple max-margin learning framework with a rich set of hand-engineered linguistic features, including syntactic dependencies, semantic frames, coreference resolution, discourse relations and word embeddings. The performance was improved modestly from 63% to around 70% on the MC500 portion. On the PROCESSBANK dataset, Berant et al. (2014) proposed a statistical model which learns to predict the process structure first and then maps the question to formal queries that can be executed against the structure. Similarly, the model

| | |
|---|---|
| (a) | **CNN/Daily Mail** (cloze style)<br>**passage**: ( @entity4 ) if you feel a ripple in the force today , it may be the news that the official @entity6 is getting its first gay character . according to the sci-fi website @entity9 , the upcoming novel " @entity11 " will feature a capable but flawed @entity13 official named @entity14 who " also happens to be a lesbian . " the character is the first gay figure in the official @entity6 – the movies , television shows , comics and books approved by @entity6 franchise owner @entity22 – according to @entity24 , editor of " @entity6 " books at @entity28 imprint @entity26 .<br>**question**: characters in " _____ " movies have gradually become more diverse<br>**answer**: @entity6 |
| (b) | **MCTest** (multiple choice)<br>**passage**: Once upon a time, there was a cowgirl named Clementine. Orange was her favorite color. Her favorite food was the strawberry. She really liked her Blackberry phone, which allowed her to call her friends and family when out on the range. One day Clementine thought she needed a new pair of boots, so she went to the mall. Before Clementine went inside the mall, she smoked a cigarette. Then she got a new pair of boots. She couldn't choose between brown and red. Finally she chose red, which the seller really liked. Once she got home, she found that her red boots didn't match her blue cowgirl clothes, so she knew she needed to return them. She traded them for a brown pair. While she was there, she also bought a pretzel from Auntie Anne's.<br>**question**: What did the cowgirl do before buying new boots?<br>**hypothesized answers**: A. She ate an orange B. She ate a strawberry C. She called her friend D. She smoked a cigarette<br>**answer**: D. She smoked a cigarette |
| (c) | **SQuAD** (span prediction) |

**passage**: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.
**question**: Which NFL team won Super Bowl 50?
**answer**: Denver Broncos

(d) **NarrativeQA** (free-form text)
**passage**: . . . In the eyes of the city, they are now considered frauds. Five years later, Ray owns an occult bookstore and works as an unpopular children s entertainer with Winston; Egon has returned to Columbia University to conduct experiments into human emotion; and Peter hosts a pseudo-psychic television show. Peter's former girlfriend Dana Barrett has had a son, Oscar, with a violinist whom she married then divorced when he received an offer to join the London Symphony Orchestra.. . .
**question**: How is Oscar related to Dana?
**answer**: He is her son

Table 2.1: A few examples from representative reading comprehension datasets: (a) CNN/DAILY MAIL (Hermann et al., 2015), (b) MCTEST (Richardson et al., 2013), (c) SQUAD (Rajpurkar et al., 2016) and (d) NARRATIVEQA (Kočiskỳ et al., 2018).

incorporates a large set of manual features,[2] and eventually obtains 66.7% accuracy on the binary classification task.

These machine learning models have achieved modest progress compared to earlier rule-based heuristic methods. However, their improvements are still rather limited and

[2]See https://nlp.stanford.edu/pubs/berant-srikumar-manning-emnlp14-supp.pdf.

their weaknesses are summarized as follows:

- These models relied heavily on existing linguistic tools such as dependency parsers and semantic role labeling (SRL) systems. However, these linguistic representation tasks are far from solved and off-the-shelf tools are often trained from one single domain (e.g., newswire articles) and suffer from generalization problems in practical use. Therefore, leveraging existing linguistic annotations as features sometimes adds noise in these feature-based machine learning models and the situation gets worse for higher level annotations (e.g., discourse relations vs. part-of-speech tagging).

- Simulating human-level comprehension is an elusive challenge and it is always the case that it is difficult to construct effective features from current linguistic representations. For example, for the third question in Figure 1.1: *How many friends does Alyssa have in this story?*, it is impossible to construct an effective feature when the evidence is spread over the passage.

- Although it is inspiring that we can train models from human-labeled reading comprehension examples, these datasets are still too small to support expressive statistical models. For example, the English Penn Treebank dataset for training dependency parsers consists of 39,832 examples, while in MCTEST, only 1,480 examples are used for training — let alone reading comprehension which, as a comprehensive language understanding task, is more complex and requires different reasoning capabilities.

### 2.1.3 A Resurgence: The Deep Learning Era

A turning point for this field came in 2015. The DeepMind researchers Hermann et al. (2015) proposed a novel and cheap solution for creating large-scale supervised training data for learning reading comprehension models. They also proposed a neural network model — an attention-based LSTM model named THE ATTENTIVE READER — and demonstrated that it outperformed symbolic NLP approaches by a large margin. In their experiments, the ATTENTIVE READER achieved 63.8% accuracy while symbolic NLP systems obtained 50.9% at most on the CNN dataset. The idea of the data creation is as follows: CNN and

Daily Mail are accompanied by a number of bullet points, summarizing aspects of the information contained in the article. They take a news article as the passage and convert one of its bullet points as a cloze style question by replacing one entity at at time with a placeholder, and the answer is this replaced entity. In order to ensure that systems approaching this task need to genuinely understand the passage, rather than using world knowledge or a language model to answer questions, they run entity recognition and coreference resolution systems and replace all the entity mentions in each coreference chain by an abstract entity marker e.g., *@entity6* (see an example in Table 2.1 (a)). As a result, nearly 1 million data examples were collected at almost no cost.

Taking a step further, our work (Chen et al., 2016) investigated this first-ever large reading comprehension dataset and demonstrated that a simple, carefully designed neural network model (Section 3.2) is able to push the performance to 72.4% on the CNN dataset, another 8.6% absolute improvement. More importantly, we justified that the neural network models are better at recognizing lexical matches and paraphrases compared to conventional feature-based classifiers. However, although this semi-synthetic dataset provides a promising avenue for training effective statistical models, we concluded that the dataset appears to be noisy due to its method of data creation and coreference errors and is limited for driving further progress.

To address these limitations, Rajpurkar et al. (2016) collected a new dataset named THE STANFORD QUESTION ANSWERING DATASET (SQUAD). The dataset contains 107,785 question-answer pairs on 536 Wikipedia articles and the questions were posed by crowdworkers, and the answer to every question is a span of text from the corresponding reading passage (Table 2.1 (c)). SQUAD was the first large-scale reading comprehension dataset with natural questions. Thanks to its high quality and reliable automatic evaluation, this dataset has spurred tremendous interest in the NLP community and become the central benchmark in this field. It in turn inspired a wide array of new reading comprehension models (Wang and Jiang, 2017; Seo et al., 2017; Chen et al., 2017; Wang et al., 2017; Yu et al., 2018) and the progress has been rapid — as of Oct 2018, the best-performing single system achieved an F1 score of 91.8% (Devlin et al., 2018) which already exceeds the estimated human performance of 91.2%, while a feature-based classifier built by the original authors in 2016 only obtained an F1 of 51.0%, as shown in Figure 2.1.

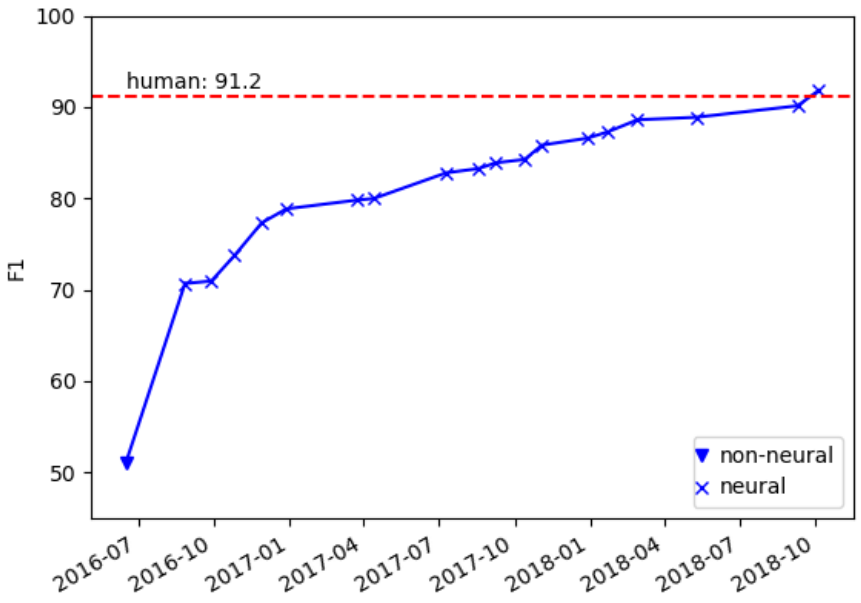



Figure 2.1: The progress on SQuAD 1.1 (single model) since the dataset was released in June 2016. The data points are taken from the leaderboard at http://stanford-qa.com/.

All the current top-performing systems on SQuAD are built on *end-to-end neural networks*, or *deep learning* models. These models usually start off from the idea of representing every single word in the passage and question as a dense vector (e.g., 300 dimensions), passing through several modeling or interaction layers, and finally making predictions. All the parameters can be optimized jointly using the gradient descent algorithm or its variants. This class of models can be referred to as *neural reading comprehension* and we will describe it in detail in Chapter 3. Differing from feature-based classifiers, neural reading comprehension models have several great advantages:

- They don't rely on any downstream linguistic features (e.g., dependency parsing or coreference resolution) and all the features are learned on their own in one unified end-to-end framework. This can avoid noise in linguistic annotations while also providing great flexibility in the space of useful features.

- Conventional symbolic NLP systems suffer from one severe problem: features are usually very sparse and generalize poorly. For example, to answer a question "*How many individual libraries **make up** the main school library?*" from a passage "... *Harvard Library, which is the world's largest academic and private library system, **comprising** 79 individual libraries with over 18 million volumes.*", a system has to learn the correspondence between *comprising* and *make up* based on indicator features such as:

  $$\text{pw}_i = \text{comprising} \wedge \text{qw}_j = \text{make} \wedge \text{qw}_{j+1} = \text{up}.$$

  There is insufficient data to correctly weight most such features. It is a common problem in all non-neural NLP models. Making use of low-dimensional, dense word embeddings can effectively alleviate sparsity by sharing statistical strength between similar words.

- They are relieved from the labor of constructing a large set of manual features. Therefore, neural models are conceptually simpler and the focus can move to the design of neural architectures instead. Thanks to the development of modern deep learning frameworks such as TENSORFLOW and PYTORCH, great progress has been made, and now it is fast and easy to develop new models.

There is no doubt that achieving human-performance on SQUAD is incredible and arguably one of the biggest results we have seen in the NLP community in the past few years. Nevertheless, solving the SQUAD task isn't equivalent to solving machine reading comprehension. We need to acknowledge that SQuAD is restricted in that questions must be answered using a single span in the passage and most SQuAD examples are fairly simple and don't really need complex reasoning.

The field has been further evolving. Following the theme of creating large-scale and more challenging reading comprehension datasets, a multitude of datasets have been collected recently: TRIVIAQA (Joshi et al., 2017), RACE (Lai et al., 2017), QANGAROO (Welbl et al., 2018), NARRATIVEQA (Kočiskỳ et al., 2018), MULTIRC (Khashabi et al., 2018), SQuAD 2.0 (Rajpurkar et al., 2018), HOTPOTQA (Yang et al., 2018) and many others. These datasets were collected from a variety of sources (Wikipedia, newswire articles,

fictional stories or other Web resources) and constructed in very different ways and they aim to tackle many challenges that haven't been addressed before — questions which are curated independent of the passages, questions which require multiple sentences or even multiple documents to answer, questions based on long documents like a full book, or questions which are not answerable from the passage. At the time of this writing, most of these datasets have not been solved yet and there remains a large gap between state-of-the-art methods and human performance levels. Reading comprehension has become one of the most active fields in NLP today and there are still many open questions to solve. We will discuss the recent development of reading comprehension datasets in more detail in Section 4.2.

## 2.2 Task Definition

### 2.2.1 Problem Formulation

The task of reading comprehension can be formulated as a supervised learning problem: given a collection of training examples $\{(p_i, q_i, a_i)\}_{i=1}^{n}$, the goal is to learn a predictor $f$ which takes a passage of text $p$ and a corresponding question $q$ as inputs and gives the answer $a$ as output.

$$f : (p, q) \longrightarrow a \tag{2.1}$$

Let $p = (p_1, p_2, \ldots, p_{l_p})$ and $q = (q_1, q_2, \ldots, q_{l_q})$[3] where $l_p$ and $l_q$ denote the length of the passage and the question, $p_i \in \mathcal{V}$ for $i = 1, \ldots, l_p$ and $q_i \in \mathcal{V}$ for $i = 1, \ldots, l_q$ where $\mathcal{V}$ is a pre-defined vocabulary. Here we only consider the passage $p$ as a short paragraph represented as a sequence of $l_p$ words. It is straightforward to extend it to a multi-paragraph setting (Clark and Gardner, 2018) where $p$ is a set of paragraphs or decompose it into smaller linguistic units such as sentences.[4]

Depending on the answer type, the answer $a$ can take very different forms. Generally, we can divide existing reading comprehension tasks into four categories:

[3] A preprocessing step of tokenization is usually required on most current reading comprehension datasets.

[4] There have been some efforts (e.g., (Xie and Xing, 2017)) which model the paragraph as a sequence of sentences, but there is no clear evidence that it outperforms methods that treat the whole paragraph as a long sequence at this point.

**Cloze style.** The question contains a placeholder. For instance,

> Tottenham manager Juande Ramos has hinted he will allow _____ to leave if the Bulgaria striker makes it clear he is unhappy.

In these tasks, the systems must guess which word or entity completes the sentence (question), based on the passage, and the answer $a$ is either chosen from a pre-defined set of choices $\mathcal{A}$ or the full vocabulary $\mathcal{V}$. For example, in the WHO-DID-WHAT dataset (Onishi et al., 2016), $a$ must be one of the person named entities in the passage and $|\mathcal{A}| = 3.5$ on average.

**Multiple choice.** In this category, the correct answer is chosen from $k$ hypothesized answers (e.g., $k = 4$):

$$\mathcal{A} = \{a_1, \ldots, a_k\} \text{ where } a_k = (a_{k,1}, a_{k,2} \ldots, a_{k,l_{a,k}}), a_{k,i} \in \mathcal{V},$$

can be a word, a phrase or a sentence. One of the hypothesized answers is correct and thus $a$ must be chosen from $\{a_1, \ldots, a_k\}$.

**Span prediction.** This category is also referred to as *extractive question answering* and the answer $a$ must be a single span in the passage. Therefore, $a$ can be represented as $(a_{start}, a_{end})$ where $1 \leq a_{start} \leq a_{end} \leq l_p$, and the answer string corresponds to $p_{a_{start}}, \ldots, p_{a_{end}}$.

**Free-form answer.** The last category allows the answer to be any free-text form (i.e., a word sequence of arbitrary length), formally, $a \in \mathcal{V}^*$.

Table 2.1 gives an example in each of the categories from four representative datasets: CNN/DAILY MAIL (Hermann et al., 2015) (cloze style), MCTEST (Richardson et al., 2013) (multiple choice), SQUAD (Rajpurkar et al., 2016) (span prediction) and NARRATIVEQA (Kočiskỳ et al., 2018) (free-form answer).

### 2.2.2 Evaluation

We have formally defined the four different categories of reading comprehension tasks, next we discuss their evaluation metrics.

For **multiple choice** or **cloze style** tasks, it is quite straightforward to measure the accuracy: the percentage of questions for which systems give the exactly correct answer, as the answer is chosen from a small set of hypothesized answers.

For **span prediction** tasks, we need to compare the predicted answer string to the ground truth. Typically, we use the two evaluation metrics proposed in Rajpurkar et al. (2016), which measure both exact match and partial scores:

- **Exact match (EM)** assigns a full credit $1.0$ if the predicted answer is equal to the gold answer and $0.0$ otherwise.
- **F1 score** computes the average word overlap between predicted and gold answers. The prediction and the gold answer are treated as a bag of tokens and a token-level F1 score is calculated as:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Following Rajpurkar et al. (2016), all punctuation is ignored in the evaluation and for English, articles *a*, *an*, and *the* are also ignored.

To make the evaluation more reliable, it is also common to collect multiple gold answers to each question. Therefore, the exact match score is only required to match any of the gold answers while the F1 score is to compute the maximum over all of the gold answers and then averaged over all of the questions.

Lastly, for the **free-form answer** reading comprehension tasks, there isn't a consensus yet on what is the most ideal evaluation metric. A common way is to use standard evaluation metrics in natural language generation (NLG) tasks such as machine translation or summarization, including BLEU (Papineni et al., 2002), Meteor (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004).

## 2.3 Reading Comprehension vs. Question Answering

There is a close relationship between reading comprehension and question answering. We can view reading comprehension as an instance of question answering because it is essentially a question answering problem over a short passage of text. Nevertheless, although reading comprehension and general question answering share many common characteristics in problem formulation, approaches and evaluation, we think they emphasize different thing as their final goals:

- The ultimate goal of question answering is to build computer systems which are able to automatically *answer questions* posed by humans, no matter what sort of resources they depend on. These resources can be structured knowledge bases, unstructured text collections (encyclopedias, dictionaries, newswire articles and general Web documents), semi-structured tables or even other modalities. Towards the better performance of QA systems, a lot of efforts have been put into (1) how to search and identify relevant resources, (2) how to integrate answers from different pieces of information, or even (3) to study what types of questions humans usually ask in the real world.

- However, reading comprehension puts more emphasis on *text understanding* with answering questions regarded as a way to measure language understanding. Therefore it requires a deep understanding of the given passage in order to answer the question. Due to this key difference, early works in this field mostly focused on fictional stories (Lehnert, 1977) (later extended to Wikipedia or Web documents), so all the information to answer comprehension questions comes from the passage itself instead of any world knowledge. The questions are also specifically devised to test different aspects of text comprehension. This distinction is akin to what questions people usually ask on search engines versus what sorts of questions are usually posed in human reading comprehension tests.

Similarly, early work (Mitchell et al., 2009) used the terms **micro-reading** and **macro-reading** to differentiate these two scenarios. Micro-reading focuses on reading a single text document and aims to extract the full information content of that document (similar to our

reading comprehension setting), while macro-reading takes a large text collection (e.g., the Web) as input and extracts a large collection of facts expressed in the text, without requiring that every single fact is extracted. Macro-reading can effectively leverage the *redundancy* of information across documents by focusing on analyzing simple wordings of the fact in the text, while macro-reading has to investigate deeper level of language understanding.

This thesis mostly focuses on reading comprehension. In Chapter 5, we will come back to more general question answering problems, discuss its related work and also demonstrate that reading comprehension can be also helpful in building question answering systems.

## 2.4 Datasets and Models

As seen in Section 2.1.3, the recent success of reading comprehension has been mainly driven by two key components: *large-scale reading comprehension datasets* and *end-to-end neural reading comprehension models*. They work together to advance the field and push the boundaries of building better reading comprehension systems:

On the one hand, the creation of large-scale reading comprehension datasets has made it possible to train neural models, while demonstrating their competitiveness over symbolic NLP systems. The availability of these datasets further attracted a lot of attention in our research community and inspired a series of modeling innovations. Tremendous progress has been made thanks to all these efforts.

On the other hand, understanding the performance of existing models further helps identity the limitations of existing datasets. This motivates us to seek better ways to construct more challenging datasets, towards the ultimate goal of machine comprehension of text.

Figure 2.2 shows a timeline of the recent development of key datasets and models since 2016. As is seen, although it has been only 3 years, the field has been moving strikingly fast. The innovations in building better datasets and more effective models have occurred alternately and both contributed to the development of the field. In the future, we believe it will be equally important to continue to develop both components.

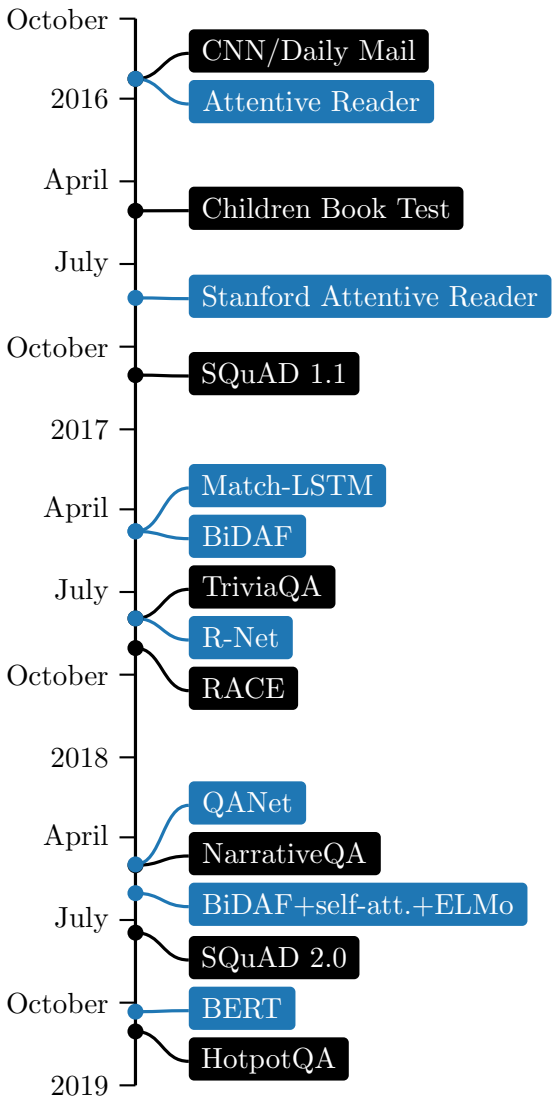



Figure 2.2: The recent development of datasets (black) and models (blue) in neural reading comprehension. For the timeline, we use the date that the corresponding papers were published, except BERT (Devlin et al., 2018).

In the next chapter, we will mainly focus on the modeling aspect, using the two representative datasets that we described earlier: CNN/DAILY MAIL and SQUAD. In Chapter 4, we will discuss more about the advances and future work, for both datasets and models.

# Chapter 3

# Neural Reading Comprehension Models

In this chapter, we will cover the essence of neural network models: from the basic building blocks, to more recent advances.

Before delving into the details of neural models, we give a brief introduction to non-neural, feature-based models for reading comprehension in Section 3.1. In particular, we describe a model that we built in Chen et al. (2016). We hope this will give readers a better sense about how these two approaches differ fundamentally.

In Section 3.2, we present a neural approach to reading comprehension called THE STANFORD ATTENTIVE READER, which we first proposed in Chen et al. (2016) for the cloze style reading comprehension tasks, and then later adapted to the span prediction problems (Chen et al., 2017) for SQUAD. We first briefly review the basic building blocks of modern neural NLP models, and then describe how our model is built on top of them. We discuss its extensions to the other types of reading comprehension problems in the end.

Next we present the empirical results of our model on the CNN/DAILY MAIL and the SQUAD datasets, and provide more implementation details in Section 3.3. We further conduct careful error analyses to help us better understand: 1) which components are most important for final performance; 2) where the neural models excel compared to non-neural feature-based models empirically.

Finally, we summarize recent advances in neural reading comprehension in Section 3.4.

## 3.1 Previous Approaches: Feature-based Models

We first describe a strong feature-based model that we built in Chen et al. (2016) for cloze style problems, in particular, the CNN/DAILY MAIL dataset (Hermann et al., 2015). We will then discuss similar models built for multiple choice and span prediction problems.

For the cloze style problems, the task is formulated as predicting the correct entity $a \in \mathcal{E}$ that can fill in the blank of the question $q$ based on reading the passage $p$ (one example can be found in Table 2.1), where $\mathcal{E}$ denotes the candidate set of entities. Conventional linear, feature-based classifiers usually need to construct a feature vector $f_{p,q}(e) \in \mathbb{R}^d$ for each candidate entity $e \in \mathcal{E}$, and to learn a weight vector $\mathbf{w} \in \mathbb{R}^d$ such that the correct answer $a$ is expected to rank higher than all other candidate entities:

$$\mathbf{w}^\intercal f_{p,q}(a) > \mathbf{w}^\intercal f_{p,q}(e), \forall e \in \mathcal{E} \setminus \{a\}, \tag{3.1}$$

After all the feature vectors are constructed for each entity $e$, we can then apply any popular machine learning algorithms (e.g., logistic regression or SVM). In Chen et al. (2016), we chose to use LAMBDAMART (Wu et al., 2010), as it is naturally a ranking problem and forests of boosted decision trees have been very successful lately.

The key question left is how can we build useful feature vectors from the passage $p$, the question $q$ and every entity $e$? Table 3.1 lists 8 sets of features that we proposed for the CNN/DAILY MAIL task. As shown in the table, these features are well designed and characterize the information of the entity (e.g., frequency, position and whether it is a question/passage word) and how they are aligned with the passage/question (e.g., co-occurrence, distance, linear and syntactic matching). Some features (#6 and #8) also rely on linguistic tools such as dependency parsing and part-of-speech tagging (deciding whether a word is a verb or not). Generally speaking, for non-neural models, how to construct a useful set of features always remains as a challenge. Useful features need to be informational and well-tailored to specific tasks, while not too sparse to generalize well from the training set. We have argued before in Sec 2.1.2 that this is a common problem in most of the feature-based models. Also, using the off-the-shelf linguistic tools makes the model more expensive and their final performance depends on the the accuracy level of these annotations.

| # | Feature |
|---|---|
| 1 | Whether entity $e$ occurs in the passage. |
| 2 | Whether entity $e$ occurs in the question. |
| 3 | The **frequency** of entity $e$ in the passage. |
| 4 | The **first position** of occurrence of entity $e$ in the passage. |
| 5 | **Word distance**: we align the placeholder with each occurrence of entity $e$, and compute the average minimum distance of each non-stop question word from the entity in the passage. |
| 6 | **Sentence co-occurrence**: whether entity $e$ co-occurs with another entity or verb that appears in the question, in some sentence of the passage. |
| 7 | **$n$-gram exact match**: whether there is an exact match between the text surrounding the placeholder and the text surrounding entity $e$. We have features for all combinations of matching left and/or right one or two words. |
| 8 | **Dependency parse match**: we dependency parse both the question and all the sentences in the passage, and extract an indicator feature of whether $w \xrightarrow{r}$ @placeholder and $w \xrightarrow{r} e$ are both found; similar features are constructed for @placeholder $\xrightarrow{r} w$ and $e \xrightarrow{r} w$. |

Table 3.1: Features used in our entity-centric classifier in Chen et al. (2016).

Rajpurkar et al. (2016) and Joshi et al. (2017) also attempted to build feature-based models for the SQuAD and TriviaQA datasets respectively. The models are similar in spirit to ours, except that for these span prediction tasks, they need to first decide on a set of possible answers. For SQuAD, Rajpurkar et al. (2016) consider all constituents in parses generated by Stanford CoreNLP (Manning et al., 2014) as candidate answers; while for TriviaQA, Joshi et al. (2017) consider all $n$-gram ($1 \leq n \leq 5$) that occurs in the sentences which contain at least one word in common with the question. They also tried to add more lexicalized features and labels from constituency parses. Other attempts have been made for multiple choice problems such as (Wang et al., 2015) for the MCTest dataset and a rich set of features have been used including semantic frames, word embeddings and coreference resolution.

We will demonstrate the empirical results of these feature-based classifiers and compare them to the neural models in Section 3.3.

## 3.2 A Neural Approach: The Stanford Attentive Reader

### 3.2.1 Preliminaries

In the following, we outline a minimal set of elements and the key ideas which form the basis of modern neural NLP models. For more details, we refer readers to (Cho, 2015; Goldberg, 2017).

#### Word embeddings

The first key idea is to represent words as low-dimensional (e.g., 300), real-valued vectors. Before the deep learning age, it was common to represent a word as an index into the vocabulary, which is a notational variant of using one-hot word vectors: each word is represented as a high-dimensional, sparse vector where only one entry of that word is 1 and all other entires are 0's:

$$\mathbf{v}_{\text{car}} = [0, 0, \ldots, 0, 0, 1, 0, \ldots, 0]^\intercal$$
$$\mathbf{v}_{\text{vehicle}} = [0, 1, \ldots, 0, 0, 0, 0, \ldots, 0]^\intercal$$

The biggest problem with these sparse vectors is that they don't share any semantic similarity between words, i.e., for any pair of different words $a, b$, $\cos(\mathbf{v}_a, \mathbf{v}_b) = 0$. Low-dimensional word embeddings effectively alleviated this problem and similar words can be encoded as similar vectors in geometry space: $\cos(\mathbf{v}_{\text{car}}, \mathbf{v}_{\text{vechicle}}) < \cos(\mathbf{v}_{\text{car}}, \mathbf{v}_{\text{man}})$.

These word embeddings can be effectively learned from large unlabeled text corpora, based on the assumption that words occur in similar contexts tend to have similar meanings (a.k.a. the *distributional hypothesis*). Indeed, learning word embeddings from text has a long-standing history and has been finally popularized by recent scalable algorithms and released sets of pretrained word embeddings such as WORD2VEC (Mikolov et al., 2013), GLOVE (Pennington et al., 2014) and FASTTEXT (Bojanowski et al., 2017). They have become the mainstay of modern NLP systems.

### Recurrent neural networks

The second important idea is the use of recurrent neural networks (RNNs) to model sentences or paragraphs in NLP. *Recurrent neural networks* are a class of neural networks which are suitable to handle sequences of variable length. More concretely, they apply a parameterized function recursively on a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_n$:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \Theta) \tag{3.2}$$

For NLP applications, we represent a sentence or a paragraph as a sequence of words where each word is transformed into a vector (usually through pre-trained word embeddings): $\mathbf{x} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and $\mathbf{h}_t \in \mathbb{R}^h$ can be used to model the contextual information of $\mathbf{x}_{1:t}$.

Vanilla RNNs take the form of

$$\mathbf{h}_t = \tanh(\mathbf{W}^{hh}\mathbf{h}_{t-1} + \mathbf{W}^{hx}\mathbf{x}_t + \mathbf{b}), \tag{3.3}$$

where $\mathbf{W}^{hh} \in \mathbb{R}^{h \times h}, \mathbf{W}^{hx} \in \mathbb{R}^{h \times d}$, $\mathbf{b} \in \mathbb{R}^h$ are the parameters to be learned. To ease the optimization, many variants of RNNs have been proposed. Among them, long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRUs) (Cho et al., 2014) are the commonly used ones. Arguably, LSTM is still the most competitive RNN variant for NLP applications today and also our default choice for the neural models that we will describe. Mathematically, LSTMs can be formulated as follows:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^{ih}\mathbf{h}_{t-1} + \mathbf{W}^{ix}\mathbf{x_t} + \mathbf{b}^i) && (3.4)\\
\mathbf{f}_t &= \sigma(\mathbf{W}^{fh}\mathbf{h}_{t-1} + \mathbf{W}^{fx}\mathbf{x_t} + \mathbf{b}^f) && (3.5)\\
\mathbf{o}_t &= \sigma(\mathbf{W}^{oh}\mathbf{h}_{t-1} + \mathbf{W}^{ox}\mathbf{x_t} + \mathbf{b}^o) && (3.6)\\
\mathbf{g}_t &= \tanh(\mathbf{W}^{gh}\mathbf{h}_{t-1} + \mathbf{W}^{gx}\mathbf{x_t} + \mathbf{b}^g) && (3.7)\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t && (3.8)\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t), && (3.9)
\end{aligned}$$

where $\mathbf{W}^{ih}, \mathbf{W}^{fh}, \mathbf{W}^{oh}, \mathbf{W}^{gh} \in \mathbb{R}^{h \times h}$, $\mathbf{W}^{ix}, \mathbf{W}^{fx}, \mathbf{W}^{ox}, \mathbf{W}^{gx} \in \mathbb{R}^{h \times d}$ and $\mathbf{b}^i, \mathbf{b}^f, \mathbf{b}^o, \mathbf{b}^g \in \mathbb{R}^h$ are the parameters to be learned.

Finally, a useful elaboration of an RNN is a *bidirectional RNN*. The idea is simple: for a sentence or a paragraph, $\mathbf{x} = \mathbf{x}_1, \ldots, \mathbf{x}_n$, a forward RNN is used from left to right and then another backward RNN is used from right to left:

$$\overrightarrow{\mathbf{h}}_t = f(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{x}_t; \overrightarrow{\Theta}), \quad t = 1, \ldots, n \tag{3.10}$$

$$\overleftarrow{\mathbf{h}}_t = f(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{x}_t; \overleftarrow{\Theta}), \quad t = n, \ldots, 1 \tag{3.11}$$

We define $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t] \in \mathbb{R}^{2h}$ which takes the concatenation of the hidden vectors from the RNNs in both directions. These representations can usefully encode information from both the left context and the right context and are suitable for general-purpose trainable feature-extracting component of many NLP tasks.

### Attention mechanism

The third important component is an attention mechanism. It was first introduced in the *sequence-to-sequence* (seq2seq) models (Sutskever et al., 2014) for neural machine translation (Bahdanau et al., 2015; Luong et al., 2015) and has later been extended to other NLP tasks.

The key idea is, if we want to predict the sentiment of a sentence, or translate a sentence of one language to the other, we usually apply recurrent neural networks to encode a single sentence (or the source sentence for machine translation): $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n$ and use the last time step $\mathbf{h}_n$ to predict the sentiment label or the first word in the target language:

$$P(Y = y) = \frac{\exp(\mathbf{W}_y \mathbf{h}_n)}{\sum_{y'} \exp(\mathbf{W}_{y'} \mathbf{h}_n)} \tag{3.12}$$

This requires the model to be able to compress all the necessary information of a sentence into a fixed-length vector, which causes an information bottleneck in improving performance. An attention mechanism is designed to solve this problem: instead of squashing all the information into the last hidden vector, it looks at the hidden vectors at all time steps

and chooses a subset of these vectors adaptively:

$$\alpha_i = \frac{\exp\left(g(\mathbf{h}_i, \mathbf{w}; \Theta_g)\right)}{\sum_{i'=1}^{n} \exp\left(g(\mathbf{h}_{i'}, \mathbf{w}; \Theta_g)\right)} \tag{3.13}$$

$$\mathbf{c} = \sum_{i=1}^{n} \alpha_i \mathbf{h}_i \tag{3.14}$$

Here $\mathbf{w}$ can be a task-specific vector learned from the training process, or taken as the current target hidden state in machine translation and $g$ is a parameteric function which can be chosen in various ways, such as dot product, bilinear product, or one hidden layer of MLP:

$$g_{\text{dot}}(\mathbf{h}_i, \mathbf{w}) = \mathbf{h}_i{}^{\intercal}\mathbf{w} \tag{3.15}$$

$$g_{\text{bilinear}}(\mathbf{h}_i, \mathbf{w}) = \mathbf{h}_i{}^{\intercal}\mathbf{W}\mathbf{w} \tag{3.16}$$

$$g_{\text{MLP}}(\mathbf{h}_i, \mathbf{w}) = \mathbf{v}^{\intercal} \tanh(\mathbf{W}^h\mathbf{h}_i + \mathbf{W}^w\mathbf{w}) \tag{3.17}$$

Roughly, an attention mechanism computes a similarity score for each $\mathbf{h}_i$ and then a softmax function is applied which returns a discrete probability distribution over all the time steps. Thus $\alpha$ essentially captures which parts of the sentence are indeed relevant and $\mathbf{c}$ aggregates over all the time steps with a weighted sum and can be used for final prediction. We are not going into more details and interested readers are referred to Bahdanau et al. (2015); Luong et al. (2015).

Attention mechanisms have been proved widely effective in numerous applications and become an integral part of neural NLP models. Recently, Parikh et al. (2016) and Vaswani et al. (2017) conjectured that attention mechanisms don't have to be used in conjunction with recurrent neural networks and can be built purely based on word embeddings and feed-forward networks, while providing minimal sequence information. This class of models usually requires less parameters and is more parallelizable and scalable — in particular, the TRANSFORMER model proposed in Vaswani et al. (2017) has become a recent trend and we will discuss it more in Section 3.4.3.

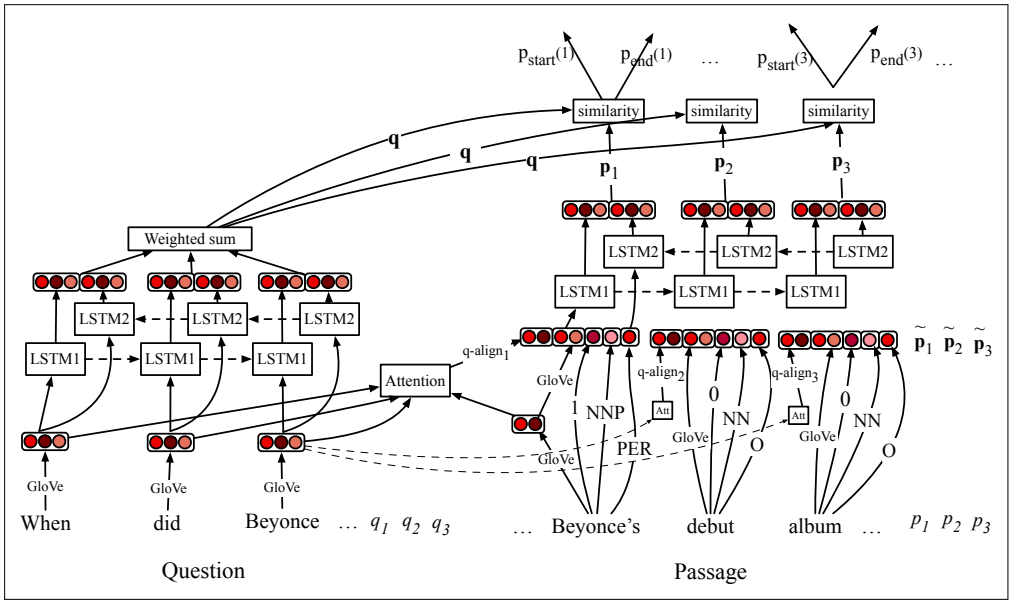



Figure 3.1: A full model of STANFORD ATTENTIVE READER. Image courtesy: https://web.stanford.edu/ jurafsky/slp3/23.pdf.

### 3.2.2 The Model

At this point, we are equipped with all the building blocks. How can we build effective neural models out of them for reading comprehension? What are the key ingredients? Next we introduce our model: the STANFORD ATTENTIVE READER. Our model is inspired by the ATTENTIVE READER described in Hermann et al. (2015) and other concurrent works, with a goal of making the model simple yet powerful. We first describe its full form for span prediction problems that we introduced in Chen et al. (2017) and then later we discuss its other variants.

Let's first recap the setting of span-based reading comprehension problems: Given a single passage $p$ consisting of $l_p$ tokens $(p_1, p_2, \ldots, p_{l_p})$ and a question $q$ consisting of $l_q$ tokens $(q_1, q_2, \ldots, q_{l_q})$, the goal is to predict a span $(a_{\text{start}}, a_{\text{end}})$ where $1 \leq a_{\text{start}} \leq a_{\text{end}} \leq l_p$ so that the corresponding string $p_{a_{\text{start}}}, p_{a_{\text{start}}+1}, \ldots, p_{a_{\text{end}}}$ gives the answer to the question.

The full model is illustrated in Figure 3.1. At a high level, the model first builds a vector representation for the question and builds a vector representation for each token in

the passage. It then computes a similarity function between the question and its passage word in context, and then uses the question-passage similarity scores to decide the starting and ending positions of the answer span. The model builds on top of the low-dimensional, pre-trained word embeddings for each word in the passage and question (with linguistic annotations optionally). All the parameters for passage/question encoding and similarity functions are optimized jointly for the final answer prediction. Let's go into further details of each component:

**Question encoding**

The question encoding is relatively simple: we first map each question word $q_i$ into its word embedding $\mathbf{E}(q_i) \in \mathbb{R}^d$ and then we apply a bi-directional LSTM on top of them and finally obtain:

$$\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{l_q} = \text{BiLSTM}(\mathbf{E}(q_1), \mathbf{E}(q_2), \ldots, \mathbf{E}(q_{l_q}); \Theta^{(q)}) \in \mathbb{R}^h \tag{3.18}$$

We then aggregate these hidden units into one single vector through an attention layer:

$$b_j = \frac{\exp(\mathbf{w}^{q\intercal}\mathbf{q}_j)}{\sum_{j'} \exp(\mathbf{w}^{q\intercal}\mathbf{q}_{j'})} \tag{3.19}$$

$$\mathbf{q} = \sum_j b_j \mathbf{q}_j \tag{3.20}$$

$b_j$ measures the importance of each question word and $\mathbf{w}^q \in \mathbb{R}^h$ is a weight vector to be learned. Therefore, $\mathbf{q} \in \mathbb{R}^h$ is the final vector representation for the question. Indeed, it was simpler (and also common) to represent $\mathbf{q}$ as the concatenation of the last hidden vector from the LSTMs in both directions. However, based on the empirical performance, we find that adding this attention layer helps consistently as it adds more weight to the more relevant question words.

### Passage encoding

Passage encoding is similar, as we also first form an input representation $\tilde{\mathbf{p}}_i \in \mathbb{R}^{\tilde{d}}$ for each word in the passage and pass them through another bidirectional LSTM:

$$\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{l_p} = \text{BiLSTM}\left(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2, \ldots, \tilde{\mathbf{p}}_{l_p}; \Theta^{(p)}\right) \in \mathbb{R}^h \tag{3.21}$$

The input representation $\tilde{\mathbf{p}}_i$ can be divided into two categories: one is to encode *the properties of each word itself*, and the other is to encode *its relevance with respect to the question*.

For the first category, in addition to word embedding $f_{emb}(p_i) = \mathbf{E}(p_i) \in \mathbb{R}^d$, we also add some manual features which reflect the properties of word $p_i$ in its context, including its part-of-speech (POS) and named entity recognition (NER) tags and its (normalized) term frequency (TF): $f_{token}(p_i) = (\text{POS}(p_i), \text{NER}(p_i), \text{TF}(p_i))$. For POS and NER tags, we run off-the-shelf tools and convert it into a one-hot representation as the set of tags is small. The TF feature is real-valued number which measures how many times the word appears in the passage divided by the total number of words.

For the second category, we consider two types of representations:

- **Exact match**: $f_{exact\_match}(p_i) = \mathbb{I}(p_i \in q) \in \mathbb{R}$. In practice, we use three simple binary features, indicating whether $p_i$ can be exactly matched to one question word in $q$, either in its original, lowercase or lemma form.

- **Aligned question embeddings**: The exact match features encode the hard alignment between question words and passage words. Aligned question embeddings aim to encode a soft notion of alignment between words in the word embedding space, so that similar (but non-identical) words, e.g., *car* and *vehicle*, can be well aligned. Concretely, we use

$$f_{align}(p_i) = \sum_j a_{i,j} \mathbf{E}(q_j) \tag{3.22}$$

  where $a_{i,j}$ are the attention weights which capture the similarity between $p_i$ and each question words $q_j$ and $\mathbf{E}(q_j) \in \mathbb{R}^d$ is the word embedding for each question word. $a_{i,j}$

is computed by the dot product between nonlinear mappings of word embeddings:

$$a_{i,j} = \frac{\exp\left(\text{MLP}(\mathbf{E}(p_i))^\intercal \text{MLP}(\mathbf{E}(q_j))\right)}{\sum_{j'} \exp\left(\text{MLP}(\mathbf{E}(p_i))^\intercal \text{MLP}(\mathbf{E}(q_{j'}))\right)}, \tag{3.23}$$

and $\text{MLP}(\mathbf{x}) = \max(0, \mathbf{W}_{\text{MLP}}\mathbf{x} + \mathbf{b}_{\text{MLP}})$ is a single dense layer with ReLU nonlinearity, where $\mathbf{W}_{\text{MLP}} \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_{\text{MLP}} \in \mathbb{R}^d$.

Finally, we simply concatenate the four components and form the input representation:

$$\tilde{\mathbf{p}}_i = (f_{emb}(p_i), f_{token}(p_i), f_{exact\_match}(p_i), f_{align}(p_i)) \in \mathbb{R}^{\tilde{d}} \tag{3.24}$$

### Answer prediction

We have vector representations for both the passage $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{l_p} \in \mathbb{R}^h$ and the question $\mathbf{q} \in \mathbb{R}^h$ and the goal is to predict the span that is most likely the correct answer. We employ the idea of attention mechanism again and train two separate classifiers, one is to predict the start position of the span while the other is to predict the end position. More specifically, we use a bilinear product to capture the similarity between $\mathbf{p}_i$ and $\mathbf{q}$:

$$P^{(\text{start})}(i) = \frac{\exp\left(\mathbf{p}_i \mathbf{W}^{(\text{start})} \mathbf{q}\right)}{\sum_{i'} \exp\left(\mathbf{p}_{i'} \mathbf{W}^{(\text{start})} \mathbf{q}\right)} \tag{3.25}$$

$$P^{(\text{end})}(i) = \frac{\exp\left(\mathbf{p}_i \mathbf{W}^{(\text{end})} \mathbf{q}\right)}{\sum_{i'} \exp\left(\mathbf{p}_{i'} \mathbf{W}^{(\text{end})} \mathbf{q}\right)}, \tag{3.26}$$

where $\mathbf{W}^{(\text{start})}, \mathbf{W}^{(\text{end})} \in \mathbb{R}^{h \times h}$ are additional parameters to be learned. This is slightly different from the formulation of attention as we don't need to take the weighted sum of all the vector representations. Instead, we use the normalized weights to make direct predictions. We use bilinear products because we find them to work well empirically.

### Training and inference

The final training objective is to minimize the cross-entropy loss:

$$\mathcal{L} = -\sum \log P^{(\text{start})}(a_{\text{start}}) - \sum \log P^{(\text{end})}(a_{\text{end}}), \tag{3.27}$$

and all the parameters $\Theta = \Theta^{(p)}, \Theta^{(q)}, \mathbf{w}^{(q)}, \mathbf{W}_{\text{MLP}}, \mathbf{b}_{\text{MLP}}, \mathbf{W}^{(\text{start})}, \mathbf{W}^{(\text{end})}$ are optimized jointly with stochastic gradient methods.[1]

During inference, we choose the span $p_i, \ldots, p_{i'}$ such that $i \leq i' \leq i + max\_len$ and $P^{(\text{start})}(i) \times P^{(\text{end})}(i')$ is maximized. $max\_len$ is a pre-defined constant (e.g., 15) which controls the maximum length of the answer.

### 3.2.3 Extensions

In the following, we give a few variants of the STANFORD ATTENTIVE READER for other types of reading comprehension problems. All these models follow the same process of passage encoding and question encoding as described above, hence we have $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{l_p} \in \mathbb{R}^h$ and $\mathbf{q} \in \mathbb{R}^h$. We only discuss the answer prediction component and training objectives.

**Cloze style.** Similarly, we can compute an attention function using a bilinear product of the question over all the words in the passage, and then compute an output vector $\mathbf{o}$ which takes a weighted sum of all the paragraph representations:

$$\alpha_i = \frac{\exp\left(\mathbf{p}_i \mathbf{W} \mathbf{q}\right)}{\sum_{i'} \exp\left(\mathbf{p}_{i'} \mathbf{W} \mathbf{q}\right)} \tag{3.28}$$

$$\mathbf{o} = \sum_i \alpha_i \mathbf{p}_i \tag{3.29}$$

The output vector $\mathbf{o}$ can be used to predict the missing word or entity:

$$P(Y = e \mid p, q) = \frac{\exp(\mathbf{W}_e^{(a)} \mathbf{o})}{\sum_{e' \in \mathcal{E}} \exp\left(\mathbf{W}_{e'}^{(a)} \mathbf{o}\right)}, \tag{3.30}$$

where $\mathcal{E}$ denotes the candidate set of entities or words. It is straightforward to adopt a negative log-likelihood objective for training and choose $e \in \mathcal{E}$ which maximizes $\mathbf{W}_e^{(a)} \mathbf{o}$ during prediction. This model has been studied in our earlier paper (Chen et al., 2016) for the CNN/DAILY MAIL dataset and (Onishi et al., 2016) for the WHO-DID-WHAT dataset.

[1] We exclude word embeddings here but it is also common to treat all or a subset of the word embeddings as parameters and fine-tune them during training.

**Multiple choice.** In this setting, $k$ hypothesized answers are given $\mathcal{A} = \{a_1, \ldots, a_k\}$ and we can encode each of them into a vector $\mathbf{a}_i$ by applying a third BiLSTM, similar to our question encoding step. We can then compute the output vector $\mathbf{o}$ as in Equation 3.29 and compare it with each hypothesized answer vector $\mathbf{a}_i$ through another similarity function using a bilinear product:

$$P(Y = i \mid p, q) = \frac{\exp(\mathbf{a}_i \mathbf{W}^{(a)} \mathbf{o})}{\sum_{i'=1,\ldots,k} \exp\left(\mathbf{a}_{i'} \mathbf{W}^{(a)} \mathbf{o}\right)} \tag{3.31}$$

The cross-entroy loss is also used for training. This model has been studied in Lai et al. (2017) for the RACE dataset.

**Free-form answer.** For this type of problems, the answer isn't restricted to a single entity or a span in the passage and can take any sequence of words and the most common solution is to incorporate an LSTM sequence decoder into the current framework. In more detail, assume the answer string is $a = (a_1, a_2, \ldots, a_{l_a})$ and a special "end-of-sequence" token $\langle eos \rangle$ is added to the end of each answer. We can compute the output vector $\mathbf{o}$ again as in Equation 3.29. and the decoder generates a word at a time and hence the conditional probability can be decomposed as:

$$P(a \mid p, q) = P(a \mid \mathbf{o}) = \prod_{j=1}^{l_a} P(a_j \mid a_{<j}, \mathbf{o}) \tag{3.32}$$

$P(a_j \mid a_{<j}, \mathbf{o})$ is parameterized as an LSTM which takes $\mathbf{o}$ as the initial hidden vector, and $a_j$ is predicted based on the hidden vector $\mathbf{h}_j$ for the full vocabulary $\mathcal{V} \cup \{\langle eos \rangle\}$. The training objective is

$$\mathcal{L} = -\log P(a \mid p, q) = -\log \sum_{j=1}^{l_a} P(a_j \mid a_{<j}, \mathbf{o}) \tag{3.33}$$

For prediction, one word is predicted at a time which maximizes $P(a_j \mid a_{<j}, \mathbf{o})$ and then fed into the next time step, until the token $\langle eos \rangle$ is predicted. We are not going to elaborate more on it as they are standard components in sequence-to-sequence models (Sutskever

et al., 2014).

This class of models has been studied on the MS MARCO (Nguyen et al., 2016) and the NARRATIVEQA (Kočiskỳ et al., 2018) datasets. However, as the free-form answer reading comprehension problems are more complex and more difficult to evaluate, we think that these methods haven't been fully explored yet, compared to other types of problems. Lastly, we believe that a *copy mechanism* proposed for summarization tasks (Gu et al., 2016; See et al., 2017), which allows the decoder to choose either to copy a word from the source text or to generate a word from the vocabulary, would be highly useful for reading comprehension tasks as well, as answer words are still likely to appear in the paragraph or question. We will discuss one model with a copy mechanism in Section 6.3.

## 3.3 Experiments

### 3.3.1 Datasets

We evaluate our model on CNN/DAILY MAIL (Hermann et al., 2015) and SQUAD (Rajpurkar et al., 2016), the two most popular and competitive reading comprehension datasets. We have described them before in Section 2.1.3 regarding their importance in the development of neural reading comprehension and the way the datasets were constructed. Now we give a brief review of these datasets and the statistics.

- The CNN/DAILY MAIL datasets were made from articles on the news websites CNN and Daily Mail, utilizing articles and their bullet point summaries. One bullet point is converted to a question with one entity replaced by a placeholder and the answer is this entity. The text has been run through a Google NLP pipeline. It it tokenized, lowercased, and named entity recognition and coreference resolution have been run. For each coreference chain containing at least one named entity, all items in the chain are replaced by an @entity$n$ marker, for a distinct index $n$ (Table 2.1 (a)). On average, both the CNN and DAILY MAIL contain 26.2 different entities in the article. The training, development, and testing examples were collected from the news articles at different times. The accuracy (percentage of examples predicting the correct entity) is used for evaluation.

| | cloze style | | span prediction |
|---|---|---|---|
| | **CNN** | **Daily Mail** | **SQuAD** |
| #Train | 380,298 | 879,450 | 87,599 |
| #Dev | 3,924 | 64,835 | 10,570 |
| #Test | 3,198 | 53,182 | 9,533 |
| Passage: avg. tokens | 761.8 | 813.1 | 134.4 |
| Question: avg. tokens | 12.5 | 14.3 | 11.3 |
| Answer: avg. tokens | 1.0 | 1.0 | 3.1 |

Table 3.2: Data statistics of CNN/DAILY MAIL and SQUAD. The average numbers of tokens are computed based on the training set.

- The SQUAD dataset was collected based on Wikipedia articles. 536 high-quality Wikipedia articles were sampled and crowdworkers created questions based on each individual paragraph (paragraphs shorter than 500 characters were discarded), requiring that answer must be highlighted from the paragraph (Table 2.1 (c)). The training/development/testing splits were made randomly based on articles (80% vs. 10% vs. 10%). To estimate human performance and also make evaluation more reliable, they collected a few additional answers for each question (each question in the development set has 3.3 answers on average). Exact match and macro-averaged F1 scores are used for evaluation, as we discussed in Section 2.2.2. Note that SQUAD 2.0 (Rajpurkar et al., 2018) was proposed more recently, which added 53,775 unanswerable questions to the original dataset and we will discuss it in Section 4.2. For most of this thesis, SQUAD refers to SQUAD 1.1 unless stated otherwise.

Table 3.2 gives more detailed statistics of the datasets. As it is shown, the CNN/DAILY MAIL datasets are much larger than SQUAD (almost one order of magnitude bigger) due to the way the datasets were constructed. The passages used in CNN/DAILY MAIL are also much longer — 761.8 and 813.1 tokens for CNN and DAILY MAIL respectively, while it is 134.4 tokens for SQuAD. Finally, the answers in SQUAD consists of only 3.1 tokens on average, which reflects the fact the most of the SQUAD questions are factoid and a large portion of the answers are common nouns or named entities.

### 3.3.2 Implementation Details

Besides different model architecture designs, implementation details also play a crucial role in the final performance of these neural reading comprehension systems. In the following we highlight a few important aspects that we haven't covered yet and finally give the model specifications that we used on the two datasets.

**Stacked BiLSTMs.** One simple idea is to increase the depth of bidirectional LSTMs for question and passage encoding. It computes $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t] \in \mathbb{R}^{2h}$ and then regard $\mathbf{h}_t$ as the input $\mathbf{x}_t$ of the next layer and pass them into another BiLSTM, and so on. We generally find that stacking BiLSTMs works better than a one-layer BiLSTM and we used 3 layers for the SQuAD experiment.[2]

**Dropout.** Dropout is an effective and widely used approach to regularization in neural networks. Simply put, dropout refers to masking out some units at random during the training process. For our model, dropout can be added to the word embeddings, input vectors and hidden vectors of every LSTM layer. Finally, the variational dropout approach (Gal and Ghahramani, 2016) has demonstrated to work better than the standard dropout on regularizing RNNs. The idea is to apply the same dropout mask at each time step for both inputs, outputs and recurrent layers, i.e., the same units are dropped at each time step. We suggest readers to use this variant in practice.[3]

**Handling word embeddings.** One common way (and also our default choice) to handle word embeddings is to keep the most frequent $K$ (e.g., $K = 500,000$) word types in the training set and map all other words to an $\langle unk \rangle$ token and then use pre-trained word embeddings to initialize the $K$ words. Typically, when the training set is large enough, we fine tune all the word embeddings; when the training set is relatively small (e.g., SQuAD), we usually keep all the word embeddings fixed as static features. In Chen et al. (2017), we find that it helps to fine-tune the most frequent question words because the representations of these key words such as *what*, *how*, *which* could be crucial for reading comprehension

[2] We only used a shallow one-layer BiLSTM for the CNN/Daily Mail experiments in 2016 though.
[3] We didn't include variational dropout in our published paper results but later found it useful.

systems. Some studies such as (Dhingra et al., 2017a) demonstrated the use of pre-trained embeddings and the ways of handling out-of-vocabulary words have a large impact on the performance of reading comprehension tasks.

**Model specifications.** For all the experiments which require linguistic annotations (lemma, part-of-speech tags, named entity tags, dependency parses), we use the Stanford CoreNLP toolkit (Manning et al., 2014) for preprocessing. For training all the neural models, we sort all the examples by the length of its passage, and randomly sample a mini-batch of size 32 for each update.

For the results on CNN/DAILY MAIL, we use the lowercased, 100-dimensional pre-trained GLOVE word embeddings (Pennington et al., 2014) trained on Wikipedia and Gigaword for initialization. The attention and output parameters are initialized from a uniform distribution between $(-0.01, 0.01)$, and the LSTM weights are initialized from a Gaussian distribution $\mathcal{N}(0, 0.1)$. We use a 1-layer BiLSTM of hidden size $h = 128$ for CNN and $h = 256$ for DAILY MAIL. Optimization is carried out using vanilla stochastic gradient descent (SGD), with a fixed learning rate of $0.1$. We also apply dropout with probability $0.2$ to the embedding layer and gradient clipping when the norm of gradients exceeds $10$.

For the results on SQUAD, we use 3-layer BiLSTMs with $h = 128$ hidden units for both paragraph and question encoding. We use ADAMAX for optimization as described in (Kingma and Ba, 2014). Dropout with probably $0.3$ is applied to word embeddings and all the hidden units of LSTMs. We used the $300$-dimensional GLOVE word embeddings trained from 840B Web crawl data for initialization and only fine-tune the 1000 most frequent question words.

Other implementation details can be found in the following two Github repositories:

- https://github.com/danqi/rc-cnn-dailymail for our experiments in Chen et al. (2016).
- https://github.com/facebookresearch/DrQA for our experiments in Chen et al. (2017).

We also would like to caution readers that our experimental results were published in two papers (2016 and 2017) and they differ in various places. A key difference is that our results on CNN/DAILY MAIL didn't include manual features $f_{token}(p_i)$, exact match features $f_{exact\_match}(p_i)$, aligned question embeddings $f_{align}(p_i)$ and $\tilde{\mathbf{p}}_i$ just takes the word

| Model | CNN | | DAILY MAIL | |
|---|---|---|---|---|
| | **Dev** | **Test** | **Dev** | **Test** |
| Frame-semantic model † | 36.3 | 40.2 | 35.5 | 35.5 |
| Word distance model † | 50.5 | 50.9 | 56.4 | 55.5 |
| Deep LSTM Reader † | 55.0 | 57.0 | 63.3 | 62.2 |
| Attentive Reader † | 61.6 | 63.0 | 70.5 | 69.0 |
| Impatient Reader † | 61.8 | 63.8 | 69.0 | 68.0 |
| MemNNs (window memory) ‡ | 58.0 | 60.6 | N/A | N/A |
| MemNNs (window memory + self-sup.) ‡ | 63.4 | 66.8 | N/A | N/A |
| MemNNs (ensemble) ‡ | 66.2* | 69.4* | N/A | N/A |
| Our feature-based classifier | 67.1 | 67.9 | 69.1 | 68.3 |
| Stanford Attentive Reader | 72.5 | 72.7 | 76.9 | 76.0 |
| Stanford Attentive Reader (ensemble) | 76.2* | 76.5* | 79.5* | 78.7* |

Table 3.3: Accuracy of all models on the CNN and DAILY MAIL datasets. Results marked † are from Hermann et al. (2015) and results marked ‡ are from Hill et al. (2016). The numbers marked with * indicate that the results are from ensemble models.

embedding $\mathbf{E}(p_i)$. Another difference is that we didn't have the attention layer in question encoding before but simply concatenated the last hidden vectors from the LSTMs in both directions. We believe that that these additions are useful on CNN/DAILY MAIL and other cloze style tasks as well, but we didn't further investigate it.

### 3.3.3 Experimental Results

#### 3.3.3.1 Results on CNN/DAILY MAIL

Table 3.3 presents the results that we reported in Chen et al. (2016). We run our neural models 5 times independently with different random seeds and report average performance across the runs. We also report ensemble results which average the prediction probabilities of the 5 models. We also present the results for the feature-based classifier we described in Section 3.1.

**Baselines.** We were among the earliest groups to study this first large-scale reading comprehension dataset. At the time, Hermann et al. (2015) and Hill et al. (2016) proposed a

few baselines, both symbolic approaches and neural models, for this task. The baselines include:

- A FRAME-SEMANTIC model in Hermann et al. (2015), which they run a state-of-the-art semantic parser, and extract entity-predicate triples denoted as $(e_1, V, e_2)$ from both the question and the passage, and attempt to match the correct entity using a number of heuristic rules.

- A WORD DISTANCE model in Hermann et al. (2015), in which they align the placeholder of the question with each possible entity, and compute a distance measure between the question and the passage around the aligned entity.

- Several LSTM-based neural models proposed in Hermann et al. (2015), named DEEP LSTM READER, ATTENTIVE READER and IMPATIENT READER. The DEEP LSTM READER just processes the question and the passage as one sequence using a deep LSTM (without attention mechanism), and makes a prediction in the end. The ATTENTIVE READER is similar in spirit to ours, as it computes an attention function between the question vector and all the passage vectors; while the IMPATIENT READER computes an attention function for all the question words and recurrently accumulates information as the model reads each question word.

- WINDOW-BASED MEMORY NETWORKS proposed by Hill et al. (2016) is based on the memory network architecture (Weston et al., 2015). We think the model is also similar to ours and the biggest difference is their way of encoding passages: they only use a 5-word context window when evaluating a candidate entity and they use a positional unigram approach to encode the contextual embeddings. If a window consists of $5$ words $x_1, x_2, \ldots, x_5$, then it is encoded as $\sum \mathbf{E}_i(x_i)$, resulting in $5$ separate embedding matrices to learn. They encode the $5$-word window surrounding the placeholder in a similar way and all other words in the question text are ignored. In addition, they simply use a dot product to compute the "relevance" between the question and a contextual embedding.

As seen in Table 3.3, our feature-based classifier obtains 67.9% accuracy on the CNN test set and 68.3% accuracy on the DAILY MAIL test set. It significantly outperforms any

| Method | Dev | | Test | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| Logistic regression (Rajpurkar et al., 2016) | 40.0 | 51.0 | 40.4 | 51.0 |
| Match-LSTM (Wang and Jiang, 2017) | 64.1 | 73.9 | 64.7 | 73.7 |
| RaSoR (Lee et al., 2016) | 66.4 | 74.9 | 67.4 | 75.5 |
| DCN (Xiong et al., 2017) | 65.4 | 75.6 | 66.2 | 75.9 |
| BiDAF (Seo et al., 2017) | 67.7 | 77.3 | 68.0 | 77.3 |
| **Our model** (Chen et al., 2017) | 69.5 | 78.8 | 70.0 | 79.0 |
| R-NET (Wang et al., 2017) | 71.1 | 79.5 | 71.3 | 79.7 |
| BiDAF + self-attention (Peters et al., 2018) | N/A | N/A | 72.1 | 81.1 |
| FusionNet (Huang et al., 2018b) | N/A | N/A | 76.0 | 83.9 |
| QANet (Yu et al., 2018) | 73.6 | 82.7 | N/A | N/A |
| SAN (Liu et al., 2018) | 76.2 | 84.1 | 76.8 | 84.4 |
| BiDAF + self-attention + ELMo (Peters et al., 2018) | N/A | N/A | 78.6 | 85.8 |
| BERT (Devlin et al., 2018) | 84.1 | 90.9 | N/A | N/A |
| Human performance (Rajpurkar et al., 2016) | 80.3 | 90.5 | 82.3 | 91.2 |

Table 3.4: Evaluation results on the SQuAD dataset (single model only). The results below "our model" were released after we finished the paper in Feb 2017. We only list representative models and report the results from the published papers. For a fair comparison, we didn't include the results which use other training resources (e.g., TriviaQA) or data augmentation techniques, except pre-trained language models, but we will discuss them in Section 3.4.

of the symbolic approaches reported in Hermann et al. (2015). We feel that their frame-semantic model is not suitable for these tasks due to the poor coverage of the parser and is not representative of what a straightforward NLP system can achieve. Indeed, the frame-semantic model is even markedly inferior to the word distance model. To our surprise, our feature-based classifier even outperforms all the neural network systems in Hermann et al. (2015) and the best single-system result reported from Hill et al. (2016). Moreover, our single-model neural network surpasses the previous results by a large margin (over 5%), pushing up the state-of-the-art accuracies to 72.7% and 76.0% respectively. The ensembles of 5 models consistently bring further 2-4% gains.

#### 3.3.3.2 Results on SQuAD

Table 3.4 presents our evaluation results on both the development and testing sets. SQuAD has been a very competitive benchmark since it was created and we only list a few representative models and the single-model performance. It is well known that the ensemble models can further improve the performance by a few points. We also included results from the logistic regression baseline (i.e., feature-based classifiers) created by the original authors (Rajpurkar et al., 2016).

Our system can achieve 70.0% exact match and 79.0% F1 scores on the test set, which surpassed all the published results and matched the top performance on the SQuAD leaderboard[4] at the time we wrote the paper (Chen et al., 2017). Additionally, we think that our model is conceptually simpler than most of the existing systems. Compared to the logistic regression baseline, which gets F1 $= 51.0$, this model is already close to a 30% absolute improvement and it is a big win for neural models.

Since then, SQuAD has received tremendous attention and great progress has been made on this dataset, as seen in Table 3.4. Recent advances include pre-trained language models for initialization, more fine-grained attention mechanisms, data augmentation techniques and even better training objectives. We will discuss them in Section 3.4.

#### 3.3.3.3 Ablation studies

| **Features** | **F1** |
|---|---|
| Full | 78.8 |
| No $f_{token}$ | 78.0 (-0.8) |
| No $f_{exact\_match}$ | 77.3 (-1.5) |
| No $f_{aligned}$ | 77.3 (-1.5) |
| No $f_{aligned}$ and $f_{exact\_match}$ | 59.4 (-19.4) |

Table 3.5: Feature ablation analysis of the paragraph representations of our model. Results are reported on the SQuAD development set.

In Chen et al. (2017), we conducted an ablation analysis on the components of the

[4] https://stanford-qa.com.

passage representations. As shown in Table 3.5, all the components contribute to the performance of our final system. We find that, without the aligned question embeddings (only word embeddings and a few manual features), our system is still able to achieve F1 over 77%. The effectiveness of exact match features $f_{exact\_match}$ also indicates that there are a lot of words overlapping between the passage and the question on this dataset. More interestingly, if we remove both $f_{aligned}$ and $f_{exact\_match}$, the performance drops dramatically, so we conclude that both features play a similar but complementary role in the feature representation, like the hard and soft alignments between question and passage words.

### 3.3.4 Analysis: What Have the Models Learned?

In Chen et al. (2016), we attempted to understand better what these models have actually learned, and what depth of language understanding is needed to solve these problems. We approach this by doing a careful hand-analysis of 100 randomly sampled examples from the development set of the CNN dataset.

We roughly classify them into the following categories (if an example satisfies more than one category, we classify it into the earliest one):

**Exact match** The nearest words around the placeholder are also found in the passage surrounding an entity marker; the answer is self-evident.

**Sentence-level paraphrasing** The question text is entailed/rephrased by *exactly* one sentence in the passage, so the answer can definitely be identified from that sentence.

**Partial clue** In many cases, even though we cannot find a complete semantic match between the question text and some sentence, we are still able to infer the answer through partial clues, such as some word/concept overlap.

**Multiple sentences** Multiple sentences must be processed to infer the correct answer.

**Coreference errors** It is unavoidable that there are many coreference errors in the dataset. This category includes those examples with critical coreference errors for the answer entity or key entities appearing in the question. Basically we treat this category as "not answerable".

| # | Category | |
|---|---|---|
| 1 | Exact match | 13% |
| 2 | Paraphrasing | 41% |
| 3 | Partial clue | 19% |
| 4 | Multiple sentences | 2% |
| 5 | Coreference errors | 8% |
| 6 | Ambiguous / hard | 17% |

Table 3.6: An estimate of the breakdown of the dataset into classes, based on the analysis of our sampled 100 examples from the CNN dataset.

**Ambiguous or hard** This category includes examples for which we think humans are not able to obtain the correct answer (confidently).

Table 3.6 provides our estimate of the percentage for each category, and Figure 3.2 presents one representative example from each category. We observe that *paraphrasing* accounts for 41% of the examples and 19% of the examples are in the *partial clue* category. Adding the most simple *exact match* category, we hypothesize a large portion (73% in this subset) of the examples are able to be answered by identifying the most relevant (single) sentence and inferring the answer based upon it. Additionally, only 2 examples require multiple sentences for inference. This is a lower rate than we expected and this suggests that the dataset requires less reasoning than previously thought. To our surprise, "coreference errors" and "ambiguous/hard" cases account for 25% of this sample set, based on our manual analysis, and this certainly will be a barrier for training models with an accuracy much above 75% (although, of course, a model can sometimes make a lucky guess). In fact, our ensemble neural network model is already able to achieve 76.5% on the development set, and we think that the prospect for further improving on this dataset is small.

Let's further take a closer look at the per-category performance of our neural network and feature-based classifier, based on the above categorization. As shown in Figure 3.3, we have the following observations: (i) The exact-match cases are quite simple and both systems get 100% correct. (ii) For the ambiguous/hard and entity-linking-error cases, meeting our expectations, both of the systems perform poorly. (iii) The two systems mainly differ in the paraphrasing cases, and some of the "partial clue" cases. This clearly shows how neural

| Category | Question | Passage |
|---|---|---|
| Exact Match | *it 's clear @entity0 is leaning toward* **@placeholder** , says an expert who monitors @entity0 | . . . @entity116 , who follows @entity0 's operations and propaganda closely , recently told @entity3 , *it 's clear @entity0 is leaning toward* **@entity60** in terms of doctrine , ideology and an emphasis on holding territory after operations . . . . |
| Paraphrasing | **@placeholder** says he understands why @entity0 wo n't play at his tournament | . . . @entity0 called me personally to let me know that he would n't be playing here at @entity23 , " **@entity3** said on his @entity21 event 's website . . . . |
| Partial clue | a tv movie based on @entity2 's book **@placeholder** casts a @entity76 actor as @entity5 | . . . to @entity12 @entity2 professed that his **@entity11** is not a religious book . . . . |
| Multiple sent. | he 's doing a his - and - her duet all by himself , @entity6 said of **@placeholder** | . . . we got some groundbreaking performances , here too , tonight , @entity6 said . we got **@entity17** , who will be doing some musical performances . he 's doing a his - and - her duet all by himself . . . . |
| Coref. Error | rapper **@placeholder** " disgusted , " cancels upcoming show for @entity280 | . . . with hip - hop star **@entity246** saying on @entity247 that he was canceling an upcoming show for the @entity249 . . . . (but @entity249 = @entity280 = SAEs) |
| Hard | pilot error and snow were reasons stated for **@placeholder** plane crash | . . . a small aircraft carrying **@entity5** , @entity6 and @entity7 the @entity12 @entity3 crashed a few miles from @entity9 , near @entity10 , @entity11 . . . . |

Figure 3.2: Some representative examples from each category on the CNN dataset.

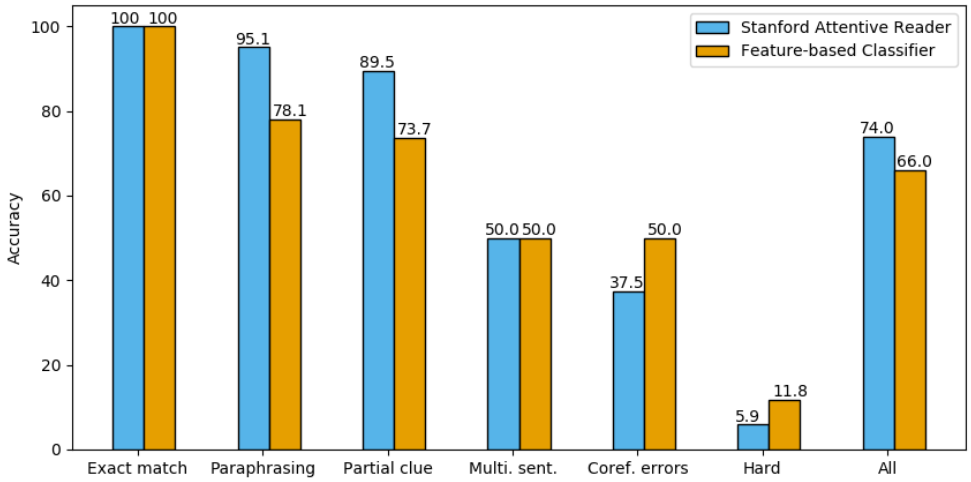



Figure 3.3: The per-category performance of our two systems: the STANFORD ATTENTIVE READER and the feature-based classifier, on the sampled 100 examples of the CNN dataset.

networks are better capable of learning semantic matches involving paraphrasing or lexical variation between the two sentences. (iv) We believe that the neural network model already achieves near-optimal performance on all the single-sentence and unambiguous cases.

To sum up, we find that neural networks are certainly more powerful at recognizing lexical matches and paraphrases compared to conventional feature-based models; while it is still unclear whether they also win out on the examples which require more complex textual reasoning as the current datasets are still quite limited in that respect.

## 3.4 Further Advances

In this section, we summarize recent advances in neural reading comprehension. We divide them into the following four categories: word representations, attention mechanisms, alternatives to LSTMs, and others (such as training objectives, data augmentation). We give a summary and discuss their importance in the end.

### 3.4.1 Word Representations

The first category is better word representations for question and passage words, so the neural models are built off of better grounds. Learning better distributed word representations from text or finding the best set of word embeddings for specific tasks still remains an active research topic — for example, Mikolov et al. (2017) find that replacing GLOVE pre-trained vectors with the new FASTTEXT vectors (Bojanowski et al., 2017) in our model brings about 1 point of improvement on SQUAD. More than that, there are two key ideas which have been proved (highly) useful:

#### Character embeddings

The first idea is to use character-level embeddings to represent words, which are especially helpful for rare or out-of-vocabulary words. Most of the existing works employ a CONVOLUTIONAL NEURAL NETWORK (CNN), which can usefully exploit the surface patterns of $n$-gram characters. More concretely, let $\mathcal{C}$ be the vocabulary of characters and each word type $x$ can be represented as a sequence of characters $(c_1, \ldots, c_{|x|}), \forall c_i \in \mathcal{C}$. We first map each character in $\mathcal{C}$ into a $d_c$-dimensional vector, so word $x$ can be represented as $\mathbf{c}_1, \ldots, \mathbf{c}_{|x|}$.

Next we apply a convolution layer with a filter $\mathbf{w} \in \mathbb{R}^{d_c \times w}$ of width $w$, and we denote $\mathbf{c}_{i:i+j}$ as the concatenation of $\mathbf{c}_i, \mathbf{c}_{i+1}, \ldots, \mathbf{c}_{i+j}$. Therefore, for $i = 1, \ldots, |x| - w + 1$, we can apply this filter $\mathbf{w}$ and after which we add a bias $b$ and apply a nonlinearity $\tanh$ as follows:

$$f_i = \tanh\left(\mathbf{w}^\intercal \mathbf{c}_{i:i+w-1} + b\right). \tag{3.34}$$

Finally we can apply a *max-pooling* operation on $f_1, \ldots, f_{|x|-w+1}$ and obtain one scalar feature:

$$f = \max_i \{f_i\} \tag{3.35}$$

This feature essentially picks out a character $n$-gram, where the size of the $n$-gram corresponds to the filter width $w$. We can repeat the above process by repeating $d^*$ different filters $\mathbf{w}_1, \ldots, \mathbf{w}_{d^*}$. As a result, we can obtain a character-based word representation for each word type $\mathbf{E}_c(x) \in \mathbb{R}^{d^*}$. All the character embeddings, filter weights $\{\mathbf{w}\}$ and biases

$\{b\}$ are learned during training. More details can be found in Kim (2014). In practice, the dimension of character embeddings $d_c$ usually takes a small value (e.g., 20), width $w$ usually takes $3-5$, while $100$ is a typical value for $d^*$.

### Contextualized word embeddings

Another important idea is *contextualized word embeddings*. Different from traditional word embeddings in which each word type is mapped to one single vector, contextualized word embeddings assign each word a vector as a function of the entire input sentence. These word embeddings can model better complex characteristics of word use (e.g., syntax and semantics) and how these uses vary across linguistic contexts (i.e., polysemy).

A concrete implementation is ELMO detailed in Peters et al. (2018): their contextualized word embeddings are learned functions of the internal states of a deep bidirectional language model, which is pretrained on a large text corpus. Basically, given a sequence of words $(x_1, x_2, \ldots, x_n)$, they run an $L$-layer forward LSTM and models the sequence probability as:

$$P(x_1, x_2, \ldots, x_n) = \prod_{k=1}^{n} P(x_k \mid x_1, \ldots, x_{k-1}) \tag{3.36}$$

Only the top layer of the LSTM $\overrightarrow{\mathbf{h}}_k^{(L)}$ is used to predict the next token $x_{k+1}$. Similarly, another $L$-layer LSTM is run backward and $\overleftarrow{\mathbf{h}}_k^{(L)}$ is used to predict the token $x_{k-1}$. The overall training objective is to maximize the log-likelihood from both directions:

$$\sum_{k=1}^{n} \left( \log P(x_k \mid x_1, \ldots, x_{k-1}; \Theta_x, \overrightarrow{\Theta}_{\text{LSTM}}, \Theta_s) + \log P(x_k \mid x_{k+1}, \ldots, x_n; \Theta_x, \overleftarrow{\Theta}_{\text{LSTM}}, \Theta_s) \right), \tag{3.37}$$

where $\Theta_x$ and $\Theta_s$ are word embeddings and softmax parameters and shared for both LSTMs. The final contextualized word embeddings are computed as a linear combination of all the biLSTM layers and the input word embeddings, multiplied by a linear scalar:

$$\text{ELMO}(x_k) = \gamma \left( s_0 \mathbf{x}_k + \sum_{j=1}^{L} \overrightarrow{s}_j \overrightarrow{\mathbf{h}}_k^{(j)} + \sum_{j=1}^{L} \overleftarrow{s}_j \overleftarrow{\mathbf{h}}_k^{(j)} \right) \tag{3.38}$$

All the weights $\gamma, s_0, \overrightarrow{s}_j, \overleftarrow{s}_j$ are task-specific and learned during the training process.

These contextualized word embeddings are usually used in conjunction with traditional word type embeddings and character embeddings. It turns out that this sort of contextualized word embeddings pre-trained on very large text corpora (e.g., 1B Word Benchmark (Chelba et al., 2014)) has been highly effective. Peters et al. (2018) demonstrated that adding ELMo embeddings ($L = 2$ biLSTM layers with $4096$ units and $512$ dimension projections) to an existing competitive model can bring the F1 score on SQuAD from $81.1$ to $85.8$ directly, a $4.7$ point of absolute improvement.

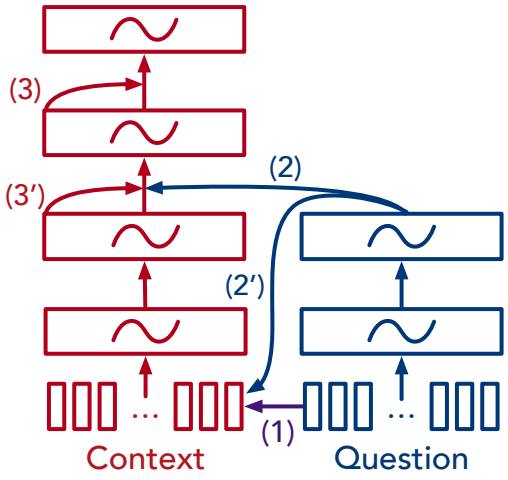
Earlier than ELMo, McCann et al. (2017) proposed CoVe, which learned contextualized word embeddings in a neural machine translation framework, and the resulting encoder can be used in a similar way as an addition to the word embeddings. They also demonstrated a $4.3$ point of absolute improvement on SQuAD.

Very recently, Radford et al. (2018) and Devlin et al. (2018) find that these contextualized word embeddings can not only be used as features of word representations in a task-specific neural architecture (a reading comprehension model in our context), but we can fine-tune the deep language models directly with minimal modifications to perform downstream tasks. This is indeed a very striking result at the time of writing this thesis and we will discuss it more in Section 4.4.2 and there still remain many open questions to answer in the future. Additionally, Devlin et al. (2018) proposed a clever way to train bidirectional language models: instead of always stacking LSTMs in one direction and predicting the next word,[5] they mask out some words at random at the input layer, stack bidirectional layers and predict these masked words at the top layer. They find this training strategy extremely useful empirically.

### 3.4.2 Attention Mechanisms

There have been a multitude of attention variants proposed for neural reading comprehension models, and they aim to capture semantic similarity between the question and the passage, at different levels, multiple granularity, or in a hierarchical way. A typical complex example in this direction can be found at (Huang et al., 2018b). To our best knowledge, there isn't a conclusion yet if there is one single variant that stands out. Our STANFORD

[5] To make it clear, although ELMo adopts a biLSTM, it is essentially the use of two unidirectional LSTMs for predicting the next word in both directions.




| Architectures | (1) | (2) | (2') | (3) | (3') |
|---|---|---|---|---|---|
| Match-LSTM (Wang and Jiang, 2017) | | ✓ | | | |
| DCN (Xiong et al., 2017) | | ✓ | | | ✓ |
| BiDAF (Seo et al., 2017) | | ✓ | | | ✓ |
| RaSoR (Lee et al., 2016) | ✓ | | ✓ | | |
| R-net (Wang et al., 2017) | | ✓ | | ✓ | |
| Our model | ✓ | | | | |

Figure 3.4: A summary of different layers of attention. Image courtesy: (Huang et al., 2018b) with minimal modifications.

ATTENTIVE READER (Section 3.2) takes the most simple possible form of attention (Figure 3.4 illustrates an overview of different layers of attention). Besides that, we think there are two ideas which can generally further improve the performance of these systems:

### Bidirectional attention

Seo et al. (2017) first introduced the idea of *bidirectional attention*. In addition to what we already have, the key difference is that they have the *question-to-passage* attention, which signifies which passage words have the closest similarity to each of the question words. In practice, this can be implemented as: for each word in the question, we can compute an attention map over all the passage words, similar as we did in Equation 3.22 and 3.23, but in opposite directions:

$$f_{q\_align}(q_i) = \sum_j b_{i,j}\mathbf{E}(p_j). \tag{3.39}$$

After this, we can simply feed $f_{q\_align}(q_i)$ into the input layer of the question encoding (Section 3.2.2).

The attention mechanism in Seo et al. (2017) is a bit more complex, but we think it is similar. We also argue that the attention function in this direction is less useful, as also demonstrated in Seo et al. (2017). This is because the questions are generally short (10-20 words on average) and using one single LSTM for question encoding (without extra attention) is usually sufficient.

### Self-attention over passage

The second idea is *self-attention* over the passage words, first introduced in Wang et al. (2017).[6] The intuition is that the passage words can be aligned to the other passage words, with the hope that it can address coreference problems and aggregate information (of the same entity) from multiple places in the passage.

In detail, Wang et al. (2017) first compute the hidden vectors for the passage: $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{l_p}$ (Equation 3.21), and then for each $\mathbf{p}_i$, they apply an attention function over $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{l_p}$ via one hidden layer of MLP (Equation 3.17):

$$a_{i,j} = \frac{\exp\left(g_{\text{MLP}}(\mathbf{p}_i, \mathbf{p}_j)\right)}{\sum_{j'} \exp\left(g_{\text{MLP}}(\mathbf{p}_i, \mathbf{p}_{j'})\right)} \tag{3.40}$$

$$\mathbf{c}_i = \sum_j a_{i,j}\mathbf{p}_j \tag{3.41}$$

Later, $\mathbf{c}_i$ and $\mathbf{p}_i$ are concatenated and fed into another BiLSTM: $\mathbf{h}_i^{(p)} = \text{BiLSTM}(\mathbf{h}_{i-1}^{(p)}, [\mathbf{p}_i, \mathbf{c}_i])$, and can be used as the final passage representations.

[6] They named it as "self-matching attention mechanism" in the paper.

### 3.4.3 Alternatives to LSTMs

All the models we discussed so far are based on recurrent neural networks (RNNs), in particular, LSTMs. It is well known that increasing the depth of neural networks can improve the capacity of models and bring gains in performance (He et al., 2016). We also discussed earlier that deep BiLSTMs of $3$ or $4$ layers usually perform better than a single layer of BiLSTM (Section 3.3.2). However, we are facing two challenges as we further increase the depth of the LSTM models: 1) It gets more difficult to optimize due to the vanishing gradient problem; 2) Scalability becomes an issue as the training/inference time increases linearly as the number of layer grows. It is known that LSTMs are difficult to parallelize and thus scale poorly due to their sequential nature.

On the one hand, there are works which attempt to add highway connections (Srivastava et al., 2015) or residual connections (He et al., 2016) between layers, so it eases the optimization and enables training more layers of LSTMs. On the other hand, people set out to find replacements for LSTMs, getting rid of recurrent structures while still performing similarly or even better.

The most notable work in this line is the TRANSFORMER model proposed by Google researchers (Vaswani et al., 2017). TRANSFORMER only builds on top of word embeddings and simple positional encodings with stacked self-attention layers and position-wise fully connected layers. With residual connections, this model is able to be trained fast with many layers. It first demonstrated superior performance on a machine translation task with $L = 6$ layers (each layer consists of a self-attention and a fully connected feedforward network), and then later was adapted by (Yu et al., 2018) for reading comprehension.

The model called QANET (Yu et al., 2018) stacks multiple convolutional layers followed by the self-attention and fully connected layer, as a building block, for both question and passage encoding as well as a few more layers stacked before the final prediction. The model demonstrated state-of-the-art performance at the time (Table 3.4) while showing significant speed-ups.

Another research work by Lei et al. (2018) proposed a lightweight recurrent unit called SIMPLE RECURRENT UNIT (SRU) by simplifying the LSTM formulation while enabling CUDA-level optimizations for high parallelization. Their results suggest that simplified

recurrence retains strong modeling capacity through layer stacking. They also demonstrate that replacing the LSTMs in our model with their SRU unit can improve the F1 score by 2 points while being faster for training and inference.

### 3.4.4 Others

**Training objectives.** It is also possible to make further progress by improving the training objectives. It is usually straightforward to employ a cross-entropy or max-margin loss for the cloze style or multiple choice problems. However, for span prediction problems, Xiong et al. (2018) suggest that there is a discrepancy between the cross-entropy loss of predicting two endpoints of the answer and the final evaluation metrics, which concerns the word overlap between gold answer and ground truth. For the following example:

> **passage**: Some believe that the Golden State Warriors team of 2017 is one of the greatest teams in NBA history ...
> **question**: Which team is considered to be one of the greatest teams in NBA history?
> **ground truth answer**: the Golden State Warriors team of 2017

Span "Warriors" is also a correct answer, however, from the perspective of cross entropy based training it is no better than the span "history". Xiong et al. (2018) propose to use a mixed training objective which combines cross entropy loss over positions and the word overlap measure trained with reinforcement learning. Basically, they use $P^{(\text{start})}(i)$ and $P^{(\text{end})}(i)$ trained with cross-entroy loss to sample the start and end positions of the answer and then use the F1 score as reward function.

For the free-form answer of reading comprehension problems, there has been many recent advances in training better SEQ2SEQ models especially in the context of neural machine translation, such as sentence-level training (Ranzato et al., 2016) and minimum risk training (Shen et al., 2016). However, we don't see many such applications in reading comprehension problems yet.

**Data augmentation.** Data augmentation has been a very successful approach for image recognition, while it is less explored in NLP problems. Yu et al. (2018) proposed a

| Components | F1 improvement | References |
|---|---|---|
| GLOVE⇒FASTTEXT | 78.9 ⇒ 79.8: +0.9 | (Mikolov et al., 2017) |
| Character embeddings | 75.4 ⇒ 77.3: +1.9 | (Seo et al., 2017) |
| Contextualized embeddings: ELMO | 81.1 ⇒ 85.8: +4.7 | (Peters et al., 2018) |
| Question to passage attention | 73.7 ⇒ 77.3: +3.6 | (Seo et al., 2017) |
| Self-attention over passage | 76.7 ⇒ 79.5: +2.8 | (Wang et al., 2017) |
| 3-layer LSTMs ⇒ 6-layer SRUs | 78.8 ⇒ 80.2: +1.4 | (Lei et al., 2018) |
| Mixed training objective | 82.1 ⇒ 83.1: +1.0 | (Xiong et al., 2018) |
| Data augmentation | 82.7 ⇒ 83.8: +1.1 | (Yu et al., 2018) |

Table 3.7: A summary of recent advances on SQUAD. The numbers are taken from the corresponding papers, on the development set of SQUAD.

simple technique for creating more training data for reading comprehension models. The technique is called *backtranslation* — basically they leverage two state-of-the-art neural machine translation models: one model from English to French and the other model from French to English, and paraphrase each single sentence in the passage by running through the two models (with some modifications to the answer if needed). They obtained 2 points gain in F1 by doing this on SQUAD. Devlin et al. (2018) also find that joint training of SQUAD and TRIVIAQA (Joshi et al., 2017) can help improve the performance on SQUAD modestly.

### 3.4.5 Summary

So far, we have discussed recent advances in different aspects, which, in sum, contribute to the latest progress on current reading comprehension benchmarks (esp. SQUAD). Which components are more important than the others? Do we need to add up all of these? Are these recent advances able to generalize to other reading comprehension tasks? How are they correlated with different capacities of language understanding? We think there isn't a clear answer to most of these questions yet and it still requires a lot of investigation.

We compiled the improvements of different components on SQUAD in Table 3.7. We would like to caution readers that these numbers are really not directly comparable, as they are built on different model architectures and different implementations. We hope that this table at least reflects some ideas regarding the importance of these components on

the SQuAD dataset. As is seen, all these components contribute to the final performance, more or less. The most important innovation is probably the use of contextualized word embeddings (e.g., ELMo), while the formulation of attention functions is also crucial. It will be important to investigate whether these advances can generalize to other reading comprehension tasks in the future.

# Chapter 4

# The Future of Reading Comprehension

In the previous chapter, we have described how neural reading comprehension models succeeded in current reading comprehension benchmarks and their key insights. Despite its rapid progress, there is still a long way to go towards genuine human-level reading comprehension. In this chapter, we will discuss future work and open questions.

We first examine the error cases of existing models in Section 4.1, and conclude that they still fail on "easy" or "trivial" cases despite their high accuracies on average.

As we discussed earlier, the success of recent reading comprehension is attributed to both the creation of large-scale datasets and the development of neural reading comprehension models. In the future, we believe both components will be still equally important. We discuss the future work of datasets and models respectively in Section 4.2 and 4.3. What is still missing in the existing datasets and models? How can we approach that?

Finally, we review several important research questions in this field in Section 4.4.

## 4.1 Is SQuAD Solved Yet?

Although we have already achieved super-human performance on the SQuAD dataset, does it indicate that our reading comprehension models are capable of solving all the SQuAD examples or any examples with the same level of difficulty?

Figure 4.1 demonstrates some failure cases of our Stanford Attentive Reader model described in Section 3.2. As we can see, the model predicts the answer type perfectly

for all these examples: it predicts a number for the question *what is the total number of . . . ?* and *what is the population . . . ?* and a team name for the question *Which team won Super Bowl 50?*. However, the model failed to understand the subtleties expressed in the text and can't distinguish among the candidate answers. In more detail,

(a) The number *2,400* modifies *professors, lecturers, and instructors* while *7,200* modifies *undergraduates*. However, the system failed to identify that and we believe that linguistic structures (e.g., syntactic parsing) can help resolve this case.

(b) Both teams *Denver Broncos* and *Carolina Panthers* are modified by the word *champion*, but the system failed to infer that "X defeated Y" so "X won".

(c) The system predicted *100,000* probably because it is closer to the word *population*. However, to answer the question correctly, the system has to identify that *3.7 million* is the population of *Los Angles*, and *1.3 million* is the population of *San Diego* and compare the two numbers and infer that *1.3 million* is the answer because it is *second largest*. This is a difficult example and probably beyond the scope of all the existing systems.

We also took a closer look at the predictions of the best SQuAD model so far — an ensemble of 7 BERT models (Devlin et al., 2018). As is demonstrated in Figure 4.2, we can see that this strong model still makes some simple mistakes which humans hardly make. It is fair to conjecture that these models have been doing very sophisticated matching of text while they still have difficulty understanding the inherent structure between entities and the events expressed in the text.

Lastly, Jia and Liang (2017) find that if we add a distracting sentence to the end of the passage (see an example in Figure 4.3), the average performance of current reading comprehension systems will drop drastically from 75.4% to 36.4%. These distracting sentences have word overlap with the question while not actually contradict the correct answer and not mislead human understanding. The performance is even worse if the distracting sentence is allowed to add ungrammatical sequences of words. These results suggest that 1) The current models strongly depend on the lexical cues between the passage and the question. That's why the distracting sentences can be so destructive; 2) Even though the

| | |
|---|---|
| (a) | **Question**: What is the total number of professors, instructors, and lecturers at Harvard?<br>**Passage**: Harvard's 2,400 professors, lecturers, and instructors instruct 7,200 undergraduates and 14,000 graduate students. The school color is crimson, which is also the name of the Harvard sports teams and the daily newspaper, The Harvard Crimson. The color was unofficially adopted (in preference to magenta) by an 1875 vote of the student body, although the association with some form of red can be traced back to 1858, when Charles William Eliot, a young graduate student who would later become Harvard's 21st and longest-serving president (1869—1909), bought red bandanas for his crew so they could more easily be distinguished by spectators at a regatta.<br>**Gold answer**: 2,400<br>**Predicted answer**: 7,200 |
| (b) | **Question**: Which team won Super Bowl 50?<br>**Passage**: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.<br>**Gold answer**: Denver Broncos<br>**Predicted answer**: Carolina Panthers |
| (c) | **Question**: What is the population of the second largest city in California?<br>**Passage**: Los Angeles (at 3.7 million people) and San Diego (at 1.3 million people), both in southern California, are the two largest cities in all of California (and two of the eight largest cities in the United States). In southern California there are also twelve cities with more than 200,000 residents and 34 cities over 100,000 in population. Many of southern California's most developed cities lie along or in close proximity to the coast, with the exception of San Bernardino and Riverside.<br>**Gold answer**: 1.3 million<br>**Predicted answer**: 100,000 |

Figure 4.1: Several failure cases of our model on SQuAD. Gold answers are marked as blue and predicted answers are marked as red.

| | |
|---|---|
| (d) | **Question**: What is the least number of members a board of trustees can have?<br>**Passage**: The Book of Discipline is the guidebook for local churches and pastors and describes in considerable detail the organizational structure of local United Methodist churches. All UM churches must have a board of trustees with at least three members and no more than nine members and it is recommended that no gender should hold more than a 2/3 majority. All churches must also have a nominations committee, a finance committee and a church council or administrative council. Other committees are suggested but not required such as a missions committee, or evangelism or worship committee. Term limits are set for some committees but not for all. The church conference is an annual meeting of all the officers of the church and any interested members. This committee has the exclusive power to set pastors' salaries (compensation packages for tax purposes) and to elect officers to the committees.<br>**Gold answer**: three<br>**Predicted answer**: nine |
| (e) | **Question**: Where does centripetal force go?<br>**Passage**: where is the mass of the object, is the velocity of the object and is the distance to the center of the circular path and is the unit vector pointing in the radial direction outwards from the center. This means that the unbalanced centripetal force felt by any object is always directed toward the center of the curving path. Such forces act perpendicular to the velocity vector associated with the motion of an object, and therefore do not change the speed of the object (magnitude of the velocity), but only the direction of the velocity vector. The unbalanced force that accelerates an object can be resolved into a component that is perpendicular to the path, and one that is tangential to the path. This yields both the tangential force, which accelerates the object by either slowing it down or speeding it up, and the radial (centripetal) force, which changes its direction.<br>**Gold answer**: the center of the curving path<br>**Predicted answer**: changes its direction |
| (f) | **Question**: How many times have the Panthers been in the Super Bowl?<br>**Passage**: The Panthers finished the regular season with a 15–1 record, and quarterback Cam Newton was named the NFL Most Valuable Player (MVP). They defeated the Arizona Cardinals 49–15 in the NFC Championship Game and advanced to their second Super Bowl appearance since the franchise was founded in 1995. The Broncos finished the regular season with a 12–4 record, and denied the New England Patriots a chance to defend their title from Super Bowl XLIX by defeating them 20–18 in the AFC Championship Game. They joined the Patriots, Dallas Cowboys, and Pittsburgh Steelers as one of four teams that have made eight appearances in the Super Bowl.<br>**Gold answer**: second<br>**Predicted answer**: eight |

Figure 4.2: Several failure cases of the currently best model (BERT ensemble) on SQuAD. Gold answers are marked as blue and predicted answers are marked as red.

**Question**: What is the name of the quarterback who was 38 in Super Bowl XXXIII?
**Passage**: Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. *Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*

Figure 4.3: An adversarial example used in (Jia and Liang, 2017), where a distracting sentence is added to the end of the passage (italicized). Blue: the correct answer and red: the predicted answer.

models achieved a high accuracy on the original development set, they are really not robust to the adversarial examples. This is a critical problem of the standard supervised learning paradigm and it makes existing models difficult to deploy in the real world. We will discuss the property of robustness more in Section 4.3.

To sum up, we believe that, although a very high accuracy was already obtained on the SQuAD dataset, the current models only focus on the surface-level information of the text, and still make simple errors when it comes to a (slightly) deeper level of understanding. On the other hand, the high accuracies also indicate that most of the SQuAD examples are rather easy and require little understanding. There are difficult examples which require complex reasoning in SQuAD (for example, (c) in Figure 4.1), but due to their scarcity, their accuracies aren't really reflected in the averaged metric. Furthermore, the high accuracies only hold when training and development come from the same distribution, and it still remains a severe problem when they differ. In the next two sections, we discuss the possibilities of creating more challenging datasets and building more effective models.

## 4.2 Future Work: Datasets

We have mostly focused on CNN/Daily Mail and SQuAD and demonstrated that both 1) neural models are able to achieve either super-human or the ceiling performance on them; 2) although these datasets are highly useful, most of the examples are rather simple and don't require much reasoning yet. What desired properties are still missing in

these datasets? What kind of datasets should we work on next? And how to collect better datasets?

We think that datasets like SQuAD mainly have the following limitations:

- The questions are *posed based on the passage*. That said, if a questioner is looking at the passage while they ask a question, they are quite likely to mirror the sentence structure and to reuse the same words. This eases the difficulty of answering questions as many questions words are overlapping with the passage words.

- It only allows questions that are *answerable by a single span in the passage*. This not only implies all the questions are answerable, but also excludes many possible questions to be posed such as *yes/no*, *counting* questions. As we discussed earlier, most of the questions in SQuAD are factoid questions and the answers are generally short (3.1 tokens on average). Therefore, there are also very few *why* (cause and effect) and *how* (procedure) questions in the dataset.

- Most of the questions can be answered by *a single supporting sentence* in the passage and don't require multiple-sentence reasoning. Rajpurkar et al. (2016) estimated that only $13.6\%$ of the examples need multiple sentence reasoning. Among them, we think that most of the cases are resolving conferences, which might be solved by a coreference system.

To address these limitations, there have been a number of new datasets collected recently. They follow a similar paradigm of SQuAD but are constructed in various ways. Table 4.1 gives an overview of a few representative datasets. As we can see, these datasets are of a similar order of magnitude (ranging from 33k to 529k training examples), and there is still a gap between the state-of-the-art and the human performance (some gaps are bigger than the others though). In the following, we describe these datasets in detail and discuss how they tackle the aforementioned limitations and their advantages/disadvantages:

[1]Joshi et al. (2017) provided oracles scores of *exact match* accuracies of 82.8% and 83.0% of the Web and Wikipedia domain respectively. These numbers measure the percentage of examples that answer can be found in the documents and differ from human performance.

[2]In contrast to the Web domain of TriviaQA, the Wikipedia domain is evaluated over questions instead of documents.

[3]We only list the setting where the summaries are given.

[4]We only list the "distractor" setting.

| Dataset | #Train | #Dev | #Test | Domain | Metric | Human | SOTA |
|---|---|---|---|---|---|---|---|
| TRIVIAQA (Web) | 528,979 | 68,621 | 65,059 | Web | F1 | N/A[1] | 71.3 |
| TRIVIAQA (Wiki.)[2] | 61,888 | 9,951 | 9,509 | Wikipedia | F1 | N/A | 68.9 |
| RACE | 87,866 | 4,887 | 4,934 | Exams | Accuracy | 100.0 | 59.0 |
| NARRATIVEQA[3] | 32,747 | 3,461 | 10,557 | Wikipedia | ROUGE-L | 57.0 | 36.3 |
| SQUAD 2.0 | 130,319 | 11,873 | 8,862 | Wikipedia | F1 | 89.5 | 83.1 |
| HOTPOTQA[4] | 90,564 | 7,405 | 7,405 | Wikipedia | F1 | 91.4 | 59.0 |

Table 4.1: A summary of more recent reading comprehension datasets. We only show the F1 results for span-prediction tasks and ROUGE-L for free-form answer tasks. The state-of-the-art results are taken from Clark and Gardner (2018) for TRIVIAQA (Joshi et al., 2017), Radford et al. (2018) for RACE (Lai et al., 2017), Kočiskỳ et al. (2018) for NARRATIVEQA, Devlin et al. (2018) for SQUAD 2.0 (Rajpurkar et al., 2018) and Yang et al. (2018) for HOTPOTQA.

**TriviaQA (Joshi et al., 2017).** The key idea of this dataset is that question/answer pairs were collected *before* constructing the corresponding passages. More specifically, they gathered 95k question-answer pairs from trivia and quiz-league websites and collected textual evidence which contained the answer from either Web search results or Wikipedia pages corresponding to the entities which are mentioned in the question. As a result, they collected 650k (passage, question, answer) triples in total. This paradigm effectively solved the problem that questions were dependent on the passage and also it is easier to construct a large dataset cheaply. It is worth noting that the passages used in this dataset are mostly long documents (the average document length is 2,895 words and it is 20 times longer than that of SQUAD), and also posed a challenge of scalability for existing models. However, it has a similar problem to the CNN/DAILY MAIL dataset — as the dataset was curated heuristically, there is no guarantee that the passage really provides the answer to the question and this influences the quality of the training data.

**RACE (Lai et al., 2017).** Humans' standardized tests are a proper way to evaluate machines' reading comprehension abilities. RACE is a multiple choice dataset collected from the English exams for middle-school and high-school Chinese students within the 12—18 age range. All the questions and answer options were created by experts. As a result, the dataset is more difficult than most existing datasets and it was estimated that 26% of the

questions require multiple sentence reasoning. The state-of-the-art performance is only 59% so far (each question has 4 candidate answers).

**NarrativeQA (Kočiskỳ et al., 2018).** This is a challenging dataset and it required crowdworkers to ask questions based on the plot summaries of a book or a movie from Wikipedia. The answers are free-form human-generated text and in particular, the annotators were encouraged to use their own words and copying is not allowed in the interface. The plot summaries usually contain more characters and events and more complex to follow than news articles or Wikipedia paragraphs. The dataset consists of two settings: one is to answer questions base on the summary (659 tokens on average) which is more similar to SQuAD, and the other is to answer questions based on the full book or movie script (62,528 tokens on average). The second setting is especially difficult, as it requires IR components to locate relevant information in the long documents. One problem with this dataset is that human agreement is low due to its free-form answer form and thus it is difficult to evaluate.

**SQuAD 2.0 (Rajpurkar et al., 2018).** SQuAD 2.0 proposed to add 53,775 negative examples to the original SQuAD dataset. These questions are not answerable from the passage, but look similar to the positive ones (relevant and the passage contains a plausible answer). To work well on the dataset, systems need to not only answer questions but also determine when no answer is supported by the paragraph and abstain from answering. This is an important aspect in practical applications but has been omitted in previous datasets.

**HotpotQA (Yang et al., 2018).** This dataset aims to construct questions which need multiple supporting documents to answer. To approach this, the crowdworkers were required to ask questions based on two relevant Wikipedia paragraphs (there is a hyperlink from the first paragraph of one article to the other). It also offers a new type of factoid comparison question, for which systems need to compare two entities on some shared properties. The dataset consists of two settings for evaluation – one is called the *distractor* setting in which each question is provided 10 passages, including the two passages used for constructing the question and 8 distractor passages retrieved from Wikipedia; the second setting is to use the full Wikipedia to answer the question.

Compared to SQuAD, these datasets either require more complex reasoning cross sentences or documents, or need to handle longer documents, or need to generate free-form answers instead of extracting a single span, or predict when there is no answer in the passage. They posed new challenges and many are still beyond the scope of existing models. We believe that these datasets will further inspire a series of modeling innovations in the future. After our models can reach the next level of performance, we will need to set out to construct even more difficult datasets to solve.

## 4.3 Future Work: Models

Next we turn to the research directions of models for future work. We first describe the desiderata of reading comprehension models. Most of the existing work only focuses on *accuracy* — given a standard training/development/testing split of a dataset, the major goal is to get the best accuracy score on the testing set. However, we argue that there are other important aspects which have been overlooked that we will need to work on in the future, including *speed and scalability*, *robustness* and *interpretability*. Lastly, we discuss what important elements are still missing in the current models, to solve more difficult reading comprehension problems.

### 4.3.1 Desiderata

Besides *accuracy* (achieving a better performance score on a standard dataset), the following desiderata are also very important for future work:

**Speed and Scalability.** How to build faster models (for both training and inference) and how to scale to longer documents is an important direction to pursue. Building faster models for training can lead to lower turnaround time for experimentation and also enable us to train on bigger datasets. Building faster models for inference is highly useful when we deploy the models in practical use. Also, it is unrealistic to encode a very long document (e.g., TriviaQA) or even a full book (e.g., NarrativeQA) using an RNN and this still remains a severe challenge. For example, the average document length of TriviaQA is

2,895 tokens and the authors truncated the documents to the first 800 tokens for the sake of scalability. The average document length of NARRATIVEQA is 62,528 tokens and the authors have to first retrieve a small number of relevant passages from the story using an IR system.

Existing solutions to these problems include:

- Replacing LSTMs with non-recurrent models such as TRANSFORMER (Vaswani et al., 2017) or lighter recurrent units such as SRU (Lei et al., 2018) as we discussed in Section 3.4.3.
- Training models which learn to skip part of the documents so they don't need to read all of the content. These models can run much faster while still retaining a similar performance. Representative works in this line include Yu et al. (2017) and Seo et al. (2018).
- The choice of optimization algorithms can also greatly affect the convergence speed. Multi-GPU training and hardware performance are also important aspects to consider but they are beyond the scope of this thesis. Coleman et al. (2017) provide a benchmark[5] which measures the end-to-end training and inference time to achieve a state-of-the-art accuracy level for a wide range of tasks, including SQUAD.

**Robustness.** We discussed in Section 4.1 that existing models are very brittle to adversarial examples which will become a severe problem when we deploy these models in the real world. Moreover, most of the current works follow the standard paradigm: training and evaluating on the splits of one dataset. It is known that if we train our models on one dataset and evaluate on another dataset, the performance will drop dramatically due to their different source of text and construction methods. For future work, we need to consider:

- How to create better adversarial training examples and incorporate them into the training process.
- Researching more on transfer learning and multi-task learning, so that we can build models with high performance across various datasets.

[5] https://dawn.cs.stanford.edu/benchmark/

- We might need to break the standard paradigm of supervised learning, and think about how to create better ways of evaluating our current models for the sake of building more robust models.

**Interpretability.** The last important aspect is *interpretability* and it has been mostly ignored in the current systems. Our future systems should not only be able to provide the final answers, but also provide the rationales behind their predictions, so users can decide if they can trust the outputs and leverage them or not. Neural networks are especially notorious for the fact that the end-to-end training paradigm makes these models like a black box and it is hard to interpret their predictions. This is especially crucial if we want to apply these systems to medical or legal domains.

Interpretability can have different definitions. In our context, we think there could be several ways to approach that:

- The easiest way is to require the models to learn to extract input pieces from the documents as supporting evidence. This has been studied before (e.g., (Lei et al., 2016)) for sentence classification problems but not yet in reading comprehension problems.

- A more complex way is that the models can indeed generate rationales. Instead of only highlighting the relevant piece of information in the passage, the models need to interpret how these pieces are connected and finally get to the answer. Take Figure 4.1 (c) as an example, the systems need to interpret that the two cities are the two largest and 3.7 million is bigger than 1.3 million thus it is the second largest. We think this desiderata is very important but far beyond the scope of current models.

- Finally, another important aspect to consider is what training resources we can get to approach this level of interpretability. Inferring rationales from the final answers is feasible but quite difficult. We should consider collecting human explanations as the supervision of training rationales in the future.

### 4.3.2 Structures and Modules

In this section, we are going to discuss what are the missing elements in the current models, if we want to solve more difficult reading comprehension problems.

First of all, current models are all built on either sequence models or tackle all pairs of words symmetrically (e.g., TRANSFORMER), and omit the inherent structure of language. On the one hand, this forces our models to learn all the relevant linguistic information from scratch, which makes the learning of our models more difficult. On the other hand, the NLP community has put a lot of effort into studying linguistic representation tasks (e.g., syntactic parsing, coreference) and building many linguistic resources and tools for years. Language encodes meaning in terms of hierarchical, nested structures on sequences of words. Would it be still useful to encode linguistic structures more explicitly in our reading comprehension models?

Figure 4.4 illustrates the CORENLP (Manning et al., 2014) output of several examples in SQUAD. We believe that this structural information would be useful as follows:

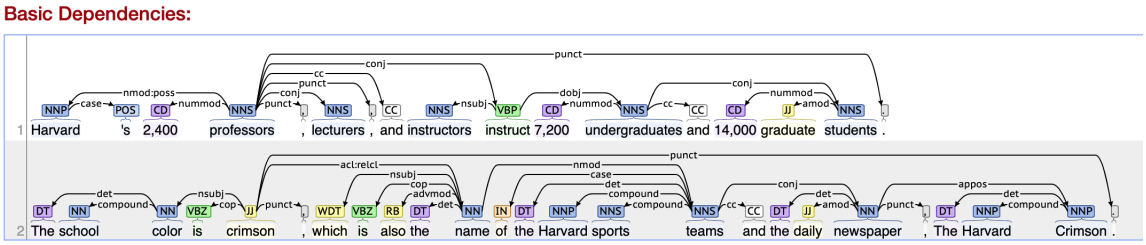
(a) The information that *2,400* is a *numeric modifier* of *professors* should help answer the question *What is the total number of professors, instructors, and lecturers at Harvard?* (We have seen this example as an error case in Figure 4.1).

(b) The coreference information that *it* refers to *Harvard* should help answer the question *Starting in what year has Harvard topped the Academic Rankings of World Universities?*.

Therefore, we think that these linguistic knowledge/structures would be still a useful addition to the current models. The remaining questions that we need to answer are: 1) What are the best ways to incorporate these structures into sequence models? 2) Do we want to model the structures as a latent variable or rely on off-the-shelf linguistic tools? For the latter case, are the current tools good enough so that the models can benefit more (rather than suffering from noise)? Can we further improve the performance of these representation tasks?

Another aspect we think is still missing from most existing models is *modules*. The task of reading comprehension is inherently very complex and different types of examples

(a)




(b)


Figure 4.4: Example output of CORENLP: (a) dependencies and (b) coreference. The image is taken from http://corenlp.run.

require different types of reasoning capabilities. It still remains a grand challenge if we want to learn everything through a giant neural network (This is reminiscent of why the attention mechanism was proposed because we don't want to squash the meaning of a sentence or a paragraph into one vector!). We believe that, if we want to approach deeper level of reading comprehension, our future models will be more structured, more modularized, and solving one comprehensive task can be decomposed into many subproblems and we can tackle each smaller subproblem (e.g., each reasoning type) separately and combine all of them in the end.

The idea of *modules* has been implemented in NEURAL MODULE NETWORKS (NMN) (Andreas et al., 2016) before. They first perform a dependency parse of the question, and then decompose the question answering problem into several "modules" based on the parse structure. One example they used for a visual question answering (VQA) task is: a question "What color is the bird?" can be decomposed as two modules. One module is used to detect the bird in the given image, and another module is used to detect the color of the found region (bird). We believe that this sort of approach holds promise to answer questions such as *What is the population of the second largest city in California?* (Figure 4.1 (c)). However, NMN has only been studied on visual question answering or small knowledge-base question question problems so far, and applying to reading comprehension problems

can be more challenging due to the flexibility of language variations and question types.

## 4.4 Research Questions

In the last section, we discuss a few central research questions in this field, which still remain as open questions and yet to be answered in the future.

### 4.4.1 How to Measure Progress?

The first question is: *How can we measure the progress of this field?* The evaluation metrics are certainly clear indicators of measuring progress on our reading comprehension benchmarks. Does this indicate that we make real progress on reading comprehension in general? How can we tell if some progress on one benchmark can generalize to others? How about if model $A$ works better than model $B$ on one dataset, while model $B$ works better on the other dataset? How to tell how far these computer systems are sill from genuine human-level reading comprehension?

On the one hand, we think that taking human's standardized tests could be a good strategy for evaluating the performance of machine reading comprehension systems. These questions are usually carefully curated and designed to test human's reading comprehension abilities at different levels. To get computer systems aligned with human measurements is a proper way in building natural language understanding systems.

On the other hand, we believe that it would be desirable to integrate many reading comprehension datasets as a testing suite for evaluation in the future, instead of only testing on one single dataset. This will help us better distinguish what are genuine progress for reading comprehension and what might be just overfitting to one specific dataset.

More importantly, we need to understand our existing datasets better: characterizing their quality and what skills are required to answer the questions. This will be a crucial step for building more challenging datasets and analyzing the behavior of our models. Besides our work on analyzing the CNN/DAILY MAIL examples in Chen et al. (2016), Sugawara et al. (2017) attempted to separate reading comprehension skills into two disjoint sets: *prerequisite skills* and *readability*. Prerequisite skills measure different types of

reasoning and knowledge required to answer the question and 13 skills are defined: object tracking, mathematical reasoning, coreference resolution, logical reasoning, analogy, causal relation, spatiotemporal relation, ellipsis, bridging, elaboration, meta-knowledge, schematic clause relation and punctuation. Readability measures the "text ease of processing", and a wide range of linguistic features/human readability measurements are used. The authors concluded that these two sets are weakly correlated and it is possible to design difficult questions from the contexts that are easy to read. These studies suggest that we could construct datasets and develop models based on these properties separately.

In addition, Sugawara et al. (2018) designed a few simple filtering heuristics and divided the examples from many existing datasets into a hard subset and an easy subset, based on 1) whether the question can be answered using only the first few words; 2) whether the answer is contained in the most similar sentence in the passage. They observed that the baseline performances for the hard subsets remarkably degrade compared to those of the entire datasets. Moreover, Kaushik and Lipton (2018) analyzed the performance of existing models using passage-only or question-only information, and found that these models sometimes can work surprisingly well and hence there exists annotation artifacts in some of the existing datasets.

In conclusion, we believe that if we want to make steady progress on reading comprehension in the future, we will have to answer these basic questions about the difficulty of examples first. Understanding what is required for the datasets, what our current systems can do and can't do will help us identify the challenges we are facing and measure the progress.

### 4.4.2 Representations vs. Architecture: Which is More Important?

The second important question is to understand the role of representations vs. architectures to the performance of reading comprehension models. Since SQuAD was created, there has been a recent trend of increasing the complexity of neural architectures. In particular, more and more complex attention mechanisms have been proposed to capture the similarity between the passage and the question (Section 3.4.2). However, recent works (Radford et al., 2018; Devlin et al., 2018) proposed that if we can pretrain a deep language model

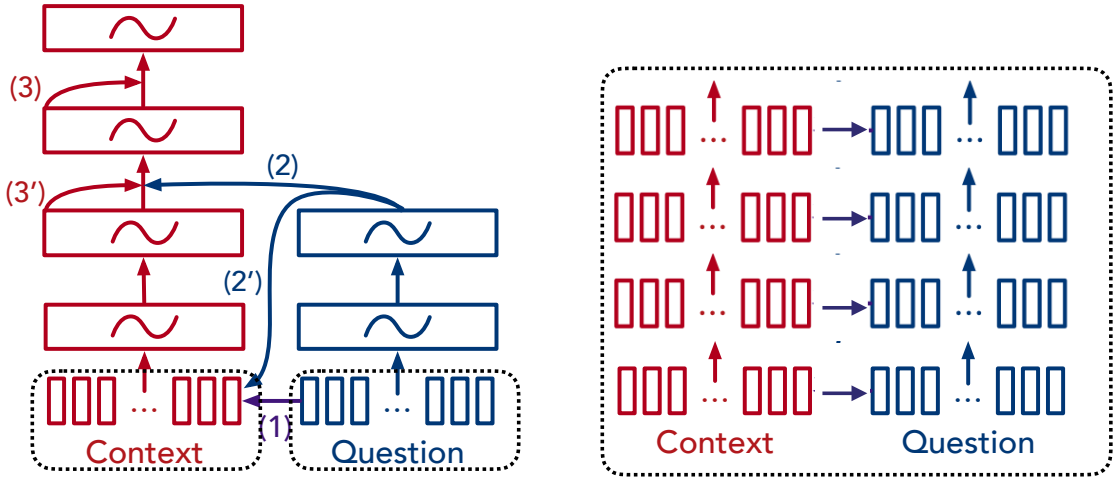



Figure 4.5: A comparison of a complex architecture (left) vs. a simple architecture with pre-training (right). The parameters in the dashed box can be pre-trained from unlabeled text, while all the remaining parameters are initialized randomly and learned from the reading comprehension datasets.

on large text corpora, a simple model which takes the concatenation of the question and the passage without modeling any direct interactions between the two can work extremely well on reading comprehension datasets such as SQuAD and RACE (see Table 3.4 and Table 4.1).

As illustrated in Figure 4.5, the first class of models (left) only builds on top of word embeddings (each word type has a vector representation) pre-trained from unlabeled text, while all the remaining parameters (including all the weights to compute various attention functions) need to be learned from the limited training data. The second class of models (right) makes the model architecture very simple and it only models the question and passage as a single sequence. The whole model is pre-trained and all the parameters are kept. Only a few new parameters are added (e.g., the parameters for predicting the start and end positions for SQuAD) and the other parameters will be fine-tuned on the training set of the reading comprehension tasks.

We think these two classes of models indicate two extremes. On the one hand, it certainly demonstrates the incredible power of unsupervised representations. As we have a powerful language model pre-trained from huge amount of text, the model already encodes

a great deal of properties about language while a simple model which concatenates the passage and the question is sufficient to learn the dependency between the two. On the other hand, when only word embeddings are given, it seems that modeling the interactions between the passage and the question carefully (or giving the model more prior knowledge) helps. In the future, we suspect that we will need to combine the two and a model like BERT is too coarse to handle the examples which require complex reasoning.

### 4.4.3 How Many Training Examples Are Needed?

The third question is *how many training examples are actually needed?* We have discussed many times that the success of neural reading comprehension is driven by large-scale supervised datasets. All the datasets that we have been actively working on contain at least 50,000 examples. Can we always embrace data abundance and further improve the performance of our systems? Is it possible to train a neural reading comprehension model with only hundreds of annotated examples today?

We think there isn't a clear answer yet. On the one hand, there is clear evidence that having more data helps. Bajgar et al. (2016) demonstrated that inflating the cloze-style training data constructed from books available through project Gutenberg can provide a boost of 7.4%–14.8% on the CHILDREN BOOK TEST (CBT) dataset (Hill et al., 2016) using the same model. We discussed before that using data augmentation techniques (Yu et al., 2018) or augmentating the training data with TRIVIAQA can help improve the performance on SQUAD (# training examples = 87,599).

On the other hand, pre-trained (language) models (Radford et al., 2018; Devlin et al., 2018) can help us reduce the dependence on large-scale datasets. In these models, most of the parameters are already pretrained on abundant unlabeled data and will be only fine-tuned during training.

In the future, we should encourage more research on unsupervised learning and transfer learning. Leveraging unlabeled data (e.g., text) or other cheap resources or supervision (e.g., datasets like CNN/DAILY MAIL) will relieve us from collecting expensive annotated data. We also should seek better and cheaper ways of collecting supervised datasets.

# Part II

# Neural Reading Comprehension: Applications

# Chapter 5

# Open Domain Question Answering

In PART I, we described the task of reading comprehension: its formulation and development over recent years, the key components of neural reading comprehension systems, and future research directions. However, it is unclear yet whether reading comprehension is merely used as a task of measuring language understanding abilities, or it can enable any useful applications. In PART II, we will answer this question and discuss our efforts at building applications which leverage neural reading comprehension as their core component.

In this chapter, we view *open domain question answering* as an application of reading comprehension. Open domain question answering has been a long-standing problem in the history of NLP. The goal of open domain question answering is to build automated computer systems which are able to answer any sort of (factoid) questions that humans might ask, based on a large collection of unstructured natural language documents, structured data (e.g., knowledge bases), semi-structured data (e.g., tables) or even other modalities such as images or videos.

We are the first to test how the neural reading comprehension methods can perform in an open-domain QA framework. We believe that the high performance of these systems can be a key ingredient in building a new generation of open-domain question answering systems, when combined with effective information retrieval techniques.

This chapter is organized as follows. We first give a high-level overview of open domain question answering and some notable systems in the history (Section 5.1). Next, we

introduce an open-domain question answering system that we built called DRQA, designed to answer questions from English Wikipedia (Section 5.2). It essentially combines an information retrieval module and the high-performing neural reading comprehension module that we described in Section 3.2. We further talk about how we can improve the system by creating distantly-supervised training examples from the retrieval module. We then present a comprehensive evaluation on multiple question answering benchmarks (Section 5.3). Finally, we discuss current limitations, follow-up work and future directions in Section 5.4.

## 5.1 A Brief History of Open-domain QA

Question answering was one of the earliest tasks for NLP systems since 1960s. One early system, which prefigures modern text-based question answering systems, was the PROTOSYNTHEX system of (Simmons et al., 1964). The system first formulated a query based on the content words in the question, retrieved candidate answer sentences based on the frequency-weighted term overlap with the question, and finally performed a dependency parse match to get the final answer. Another notable system MURAX (Kupiec, 1993), was designed to answer general-knowledge questions over GROLIER's on-line encyclopedia, using shallow linguistic processing and information retrieval (IR) techniques.

The interest in open domain question answering has increased since 1999, when the QA track was first included as part of the annual TREC competitions[1]. The task was at first defined such that the systems were to retrieve small snippets of text that contained an answer for open-domain questions. It has spurred a wide range of QA systems developed at the time, and the majority of the systems consisted of two stages: an IR system used to select the top $n$ documents or passages which match a query that has been generated from the question, and a window-based word scoring system used to pinpoint likely answers. For more details, readers are referred to (Voorhees, 1999; Moldovan et al., 2000).

More recently, with the development of knowledge bases (KBs) such as FREEBASE (Bollacker et al., 2008) and DBPEDIA (Auer et al., 2007), many innovations have occurred in the context of question answering from KBs with the creation of resources like WEBQUESTIONS (Berant et al., 2013) and SIMPLEQUESTIONS (Bordes et al., 2015) based

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

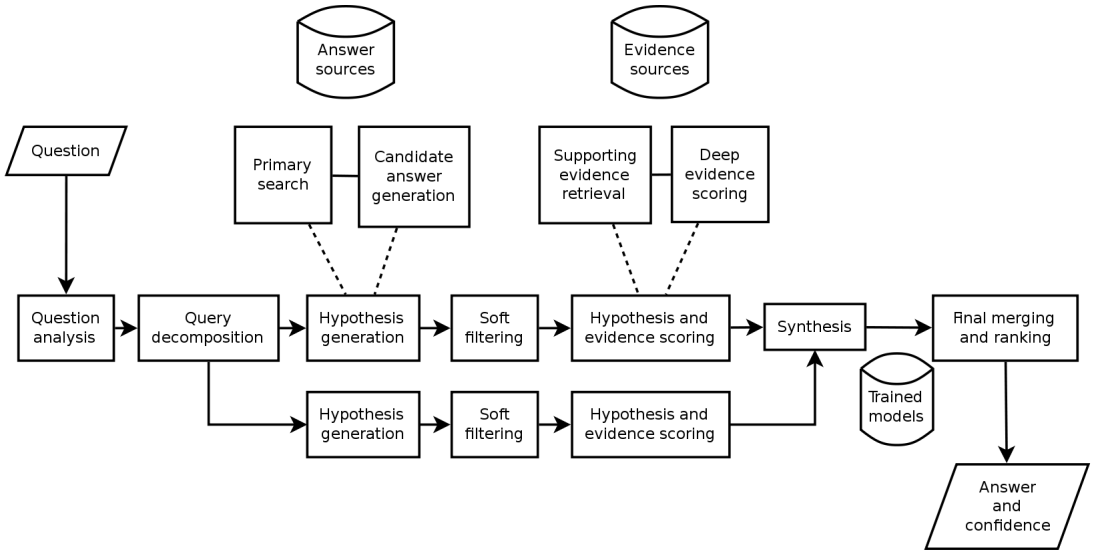



Figure 5.1: The high-level architecture of IBM's DEEPQA used in WATSON. Image courtesy: https://en.wikipedia.org/wiki/Watson_(computer).

on FREEBASE, or on automatically extracted KBs, e.g., OpenIE triples and NELL (Fader et al., 2014). A lot of progress has been made on knowledge-based question answering and the major approaches are either based on semantic parsing or information extraction techniques (Yao et al., 2014). However, KBs have inherent limitations (incompleteness and fixed schemas) that motivated researchers to return to the original setting of answering from raw text lately.

There are also a number of highly developed full pipeline QA approaches using a myriad of resources, including both text collections (Web pages, Wikipedia, newswire articles) and structured knowledge bases (FREEBASE, DBPEDIA etc.). A few notable systems include Microsoft's ASKMSR (Brill et al., 2002), IBM's DEEPQA (Ferrucci et al., 2010) and YODAQA (Baudiš, 2015) — the latter of which is open source and hence reproducible for comparison purposes. ASKMSR is a search-engine based QA system that relies on "data redundancy rather than sophisticated linguistic analyses of either questions or candidate answers". DEEPQA is the most representative modern question answering system and its victory at the TV game-show JEOPARDY! in 2011 received a great deal of attention. It is a very sophisticated system that consists of many different pieces in the pipeline and it relies on unstructured information as well as structured data to generate candidate answers

or vote over evidence. A high-level architecture is illustrated in Figure 5.1. YODAQA is an open source system modeled after DEEPQA, similarly combining websites, databases and Wikipedia in particular. Comparing against these methods provides a useful datapoint for an "upper bound" benchmark on performance.

Finally, there are other types of question answering problems based on different types of resources, including Web tables (Pasupat and Liang, 2015), images (Antol et al., 2015), diagrams (Kembhavi et al., 2017) or even videos (Tapaswi et al., 2016). We are not going into further details as our work focuses on text-based question answering.

Our DRQA system (Section 5.2) focuses on question answering using Wikipedia as the unique knowledge source, such as one does when looking for answers in an encyclopedia. QA using Wikipedia as a resource has been explored previously. Ryu et al. (2014) perform open-domain QA using a Wikipedia-based knowledge model. They combine article content with multiple other answer matching modules based on different types of semi-structured knowledge such as infoboxes, article structure, category structure, and definitions. Similarly, Ahn et al. (2004) also combine Wikipedia as a text resource with other resources, in this case with information retrieval over other documents. Buscaldi and Rosso (2006) also mine knowledge from Wikipedia for QA. Instead of using it as a resource for seeking answers to questions, they focus on validating answers returned by their QA system, and use Wikipedia categories for determining a set of patterns that should fit with the expected answer. In our work, we consider the comprehension of text only, and use Wikipedia text documents as the sole resource in order to emphasize the task of reading comprehension. We believe that adding other knowledge sources or information will further improve the performance of our system.

## 5.2 Our System: DRQA

### 5.2.1 An Overview

In the following we describe our system DRQA, which focuses on answering questions using English Wikipedia as the unique knowledge source for documents. We are interested in building a general-knowledge question answering system, which can answer any sort of

factoid questions where the answer is contained in and can be extracted from Wikipedia.

There are several reasons that we choose to use Wikipedia: 1) Wikipedia is a constantly evolving source of large-scale, rich, detailed information that could facilitate intelligent machines. Unlike knowledge bases (KBs) such as FREEBASE or DBPEDIA, which are easier for computers to process but too sparsely populated for open-domain question answering, Wikipedia contains up-to-date knowledge that humans are interested in. 2) Many reading comprehension datasets (e.g., SQUAD) are built on Wikipedia so that we can easily leverage these resources and we will describe it soon. 3) Generally speaking, Wikipedia articles are clean, high-quality and well-formed and thus they are highly useful resources for open domain question answering.

Using Wikipedia articles as the knowledge source causes the task of question answering (QA) to combine the challenges of both large-scale open-domain QA and of machine comprehension of text. In order to answer any question, one must first retrieve the few relevant articles among more than 5 million items, and then scan them carefully to identify the answer. This is reminiscent of how classical TREC QA systems worked, but we believe that neural reading comprehension models will play a crucial role of *reading* the retrieved articles/passages to obtain the final answer. As shown in Figure 5.2, our system essentially consists of two components: (1) the DOCUMENT RETRIEVER module for finding relevant articles and (2) a reading comprehension model, DOCUMENT READER, for extracting answers from a single document or a small collection of documents.

Our system treats Wikipedia as a collection of articles and does not rely on its internal graph structure. As a result, our approach is generic and could be switched to other collections of documents, books, or even daily updated newspapers. We detail the two components next.

### 5.2.2 Document Retriever

Following classical QA systems, we use an efficient (non-machine learning) document retrieval system to first narrow our search space and focus on reading only articles that are likely to be relevant. A simple inverted index lookup followed by term vector model

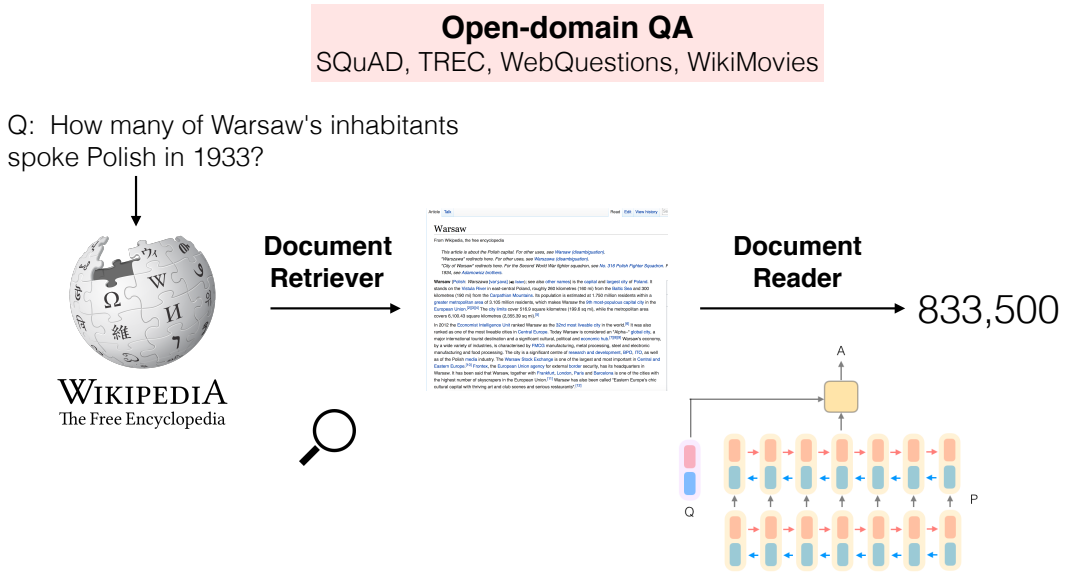



Figure 5.2: An overview of our question answering system DrQA.

scoring performs quite well on this task for many question types, compared to the built-in ElasticSearch based Wikipedia Search API (Gormley and Tong, 2015). Articles and questions are compared as TF-IDF weighted bag-of-word vectors.

We further improve our system by taking local word order into account with n-gram features. Our best performing system uses bigram counts while preserving speed and memory efficiency by using the hashing of (Weinberger et al., 2009) to map the bigrams to $2^{24}$ bins with an unsigned *murmur3* hash.

We use the DOCUMENT RETRIEVER as the first part of our full model, by setting it to return 5 Wikipedia articles given any question. Those articles are then processed by the DOCUMENT READER.

### 5.2.3 Document Reader

The DOCUMENT READER takes the top 5 Wikipedia articles and aims to read all the paragraphs and extracts the possible answers from them. This is exactly the setup as we did in span-based reading comprehension problems, and the STANFORD ATTENTIVE READER

model that we described in Section 3.2 can be directly plugged into this pipeline.

We apply our trained DOCUMENT READER for each single paragraph that appears in the top 5 Wikipedia articles and it predicts an answer span with a confidence score. To make scores compatible across paragraphs in one or several retrieved documents, we use the unnormalized exponential and take argmax over all considered paragraph spans for our final prediction. This is just a very simple heuristic and there are better ways to aggregate evidence over different paragraphs. We will discuss future work in Section 5.4.

### 5.2.4 Distant Supervision

We have built a complete pipeline which integrates a classical retrieval module and our previous neural reading comprehension component. The remaining key question is how can we train this reading comprehension module for the open-domain question answering setting?

The most direct approach is just to reuse the SQuAD dataset (Rajpurkar et al., 2016) as the training corpus, which was also built on top of Wikipedia paragraphs. However, this approach is limited in the following ways:

- As we discussed earlier in Section 4.2, the questions in SQUAD were crowdsourced after the annotators see the paragraphs to ensure they can be answered by a span in the passage. This distribution is quite specific and different from that of real-world question-answering when people have a question in mind first and try to find out he answers from the Web or other sources.

- Many SQUAD questions are indeed context-dependent. For example, a question is *What individual is the school named after?* posed on one passage of the Wikipedia article *Harvard University*, or another question is *What did Luther call these donations?* based on a passage that describes *Martin Luther*. Basically, these questions cannot be understood by themselves and thus are useless for open-domain QA problems. Clark and Gardner (2018) estimated around 32.6% of the questions in SQUAD are either document-dependent or passage-dependent.

- Finally, the size of SQuAD is rather small (80k training examples). It should further improve the system performance if wen can collect more training examples.

To overcome these problems, we propose a procedure to automatically create additional training examples from other question answering resources. The idea is to re-use the efficient information retrieval module that we built: if we already have a question answer pair $(q, a)$ and the retrieval module can help us find a paragraph relevant to the question $q$ and the answer segment $a$ appears in the paragraph, then we can create a *distantly-supervised* training example in the form of a $(p, q, a)$ triple for training the reading comprehension models:

$$f : (q, a) \Longrightarrow (p, q, a) \tag{5.1}$$
$$\text{if } p \in \text{ Document\_Retriever } (q) \text{ and } a \text{ appears in } p$$

This idea is a similar spirit to the popular approach of using distant supervision (DS) for relation extraction (Mintz et al., 2009) [2]. Despite that these examples can be noisy to some extent, it offers a cheap solution to create distantly supervised examples for open-domain question answering and will be a useful addition to SQuAD examples. We will describe the effectiveness of these distantly supervised examples in Section 5.3.

## 5.3 Evaluation

We have all the basic elements of our DrQA systems and let's take a look at the evaluation.

### 5.3.1 Question Answering Datasets

The first question is which question answering datasets we should evaluate on. As we discussed, SQuAD is one of the largest general purpose QA datasets currently available for question answering but it is very different from open-domain QA setting. We propose

[2] The idea for relation extraction is to pair textual mentions which contain the two entities which is known as a relation between them in an existing knowledge base.

to train and evaluate our system on other datasets developed for open-domain QA that have been constructed in different ways. We hence adopt the following three datasets:

**TREC** This dataset is based on the benchmarks from the TREC QA tasks that have been curated by Baudiš and Šedivỳ (2015). We use the large version, which contains a total of 2,180 questions extracted from the datasets from TREC 1999, 2000, 2001 and 2002.[3] Note that for this dataset, all the answers are written in regular expressions, for example, the answer is `Sept(ember)?|Feb(ruary)?` to the question *When is Fashion week in NYC?*, so answers *Sept*, *September*, *Feb*, *February* are all judged as correct.

**WebQuestions** Introduced in Berant et al. (2013), this dataset is built to answer questions from the Freebase KB. It was created by crawling questions through the GOOGLE SUGGEST API, and then obtaining answers using Amazon Mechanical Turk. We convert each answer to text by using entity names so that the dataset does not reference Freebase IDs and is purely made of plain text question-answer pairs.

**WikiMovies** This dataset, introduced in Miller et al. (2016), contains 96k question-answer pairs in the domain of movies. Originally created from the OMDB and MOVIELENS databases, the examples are built such that they can also be answered by using a subset of Wikipedia as the knowledge source (the title and the first section of articles from the movie domain).

We would like to emphasize that these datasets are not necessarily collected in the context of answering from Wikipedia. The TREC dataset was designed for text-based question answering (the primary TREC document sets consist mostly of newswire articles), while WEBQUESTIONS and WIKIMOVIES were mainly collected for knowledge-based question answering. We put all these resources in one unified framework, and test how well our system can answer all the questions — hoping that it can reflect the performance of general-knowledge QA.

Table 5.1 and Figure 5.3 give detailed statistics of these QA datasets. As we can see that, the distribution of SQUAD examples is quite different from that of the other QA datasets.

[3]This dataset is available at https://github.com/brmson/dataset-factoid-curated.

Due to the construction method, SQuAD has longer questions (10.4 tokens vs 6.7–7.5 tokens on average). Also, all these datasets have short answers (although the answers in SQuAD are slightly longer) and most of them are factoid.

Note that there are might be multiple answers for many of the questions in these QA datasets (see the *# answers* column of Table 5.1). For example, there are two valid answers: *English* and *Urdu* to the question *What language do people speak in Pakistan?* on WebQuestions. As our system is designed to return one answer, our evaluation considers the prediction as correct if it gives any of the gold answers.

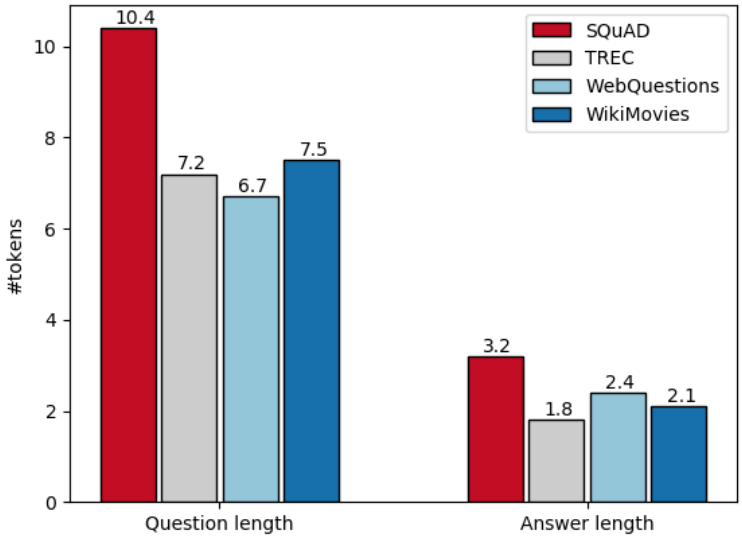



Figure 5.3: The average length of questions and answers in our QA datasets. All the statistics are computed based on the training sets.

[4]As all the answer strings are regex expressions, it is difficult to estimate # of answers. We only simply list the number of alternation symbols | in the answer.

| Dataset | # Train | # DS Train | # Test | # answers |
|---|---|---|---|---|
| SQuAD | 87,599 | 71,231 | N/A | 1.0 |
| TREC | 1,486† | 3,464 | 694 | 3.2[4] |
| WebQuestions | 3,778† | 4,602 | 2,032 | 2.4 |
| WikiMovies | 96,185† | 36,301 | 9,952 | 1.9 |

Table 5.1: Statistics of the QA datasets used for DrQA. DS Train: distantly supervised training data. †: These training sets are not used as is because no passage is associated with each question.

## 5.3.2 Implementation Details

### 5.3.2.1 Processing Wikipedia

We use the 2016-12-21 dump[5] of English Wikipedia for all of our full-scale experiments as the knowledge source used to answer questions. For each page, only the plain text is extracted and all structured data sections such as lists and figures are stripped.[6] After discarding internal disambiguation, list, index, and outline pages, we retain 5,075,182 articles consisting of 9,008,962 unique uncased token types.

### 5.3.2.2 Distantly-supervised data

We use the following process for each question-answer pair from the training portion of each dataset to build our distantly-supervised training examples. First, we run our Document Retriever on the question to retrieve the top 5 Wikipedia articles. All paragraphs from those articles without an exact match of the known answer are directly discarded. All paragraphs shorter than 25 or longer than 1500 characters are also filtered out. If any named entities are detected in the question, we remove any paragraph that does not contain them at all. For every remaining paragraph in each retrieved page, we score all positions that match an answer using unigram and bigram overlap between the question and a 20 token window, keeping up to the top 5 paragraphs with the highest overlaps. If there is no paragraph with non-zero overlap, the example is discarded; otherwise we add each found

[5] https://dumps.wikimedia.org/enwiki/latest

[6] We use the WikiExtractor script: https://github.com/attardi/wikiextractor.

| Dataset | Example | Article / Paragraph |
|---|---|---|
| TREC | **Q**: What U.S. state's motto is "Live free or Die"?<br>**A**: New Hampshire | **Article**: Live Free or Die<br>**Paragraph**: "Live Free or Die" is the official motto of the U.S. state of New Hampshire, adopted by the state in 1945. It is possibly the best-known of all state mottos, partly because it conveys an assertive independence historically found in American political philosophy and partly because of its contrast to the milder sentiments found in other state mottos. |
| WEBQUESTIONS | **Q**: What part of the atom did Chadwick discover?†<br>**A**: neutron | **Article**: Atom<br>**Paragraph**: ... The atomic mass of these isotopes varied by integer amounts, called the whole number rule. The explanation for these different isotopes awaited the discovery of the neutron, an uncharged particle with a mass similar to the proton, by the physicist James Chadwick in 1932. ... |
| WIKIMOVIES | **Q**: Who wrote the film Gigli?<br>**A**: Martin Brest | **Article**: Gigli<br>**Paragraph**: Gigli is a 2003 American romantic comedy film written and directed by Martin Brest and starring Ben Affleck, Jennifer Lopez, Justin Bartha, Al Pacino, Christopher Walken, and Lainie Kazan. |

Figure 5.4: Example training data from each QA dataset. In each case we show an associated paragraph where distant supervision (DS) correctly identified the answer within it, which is highlighted.

pair to our DS training dataset. Some examples are shown in Figure 5.4 and the number of distantly supervised examples we created for training are given in Table 5.1 (column *# DS Train*).

### 5.3.3 Document Retriever Performance

We first examine the performance of our retrieval module on all the QA datasets. Table 5.2 compares the performance of the two approaches described in Section 5.2.2 with that of

| Dataset | WIKI. SEARCH | DOCUMENT RETRIEVER | |
|---|---|---|---|
| | | unigram | bigram |
| TREC | 81.0 | 85.2 | **86.0** |
| WEBQUESTIONS | 73.7 | **75.5** | 74.4 |
| WIKIMOVIES | 61.7 | 54.4 | **70.3** |

Table 5.2: Document retrieval results. % of questions for which the answer segment appears in one of the top 5 pages returned by the method.

the Wikipedia Search Engine[7] for the task of finding articles that contain the answer given a question.

Specifically, we compute the ratio of questions for which the text span of any of their associated answers appear in at least one the top 5 relevant pages returned by each system.

Results on all datasets indicate that our simple approach outperforms Wikipedia Search, especially with bigram hashing. We also compare doing retrieval with Okapi BM25 or by using cosine distance in the word embeddings space (by encoding questions and articles as bag-of-embeddings), both of which we find performed worse.

### 5.3.4 Final Results

Finally, we assess the performance of our full system DRQA for answering open-domain questions using all these datasets. We compare three versions of DRQA which evaluate the impact of using distant supervision and multitask learning across the training sources provided to DOCUMENT READER (DOCUMENT RETRIEVER remains the same for each case):

- SQUAD: A single DOCUMENT READER model is trained on the SQUAD training set only and used on all evaluation sets. We used the model that we described in Section 5.2 (the F1 score is 79.0% on the test set of SQUAD).

- Fine-tune (DS): A DOCUMENT READER model is pre-trained on SQUAD and then fine-tuned for each dataset independently using its distant supervision (DS) training

[7] We use the Wikipedia Search API https://www.mediawiki.org/wiki/API:Search.

| Dataset | YodaQA | DrQA | | | DrQA* | |
|---|---|---|---|---|---|---|
| | | SQuAD | FT | MT | SQuAD | FT |
| TREC | 31.3 | 19.7 | 25.7 | 25.4 | 21.3 | 28.8 |
| WEBQUESTIONS | 38.9 | 11.8 | 19.5 | 20.7 | 14.2 | 24.3 |
| WIKIMOVIES | N/A | 24.5 | 34.3 | 36.5 | 31.9 | 46.0 |

Table 5.3: Full Wikipedia results. Top-1 exact-match accuracy (%). **FT**: Fine-tune (DS). **MT**: Multitask (DS). The DRQA* results are taken from Raison et al. (2018).

set.

- Multitask (DS): A single DOCUMENT READER model is jointly trained on the SQuAD training set and all the distantly-supervised examples.

For the full Wikipedia setting we use a streamlined model that does not use the CORENLP parsed $f_{token}$ features or lemmas for $f_{exact\_match}$. We find that while these help for more exact paragraph reading in SQUAD, they don't improve results in the full setting. Additionally, WEBQUESTIONS and WIKIMOVIES provide a list of candidate answers (1.6 million FREEBASE entity strings for WEBQUESTIONS and 76k movie-related entities for WIKIMOVIES) and we restrict that the answer span must be in these lists during prediction.

Table 5.3 presents the results. We only consider top-1, exact-match accuracy, which is the most restricted and challenging setting. In the original paper (Chen et al., 2017), we also evaluated the question/answer pairs in SQuAD. We omit them here because that at least a third of these questions are context-dependent and are not really suitable for open QA.

Despite the difficulty of the task compared to the reading comprehension task (where you are given the right paragraph) and unconstrained QA (using redundant resources), DRQA still provides reasonable performance across all four datasets.

We are interested in a single, full system that can answer any question using Wikipedia. The single model trained only on SQUAD is outperformed on all the datasets by the multi-task model that uses distant supervision. However, performance when training on SQuAD

alone is not far behind, indicating that task transfer is occurring. The majority of the improvement from SQuAD to Multitask (DS) learning, however, is likely not from task transfer, as fine-tuning on each dataset alone using DS also gives improvements, showing that is the introduction of extra data in the same domain that helps. Nevertheless, the best single model that we can find is our overall goal, and that is the Multitask (DS) system.

We compare our system to YodaQA (Baudiš, 2015) (an unconstrained QA system using redundant resources), giving results which were previously reported on TREC and WebQuestions.[8] Despite the increased difficulty of our task, it is reassuring that our performance is not too far behind on TREC (31.3 vs 25.4). The gap is slightly bigger on WebQuestions, likely because this dataset was created from the specific structure of Freebase which YodaQA uses directly.

We also include the results from an enhancement of our model named DrQA*, presented in Raison et al. (2018). The biggest change is that this reading comprehension model is trained and evaluated directly on the Wikipedia articles instead of paragraphs (documents are on average 40 times larger than individual paragraphs). As we can see, the performance has been improved consistently on all the datasets, and the gap from YodaQA is hence further reduced.

[8]The results are extracted from https://github.com/brmson/yodaqa/wiki/Benchmarks.

| | | |
|---|---|---|
| (a) | **Question** | What is question answering? |
| | **Answer** | a computer science discipline within the fields of information retrieval and natural language processing |
| | **Wiki. article** | Question Answering |
| | **Passage** | Question Answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language. |
| (b) | **Question** | Which state is Stanford University located in? |
| | **Answer** | California |
| | **Wiki. article** | Stanford Memorial Church |
| | **Passage** | Stanford Memorial Church (also referred to informally as MemChu) is located on the Main Quad at the center of the Stanford University campus in Stanford, California, United States. It was built during the American Renaissance by Jane Stanford as a memorial to her husband Leland. Designed by architect Charles A. Coolidge, a protégé of Henry Hobson Richardson, the church has been called ”the University’s architectural crown jewel”. |
| (c) | **Question** | Who invented LSTM? |
| | **Answer** | Sepp Hochreiter & Jürgen Schmidhuber |
| | **Wiki. article** | Deep Learning |
| | **Passage** | Today, however, many aspects of speech recognition have been taken over by a deep learning method called Long short-term memory (LSTM), a recurrent neural network published by Sepp Hochreiter & Jürgen Schmidhuber in 1997. LSTM RNNs avoid the vanishing gradient problem and can learn “Very Deep Learning” tasks that require memories of events that happened thousands of discrete time steps ago, which is important for speech. In 2003, LSTM started |

| | | |
|---|---|---|
| | | to become competitive with traditional speech recognizers on certain tasks. Later it was combined with CTC in stacks of LSTM RNNs. In 2015, Google's speech recognition reportedly experienced a dramatic performance jump of 49% through CTC-trained LSTM, which is now available through Google Voice to all smartphone users, and has become a show case of deep learning. |
| (d) | **Question** | What is the answer to life, the universe, and everything? |
| | **Answer** | 42 |
| | **Wiki. article** | Phrases from The Hitchhiker's Guide to the Galaxy |
| | **Passage** | The number 42 and the phrase, "Life, the universe, and everything" have attained cult status on the Internet. "Life, the universe, and everything" is a common name for the off-topic section of an Internet forum and the phrase is invoked in similar ways to mean "anything at all". Many chatbots, when asked about the meaning of life, will answer "42". Several online calculators are also programmed with the Question. Google Calculator will give the result to "the answer to life the universe and everything" as 42, as will Wolfram's Computational Knowledge Engine. Similarly, DuckDuckGo also gives the result of "the answer to the ultimate question of life, the universe and everything" as 42. In the online community Second Life, there is a section on a sim called "42nd Life." It is devoted to this concept in the book series, and several attempts at recreating Milliways, the Restaurant at the End of the Universe, were made. |

Table 5.4: Sample predictions of our DrQA system.

Lastly, our DrQA system is open-sourced at https://github.com/facebookresearch/DrQA (the Multitask (DS) system was deployed). Table 5.4 lists some sample predictions that we tried by ourselves (not in any of these datasets). As is seen, our system is able to return a precise answer to all these factoid questions and answering some of these questions is not trivial:

(a) It is not trivial to identify that *a computer science discipline within the fields of information retrieval and natural language processing* is the complete noun phrase and the correct answer although the question is pretty simple.

(b) Our system finds the answer in another Wikipedia article *Stanford Memorial Church*, and gives the exactly correct answer *California* as the *state* (instead of *Stanford* or *United States*).

(c) To get the correct answer, the system needs to understand the syntactic structure of the question and the context *Who invented LSTM?* and *a deep learning method called Long short-term memory (LSTM), a recurrent neural network published by Sepp Hochreiter & Jürgen Schmidhuber in 1997.*

Conceptually, our system is simple and elegant, and doesn't rely on any additional linguistic analysis or external or hand-coded resources (e.g., dictionaries). We think this approach holds great promise for a new generation of open-domain question answering systems. In the next section, we discuss current limitations and possible directions for further improvement.

## 5.4 Future Work

Our DRQA demonstrates that combining information retrieval and neural reading comprehension is an effective approach for open-domain question answering. We hope that our work takes the first step in this research direction. However, our system is still at an early stage and many implementation details can be further improved.

We think the following research directions will (greatly) improve our DRQA system and should be pursued as future work. Indeed, some of the ideas have already been implemented in the following year after we published our DRQA system and we will also describe them in detail in this section.

**Aggregating evidence from multiple paragraphs.** Our system adopted the most simple and straightforward approach: we took the argmax over the unnormalized scores of all the retrieved passages. This is not ideal because 1) It implies that each passage must contain the correct answer (as SQUAD examples) so our system will output one and only one answer for each passage. This is indeed not the case for most retrieved passages. 2) Our current

training paradigm doesn't guarantee that the scores in different passages are comparable which causes a gap between the training and the evaluation process.

Training on full Wikipedia articles is a solution to alleviate this problem (see the DRQA* results in Table 5.3), however, these models are running slowly and difficult to parallelize. Clark and Gardner (2018) proposed to perform multi-paragraph training with modified training objectives, where the span start and end scores are normalized across all paragraphs sampled from the same context. They demonstrated that it works much better than training on individual passages independently. Similarly, Wang et al. (2018a) and Wang et al. (2018b) proposed to train an explicit passage re-ranking component on the retrieved articles: Wang et al. (2018a) implemented this in a reinforcement learning framework so the re-ranker component and answer extraction components are jointly trained; Wang et al. (2018b) proposed a strength-based re-ranker and a coverage-based re-ranker which aggregate evidence from multiple paragraphs more directly.

**Using more and better training data.** The second aspect which makes a big impact is the training data. Our DRQA system only collected 44k distantly-supervised training examples from TREC, WEBQUESTIONS and WIKIMOVIES, and we demonstrated their effectiveness in Section 5.3.4. The system should be further improved if we can leverage more supervised training data — from either TRIVIAQA (Joshi et al., 2017) or generating more data from other QA resources. Moreover, these distantly supervised examples inevitably suffer from the noise problem (i.e., the paragraph doesn't imply the answer to the question even if the answer is contained) and Lin et al. (2018) proposed a solution to de-noise these distantly supervised examples and demonstrated gains in an evaluation.

We also believe that adding negative examples should improve the performance of our system substantially. We can either create some negative examples using our full pipeline: we can leverage the DOCUMENT RETRIEVAL module to help us find relevant paragraphs while they don't contain the correct answer. We can also incorporate existing resources such as SQUAD 2.0 (Rajpurkar et al., 2018) into our training process, which contains curated, high-quality negative examples.

**Making the DOCUMENT RETRIEVER trainable.** A third promising direction that has not been fully studied yet is to employ a machine learning approach for the DOCUMENT RETRIEVER module. Our system adopted a straightforward, non-machine learning model and further improvement on the retrieval performance (Table 5.2) should lead to an improvement on the full system. A training corpus for the DOCUMENT RETRIEVER component can be collected either from other resources or from the QA data (e.g., using whether an article contains the answer to the question as a label). Joint training of the DOCUMENT RETRIEVAL and the DOCUMENT READER component will be a very desirable and promising direction for future work.

Related to this, Clark and Gardner (2018) also built an open-domain question answering system[9] on top of a search engine (Bing web search) and demonstrated superior performance compared to ours. We think the results are not directly comparable and the two approaches (using a commercial search engine or building an independent IR component) both have pros and cons. Building our own IR component gets rid of an existing API call and can run faster and easily adapt to new domains.

**Better DOCUMENT READER module.** For our DRQA system, we used the neural reading comprehension model which achieved F1 of 79.0% on the test set of SQUAD 1.1. With the recent development of neural reading comprehension models (Section 3.4), we are confident that if we replace our current DOCUMENT READER model with the state-of-the-art models (Devlin et al., 2018), the performance of our full system will be improved as well.

**More analysis is needed.** Another important missing work is to conduct an in-depth analysis of our current systems: to understand which questions they can answer, and which they can't. We think it is important to compare our modern systems to the earlier TREC QA results under the same conditions. It will help us understand where we make genuine progress and what techniques we can still use from the pre-deep learning era, to build better question answering systems in the future.

Concurrent to our work, there are several works in a similar spirit to ours, including

[9] The demo is at https://documentqa.allenai.org.

SEARCHQA (Dunn et al., 2017) and QUASAR-T (Dhingra et al., 2017b), which both collected relevant documents for trivia or JEOPARDY! questions — the former one retrieved documents from CLUEWEB using the LUCENE index and the latter used GOOGLE search. TRIVIAQA (Joshi et al., 2017) also has an open-domain setting where all the retrieved documents from Bing web search are kept. However, these datasets still focus on the task of question answering from the retrieved documents, while we are more interested in building an end-to-end QA system.

# Chapter 6

# Conversational Question Answering

In the last chapter, we discussed how we built a general-knowledge question-answering system from neural reading comprehension. However, most current QA systems are limited to answering isolated questions, i.e., every time we ask a question, the systems return an answer without the ability to consider any context. In this chapter, we set out to tackle another challenging problem *Conversational Question Answering*, where a machine has to understand a text passage and answer a series of questions that appear in a conversation.

Humans gather information by engaging in conversations involving a series of interconnected questions and answers. For machines to assist in information gathering, it is therefore essential to enable them to answer conversational questions. Figure 6.1 shows a conversation between two humans who are reading a passage, one acting as a questioner and the other as an answerer. In this conversation, every question after the first is dependent on the conversation history. For instance, $Q_5$ *Who?* is only a single word and is impossible to answer without knowing what has already been said. Posing short questions is an effective human conversation strategy, but such questions are really difficult for machines to parse. Therefore, conversational question answering combines the challenges from both dialogue and reading comprehension.

We believe that building systems which are able to answer such conversational questions will play a crucial role in our future conversational AI systems. To approach this problem, we need to build effective *datasets* and conversational QA *models* and we will describe both of them in this chapter.

Jessica went to sit in her rocking chair. Today was her birthday and she was turning 80. Her granddaughter Annie was coming over in the afternoon and Jessica was very excited to see her. Her daughter Melanie and Melanie's husband Josh were coming as well. Jessica had . . .

$Q_1$: Who had a birthday?
$A_1$: Jessica
$R_1$: Jessica went to sit in her rocking chair. Today was her birthday and she was turning 80.

$Q_2$: How old would she be?
$A_2$: 80
$R_2$: she was turning 80

$Q_3$: Did she plan to have any visitors?
$A_3$: Yes
$R_3$: Her granddaughter Annie was coming over

$Q_4$: How many?
$A_4$: Three
$R_4$: Her granddaughter Annie was coming over in the afternoon and Jessica was very excited to see her. Her daughter Melanie and Melanie's husband Josh were coming as well.

$Q_5$: Who?
$A_5$: Annie, Melanie and Josh
$R_5$: Her granddaughter Annie was coming over in the afternoon and Jessica was very excited to see her. Her daughter Melanie and Melanie's husband Josh were coming as well.

Figure 6.1: A conversation from our CoQA dataset. Each turn contains a question ($Q_i$), an answer ($A_i$) and a rationale ($R_i$) that supports the answer.

This chapter is organized as follows. We first discuss related work in Section 6.1 and then we introduce CoQA (Reddy et al., 2019) in Section 6.2, a **Co**nversational **Q**uestion **A**nswering challenge for measuring the ability of machines to participate in a question-answering style conversation.[1] Our dataset contains 127k questions with answers, obtained from 8k conversations about text passages from seven diverse domains. We define our task

[1] We launch CoQA as a challenge to the community at https://stanfordnlp.github.io/coqa/.

and describe the dataset collection process. We also analyze the dataset in depth and show that conversational questions have challenging phenomena not present in existing reading comprehension datasets, e.g., coreference and pragmatic reasoning. Next we describe several strong conversational and reading comprehension models we built for CoQA in Section 6.3 and present experimental results in Section 6.4. Finally, we discuss future work of conversational question answering (Section 6.5).

## 6.1 Related Work

Conversational question answering is directly related to **dialogue**. Building conversational agents, or dialogue systems to converse with humans in natural language is one of the major goals of natural language understanding. The two most common classes of dialogue systems are: *task-oriented*, and *chit-chat* (or *chatbot*) dialogue agents. Task-oriented dialogue systems are designed for a particular task and set up to have short conversations (e.g., booking a flight or making a restaurant reservation). They are evaluated based on task-completion rate or time to task completion. In contrast, chit-chat dialogue systems are designed for extended, casual conversations, without a specific goal. Usually, the longer the user engagement and interaction, the better these systems are.

Answering questions is also a core task of dialogue systems, because one of the most common needs for humans to interact with dialogue agents is to seek information and ask questions of various topics. QA-based dialogue techniques have been developed extensively in automated personal assistant systems such as Amazon's ALEXA, Apple's SIRI or GOOGLE ASSISTANT, either based on structured knowledge bases, or unstructured text collections. Modern dialogue systems are mostly built on top of deep neural networks. For a comprehensive survey of neural approaches to different types of dialogue systems, we refer readers to (Gao et al., 2018).

Our work is closely related to the *Visual Dialog* task of (Das et al., 2017) and the *Complex Sequential Question Answering* task of (Saha et al., 2018), which perform conversational question answering on images and a knowledge graph (e.g. WIKIDATA) respectively, with the latter focusing on questions obtained by paraphrasing templates. Figure 6.2 demonstrates an example from each task. We focus on conversations over a passage of text,

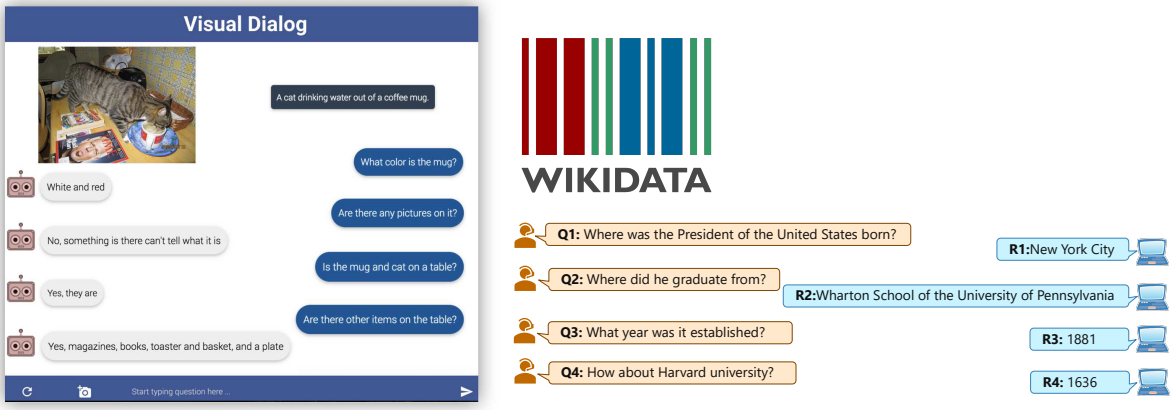



Figure 6.2: Other conversational question answering tasks on images (left) and KBs (right). Images courtesy: (Das et al., 2017) and (Guo et al., 2018) with modifications.

which requires the ability of reading comprehension.

Another related line of research is *sequential question answering* (Iyyer et al., 2017; Talmor and Berant, 2018), in which a complex question is decomposed into a sequence of simpler questions. For example, a question *What super hero from Earth appeared most recently?* can be decomposed into the following three questions: 1) *Who are all of the super heroes?*, 2) *Which of them come from Earth?*, and 3) *Of those, who appeared most recently?*. Therefore, their focus is how to answer a complex question via sequential question answering, while we are more interested in a natural conversation of a variety of topics while the questions can be dependent on the dialogue history.

## 6.2 CoQA: A Conversational QA Challenge

In this section, we introduce CoQA, a novel dataset for building **Co**nversational **Q**uestion **A**nswering systems. We develop CoQA with three main goals in mind. The first concerns the nature of questions in a human conversation. As an example seen in Figure 6.1, in this conversation, every question after the first is dependent on the conversation history. At present, there are no large scale reading comprehension datasets which contain questions

that depend on a conversation history and this is what CoQA is mainly developed for.[2]

The second goal of CoQA is to ensure the naturalness of answers in a conversation. As we discussed in the earlier chapters, most existing reading comprehension datasets either restrict answers to a contiguous span in a given passage, or allow free-form answers with a low human agreement (e.g., NarrativeQA). Our desiderata are 1) the answers should not be only span-based so that anything can be asked and the conversation can flow naturally. For example, there is no extractive answer for $Q_4$ *How many?* in Figure 6.1. 2) It still supports reliable automatic evaluation with a a strong human performance. Therefore, we propose that the answers can be free-form text (abstractive answers), while the extractive spans act as rationales for the actual answers. Therefore, the answer for $Q_4$ is simply *Three* while its rationale is spanned across multiple sentences.

The third goal of CoQA is to enable building QA systems that perform robustly across domains. The current reading comprehension datasets mainly focus on a single domain which makes it hard to test the generalization ability of existing models. Hence we collect our dataset from seven different domains — children's stories, literature, middle and high school English exams, news, Wikipedia, science articles and Reddit. The last two are used for out-of-domain evaluation.

### 6.2.1 Task Definition

We first define the task formally. Given a passage $P$, a conversation consists of $n$ turns, and each turn consists of $(Q_i, A_i, R_i), i = 1, \dots n$, where $Q_i$ and $A_i$ denote the question and the answer in the $i$-th turn, and $R_i$ is the rationale which supports the answer $A_i$ and must be a single span of the passage. The task is defined as to answer the next question $Q_i$ given the conversation so far: $Q_1, A_1, \dots, Q_{i-1}, A_{i-1}$. It is worth noting that we collect $R_i$ with the hope that they can help understand how answers are derived and improve training our models, while *they are not provided during evaluation*.

For the example in Figure 6.3, the conversation begins with question $Q_1$. We answer $Q_1$ with $A_1$ based on the evidence $R_1$ from the passage. In this example, the answerer

[2]Concurrent with our work, Choi et al. (2018) also created a conversational dataset with a similar goal, but it differs in many key design decisions. We will discuss it in Section 6.5.

The Virginia governor's race, billed as the marquee battle of an otherwise anticlimactic 2013 election cycle, is shaping up to be a foregone conclusion. Democrat Terry McAuliffe, the longtime political fixer and moneyman, hasn't trailed in a poll since May. Barring a political miracle, Republican Ken Cuccinelli will be delivering a concession speech on Tuesday evening in Richmond. In recent ...

$Q_1$: What are the candidates **running** for?
$A_1$: Governor
$R_1$: The Virginia governor's race

$Q_2$: **Where**?
$A_2$: Virginia
$R_2$: The Virginia governor's race

$Q_3$: Who is the democratic candidate?
**A**$_3$: **Terry McAuliffe**
$R_3$: Democrat Terry McAuliffe

$Q_4$: Who is **his** opponent?
**A**$_4$: **Ken Cuccinelli**
$R_4$ Republican Ken Cuccinelli

$Q_5$: What party does **he** belong to?
$A_5$: Republican
$R_5$: Republican Ken Cuccinelli

$Q_6$: Which of **them** is winning?
$A_6$: Terry McAuliffe
$R_6$: Democrat Terry McAuliffe, the longtime political fixer and moneyman, hasn't trailed in a poll since May

Figure 6.3: A conversation showing coreference chains in colors. The entity of focus changes in $Q_4$, $Q_5$, $Q_6$.

wrote only the *Governor* as the answer but selected a longer rationale *The Virginia governor's race*. When we come to $Q_2$ *Where?*, we must refer back to the conversation history since otherwise its answer could be *Virginia* or *Richmond* or something else. In our task, conversation history is indispensable for answering many questions. We use conversation history $Q_1$ and $A_1$ to answer $Q_2$ with $A_2$ based on the evidence $R_2$. For an unanswerable

question, we give *unknown* as the final answer and do not highlight any rationale.

In this example, we observe that the entity of focus changes as the conversation progresses. The questioner uses *his* to refer to *Terry* in $Q_4$ and *he* to *Ken* in $Q_5$. If these are not resolved correctly, we end up with incorrect answers. The conversational nature of questions requires us to reason from multiple sentences (the current question and the previous questions or answers, and sentences from the passage). It is common that a single question may require a rationale spanned across multiple sentences (e.g., $Q_1$ $Q_4$ and $Q_5$ in Figure 6.1). We describe additional question and answer types in 6.2.3.

### 6.2.2 Dataset Collection

We detail our dataset collection process as follows. For each conversation, we employ two annotators, a questioner and an answerer. This setup has several advantages over using a single annotator to act both as a questioner and an answerer: 1) when two annotators chat about a passage, their dialogue flow is natural compared to chatting with oneself; 2) when one annotator responds with a vague question or an incorrect answer, the other can raise a flag which we use to identify bad workers; and 3) the two annotators can discuss guidelines (through a separate chat window) when they have disagreements. These measures help to prevent spam and to obtain high agreement data.[3]

We use the Amazon Mechanical Turk (AMT) to pair workers on a on a passage for which we use the ParlAI Mturk API (Miller et al., 2017). On average, each passage costs 3.6 USD for conversation collection and another 4.5 USD for collecting three additional answers for development and test data.

**Collection interface.** We have different interfaces for a questioner and an answerer (Figure 6.4 and Figure 6.5). A questioner's role is to ask questions, and an answerer's role is to answer questions in addition to highlighting rationales. We want questioners to avoid using exact words in the passage in order to increase lexical diversity. When they type a word that is already present in the passage, we alert them to paraphrase the question if possible. For the answers, we want answerers to stick to the vocabulary in the passage in

[3]Due to AMT terms of service, we allowed a single worker to act both as a questioner and an answerer after a minute of waiting. This constitutes around 12% of the data.

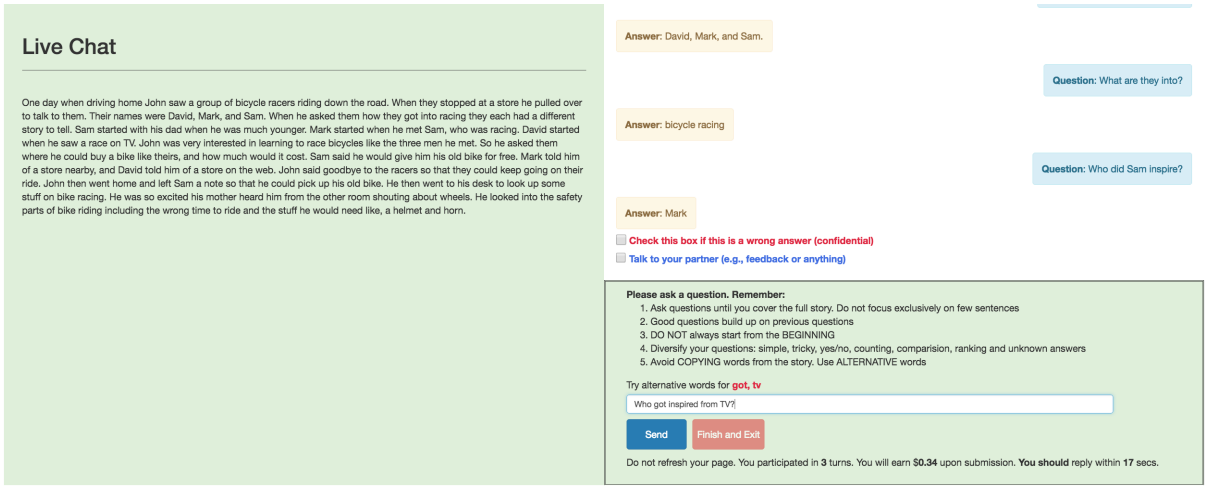



Figure 6.4: The questioner interface of our CoQA dataset.

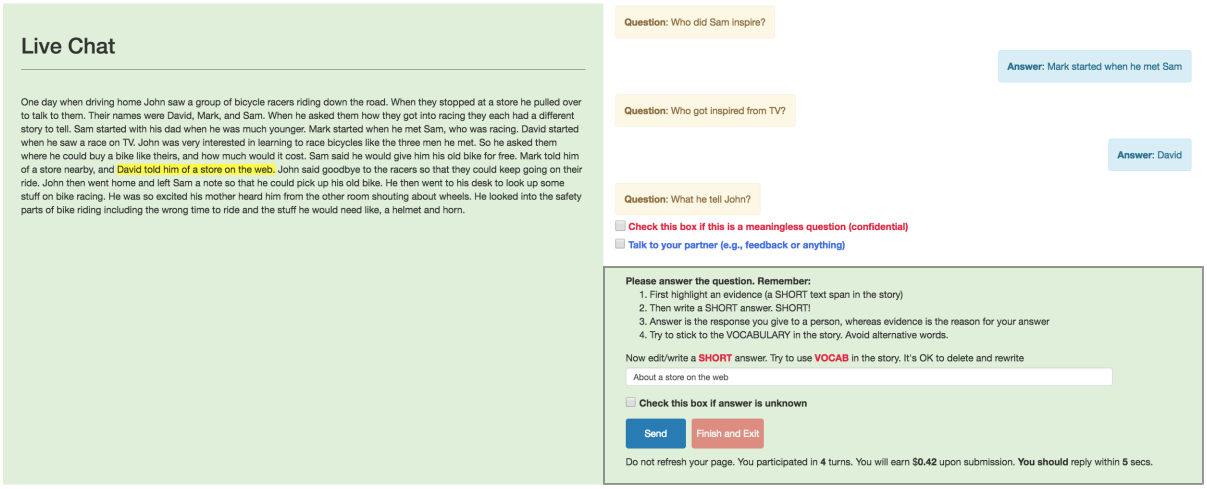



Figure 6.5: The answerer interface of our CoQA dataset.

order to limit the number of possible answers. We encourage this by automatically copying the highlighted text into the answer box and allowing them to edit copied text in order to generate a natural answer. We found 78% of the answers have at least one edit such as changing a word's case or adding a punctuation.

| Domain | # Passages | # Q/A pairs | Passage length | # Turns per passage |
|---|---|---|---|---|
| Children's Stories | 750 | 10.5k | 211 | 14.0 |
| Literature | 1,815 | 25.5k | 284 | 15.6 |
| Mid/High School Exams | 1,911 | 28.6k | 306 | 15.0 |
| News | 1,902 | 28.7k | 268 | 15.1 |
| Wikipedia | 1,821 | 28.0k | 245 | 15.4 |
| Out of domain | | | | |
| Science | 100 | 1.5k | 251 | 15.3 |
| Reddit | 100 | 1.7k | 361 | 16.6 |
| Total | 8,399 | 127k | 271 | 15.2 |

Table 6.1: Distribution of domains in CoQA.

**Passage selection.** We select passages from seven diverse domains: children's stories from MCTest (Richardson et al., 2013), literature from Project Gutenberg[4], middle and high school English exams from RACE (Lai et al., 2017), news articles from CNN (Hermann et al., 2015), articles from Wikipedia, science articles from AI2 Science Questions (Welbl et al., 2017) and Reddit articles from the Writing Prompts dataset (Fan et al., 2018).

Not all passages in these domains are equally good for generating interesting conversations. A passage with just one entity often result in questions that entirely focus on that entity. We select passages with multiple entities, events and pronominal references using Stanford CoreNLP (Manning et al., 2014). We truncate long articles to the first few paragraphs that result in around 200 words.

Table 6.1 shows the distribution of domains. We reserve the Science and Reddit domains for out-of-domain evaluation. For each in-domain dataset, we split the data such that there are 100 passages in the development set, 100 passages in the test set, and the rest in the training set. In contrast, for each out-of-domain dataset, we just have 100 passages in the test set without any passages in the training or the development sets.

**Collecting multiple answers.** Some questions in CoQA may have multiple valid answers. For example, another answer for $Q_4$ in Figure 6.3 is *A Republican candidate*. In

[4]Project Gutenberg https://www.gutenberg.org

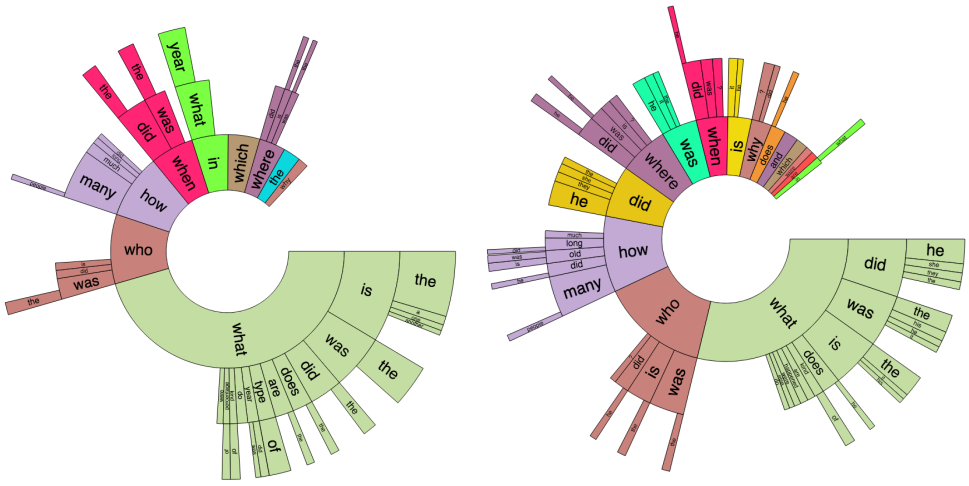
order to account for answer variations, we collect three additional answers for all questions in the development and test data. Since our data is conversational, questions influence answers which in turn influence the follow-up questions. In the previous example, if the original answer was *A Republican Candidate*, then the following question *Which party does he belong to?* would not have occurred in the first place. When we show questions from an existing conversation to new answerers, it is likely they will deviate from the original answers which makes the conversation incoherent. It is thus important to bring them to a common ground with the original answer.

We achieve this by turning the answer collection task into a game of predicting original answers. First, we show a question to a new answerer, and when she answers it, we show the original answer and ask her to verify if her answer matches the original. For the next question, we ask her to guess the original answer and verify again. We repeat this process until the conversation is complete. In our pilot experiment, the human F1 score increased by 5.4% when we use this verification setup.

### 6.2.3 Dataset Analysis

What makes the CoQA dataset conversational compared to existing reading comprehension datasets like SQuAD? How does the conversation flow from one turn to the other? What linguistic phenomena do the questions in CoQA exhibit? We answer these questions below.

**Comparison with SQuAD 2.0.** In the following, we perform an in-depth comparison of CoQA and SQuAD 2.0 (Rajpurkar et al., 2018). Figure 6.6 shows the distribution of frequent trigram prefixes. While coreferences are non-existent in SQuAD 2.0, almost every sector of CoQA contains coreferences (*he, him, she, it, they*) indicating CoQA is highly conversational. Because of the free-form nature of answers, we expect a richer variety of questions in CoQA than SQuAD 2.0. While nearly half of SQuAD 2.0 questions are dominated by *what* questions, the distribution of CoQA is spread across multiple question types. Several sectors indicated by prefixes *did, was, is, does, and* are frequent in CoQA but are completely absent in SQuAD 2.0.




(a) SQuAD (b) CoQA

Figure 6.6: Distribution of trigram prefixes of questions in SQuAD 2.0 and CoQA.

Since a conversation is spread over multiple turns, we expect conversational questions and answers to be shorter than in a standalone interaction. In fact, questions in CoQA can be made up of just one or two words (*who?*, *when?*, *why?*). As seen in Table 6.2, on average, a question in CoQA is only 5.5 words long while it is 10.1 for SQuAD. The answers are also usually shorter in CoQA than SQuAD 2.0.

Table 6.3 provides insights into the type of answers in SQuAD 2.0 and CoQA. While the original version of SQuAD 2.0 (Rajpurkar et al., 2016) does not have any unanswerable questions, SQuAD 2.0 (Rajpurkar et al., 2018) focuses solely on obtaining them resulting in higher frequency than in CoQA. SQuAD 2.0 has 100% extractive answers by design, whereas in CoQA, 66.8% answers can be classified as extractive after ignoring punctuation and case mismatches.[5] This is higher than we anticipated. Our conjecture is that human factors such as wage may have influenced workers to ask questions that elicit faster responses by selecting text. It is worth noting that CoQA has 11.1% and 8.7% questions with *yes* or *no* as answers whereas SQuAD 2.0 has 0%. Both datasets have a high number of named entities and noun phrases as answers.

[5]If punctuation and case are not ignored, only 37% of the answers are extractive.

| | SQuAD 2.0 | CoQA |
|---|---|---|
| Passage Length | 117 | 271 |
| Question Length | 10.1 | 5.5 |
| Answer Length | 3.2 | 2.7 |

Table 6.2: Average number of words in passage, question and answer in SQuAD 2.0 and CoQA.

| | SQuAD 2.0 | CoQA |
|---|---|---|
| Answerable | 66.7% | 98.7% |
| Unanswerable | 33.3% | 1.3% |
| Extractive | 100.0% | 66.8% |
| Abstractive | 0.0% | 33.2% |
| Named Entity | 35.9% | 28.7% |
| Noun Phrase | 25.0% | 19.6% |
| Yes | 0.0% | 11.1% |
| No | 0.1% | 8.7% |
| Number | 16.5% | 9.8% |
| Date/Time | 7.1% | 3.9% |
| Other | 15.5% | 18.1% |

Table 6.3: Distribution of answer types in SQuAD 2.0 and CoQA.

**Conversation flow.** A coherent conversation must have smooth transitions between turns. We expect the narrative structure of the passage to influence our conversation flow. We split the passage into 10 uniform chunks, and identify chunks of interest of a given turn and its transition based on rationale spans.

Figure 6.7 portrays the conversation flow of the top 10 turns. The starting turns tend to focus on the first few chunks and as the conversation advances, the focus shifts to the later chunks. Moreover, the turn transitions are smooth, with the focus often remaining in the same chunk or moving to a neighbouring chunk. Most frequent transitions happen to the first and the last chunks, and likewise these chunks have diverse outward transitions.

**Linguistic phenomena.** We further analyze the questions for their relationship with the passages and the conversation history. We sample 150 questions in the development set

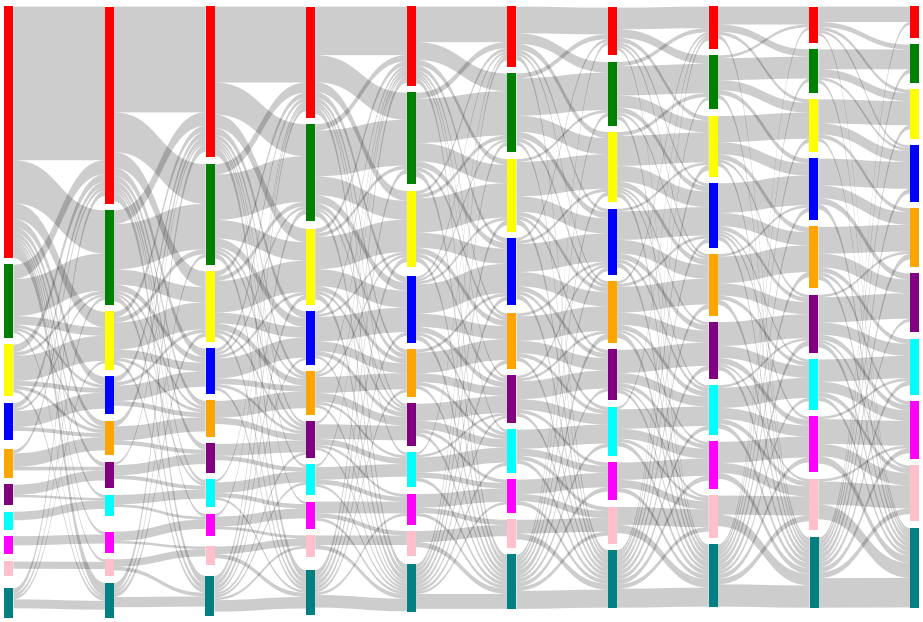


Figure 6.7: Chunks of interests as a conversation progresses. The x-axis indicates the turn number and the y-axis indicates the passage chunk containing the rationale. The height of a chunk indicates the concentration of conversation in that chunk. The width of the bands is proportional to the frequency of transition between chunks from one turn to the other.

and annotate various phenomena as shown in Table 6.4.

If a question contains at least one content word that appears in the passage, we classify it as *lexical match*. These comprise around 29.8% of the questions. If it has no lexical match but is a paraphrase of the rationale, we classify it as *paraphrasing*. These questions contain phenomena such as synonymy, antonymy, hypernymy, hyponymy and negation. These constitute a large portion of questions, around 43.0%. The rest, 27.2%, have no lexical cues, and we classify them under *pragmatics*. These include phenomena like common sense and presupposition. For example, the question *Was he loud and boisterous?* is not a direct paraphrase of the rationale *he dropped his feet with the lithe softness of a cat* but the rationale combined with world knowledge can answer this question.

| Phenomenon | Example | Percentage |
|---|---|---|
| Relationship between a question and its passage | | |
| Lexical match | Q: Who had to rescue her?<br>A: the coast guard<br>R: Outen was rescued by the coast guard | 29.8% |
| Paraphrasing | Q: Did the wild dog approach?<br>A: Yes<br>R: he drew cautiously closer | 43.0% |
| Pragmatics | Q: Is Joey a male or female?<br>A: Male<br>R: it looked like a stick man so she kept **him**.<br>She named her new noodle friend Joey | 27.2% |
| Relationship between a question and its conversation history | | |
| No coreference | Q: What is IFL? | 30.5% |
| Explicit coreference | Q: Who had Bashti forgotten?<br>A: the puppy<br>Q: What was **his** name? | 49.7% |
| Implicit coreference | Q: When will Sirisena be sworn in?<br>A: 6 p.m local time<br>Q: **Where**? | 19.8% |

Table 6.4: Linguistic phenomena in CoQA questions.

For the relationship between a question and its conversation history, we classify questions into whether they are dependent or independent on the conversation history. If dependent, whether the questions contain an explicit marker or not.

As a result, around 30.5% questions do not rely on coreference with the conversational history and are answerable on their own. Almost half of the questions (49.7%) contain explicit coreference markers such as *he, she, it*. These either refer to an entity or an event introduced in the conversation. The remaining 19.8% do not have explicit coreference markers but refer to an entity or event implicitly.

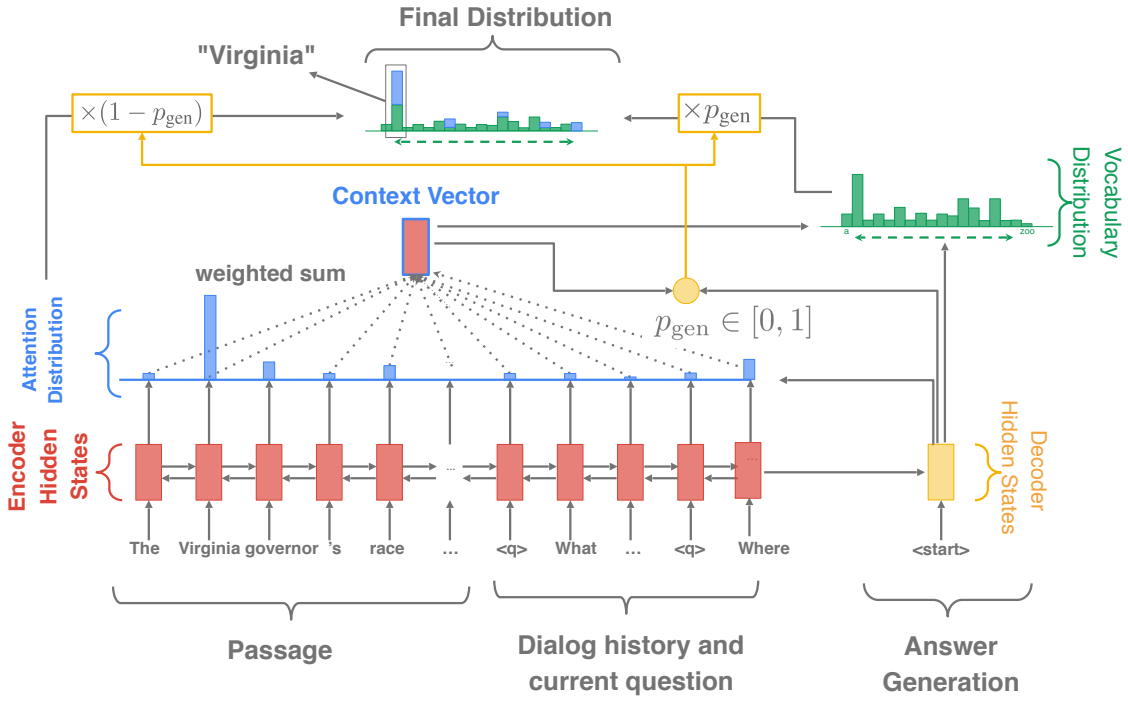



Figure 6.8: The pointer-generator network used for conversational question answering. The figure is adapted from See et al. (2017).

## 6.3 Models

Given a passage $p$, the conversation history $\{q_1, a_1, \ldots q_{i-1}, a_{i-1}\}$ and a question $q_i$, the task is to predict the answer $a_i$. Our task can be modeled as either a conversational response generation problem or a reading comprehension problem. We evaluate strong baselines from each class of models and a combination of the two on CoQA.

### 6.3.1 Conversational Models

The basic goal of conversational models is to predict the next utterance based on its conversation history. Sequence-to-sequence (seq2seq) models (Sutskever et al., 2014) have shown promising results for generating conversational responses (Vinyals and Le, 2015; Li

et al., 2016; Zhang et al., 2018). Motivated by their success, we use a standard sequence-to-sequence model with an attention mechanism for generating answers. We append the passage, the conversation history (the question/answer pairs in the last $n$ turns) and the current question as, $p$ $<q>$ $q_{i-n}$ $<a>$ $a_{i-n}$ $\ldots$ $<q>$ $q_{i-1}$ $<a>$ $a_{i-1}$ $<q>$ $q_i$, and feed it into a bidirectional LSTM encoder, where $<q>$ and $<a>$ are special tokens used as delimiters. We then generate the answer using a LSTM decoder which attends to the encoder states.

Moreover, as the answer words are likely to appear in the original passage, we adopt a copy mechanism in the decoder proposed for summarization tasks (Gu et al., 2016; See et al., 2017), which allows to (optionally) copy a word from the passage and the conversation history. We call this model the Pointer-Generator network (See et al., 2017), PGNET. Figure 6.8 illustrates a full model of PGNET. Formally, we denote the encoder hidden vectors by $\{\tilde{\mathbf{h}}_i\}$, the decoder state at timestep $t$ by $\mathbf{h}_t$ and the input vector by $\mathbf{x}_t$, an attention function is computed based on $\{\tilde{\mathbf{h}}_i\}$ and $\mathbf{h}_t$ as $\alpha_i$ (Equation 3.13) and the context vector is computed as $\mathbf{c} = \sum_i \alpha_i \tilde{\mathbf{h}}_i$ (Equation 3.14).

For a copy mechanism, it first computes the *generation probability* $p_{\text{gen}} \in [0, 1]$ which controls the probability that it generates a word from the full vocabulary $\mathcal{V}$ (rather than copying a word) as:

$$p_{\text{gen}} = \sigma\left(\mathbf{w}^{(c)\intercal}\mathbf{c} + \mathbf{w}^{(x)\intercal}\mathbf{x}_t + \mathbf{w}^{(h)\intercal}\mathbf{h}_t + b\right). \tag{6.1}$$

The final probability distribution of generating word $w$ is computed as:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} \alpha_i, \tag{6.2}$$

where $P_{\text{vocab}}(w)$ is the original probability distribution (computed based on $\mathbf{c}$ and $\mathbf{h}_t$) and $\{w_i\}$ refers to all the words in the passage and the dialogue history. For more details, we refer readers to (See et al., 2017).

### 6.3.2 Reading Comprehension Models

The second class of models we evaluate is the neural reading comprehension models. In particular, the models for the span prediction problems can't be applied directly, as a large

portion of the CoQA questions don't have a single span in the passage as their answer, e.g., $Q_3$, $Q_4$ and $Q_5$ in Figure 6.1. Therefore, we modified the STANFORD ATTENTIVE READER model we described in Section 3.2 for this problem. Since the model requires text spans as answers during training, we select the span which has the highest lexical overlap (F1 score) with the original answer as the gold answer. If the answer appears multiple times in the story we use the rationale to find the correct one. If any answer word does not appear in the passage, we fall back to an additional *unknown* token as the answer (about 17%). We prepend each question with its past questions and answers to account for conversation history, similar to the conversational models.

### 6.3.3 A Hybrid Model

The last model we build is a *hybrid* model, which combines the advantages of the aforementioned two models. The reading comprehension models can predict a text span as an answer, while they can't produce answers that do not overlap with the passage. Therefore, we combine STANFORD ATTENTIVE READER with PGNET to address this problem since PGNET can generate free-form answers effectively. In this hybrid model, we use the reading comprehension model to first point to the answer evidence in text, and PGNET naturalizes the evidence into the final answer. For example, for $Q_5$ in Figure 6.1, we expect that the reading comprehension model first predicts the rationale $R_5$ *Her granddaughter Annie was coming over in the afternoon and Jessica was very excited to see her. Her daughter Melanie and Melanie's husband Josh were coming as well.*, and then PGNET generates $A_5$ *Annie, Melanie and Josh* from $R_5$.

We make a few changes to both models based on empirical performance. For the STANFORD ATTENTIVE READER model, we only use rationales as answers for the questions with an non-extractive answer. For PGNET, we only provide current question and span predictions from the the STANFORD ATTENTIVE READER model as input to the encoder. During training, we feed the oracle spans into PGNET.

## 6.4 Experiments

### 6.4.1 Setup

For the SEQ2SEQ and PGNET experiments, we use the OPENNMT toolkit (Klein et al., 2017). For the reading comprehension experiments, we use the same implementation that we used for SQUAD (Chen et al., 2017). We tune the hyperparameters on the development data: the number of turns to use from the conversation history, the number of layers, number of each hidden units per layer and dropout rate. We initialize the word projection matrix with GLOVE (Pennington et al., 2014) for conversational models and FASTTEXT (Bojanowski et al., 2017) for reading comprehension models, based on empirical performance. We update the projection matrix during training in order to learn embeddings for delimiters such as $<q>$.

For all the experiments of SEQ2SEQ and PGNET, we use the default settings of OPENNMT: 2-layers of LSTMs with $500$ hidden units for both the encoder and the decoder. The models are optimized using SGD, with an initial learning rate of $1.0$ and a decay rate of $0.5$. A dropout rate of $0.3$ is applied to all layers.

For all the reading comprehension experiments, the best configuration we find is 3 layers of LSTMs with $300$ hidden units for each layer. A dropout rate of $0.4$ is applied to all LSTM layers and a dropout rate of $0.5$ is applied to word embeddings.

### 6.4.2 Experimental Results

Table 6.5 presents the results of the models on the development and the test data. Considering the results on the test set, the SEQ2SEQ model performs the worst, generating frequently occurring answers irrespective of whether these answers appear in the passage or not, a well known behavior of conversational models (Li et al., 2016). PGNET alleviates the frequent response problem by focusing on the vocabulary in the passage and it outperforms SEQ2SEQ by 17.8 points. However, it still lags behind STANFORD ATTENTIVE READER by 8.5 points. A reason could be that PGNET has to memorize the whole passage before answering a question, a huge overhead which STANFORD ATTENTIVE READER avoids. But STANFORD ATTENTIVE READER fails miserably in answering questions with

| | In-domain | | | | | Out-of-domain | | Overall |
|---|---|---|---|---|---|---|---|---|
| | Children | Literature | Exam | News | Wikipedia | Reddit | Science | |
| **Development data** | | | | | | | | |
| SEQ2SEQ | 30.6 | 26.7 | 28.3 | 26.3 | 26.1 | N/A | N/A | 27.5 |
| PGNET | 49.7 | 42.4 | 44.8 | 45.5 | 45.0 | N/A | N/A | 45.4 |
| SAR | 52.4 | 52.6 | 51.4 | 56.8 | 60.3 | N/A | N/A | 54.7 |
| HYBRID | **64.5** | **62.0** | **63.8** | **68.0** | **72.6** | N/A | N/A | **66.2** |
| HUMAN | 90.7 | 88.3 | 89.1 | 89.9 | 90.9 | N/A | N/A | 89.8 |
| **Test data** | | | | | | | | |
| SEQ2SEQ | 32.8 | 25.6 | 28.0 | 27.0 | 25.3 | 25.6 | 20.1 | 26.3 |
| PGNET | 49.0 | 43.3 | 47.5 | 47.5 | 45.1 | 38.6 | 38.1 | 44.1 |
| SAR | 46.7 | 53.9 | 54.1 | 57.8 | 59.4 | 45.0 | 51.0 | 52.6 |
| HYBRID | **64.2** | **63.7** | **67.1** | **68.3** | **71.4** | **57.8** | **63.1** | **65.1** |
| HUMAN | 90.2 | 88.4 | 89.8 | 88.6 | 89.9 | 86.7 | 88.1 | 88.8 |

Table 6.5: Models and human performance (F1 score) on the development and the test data. SAR: STANFORD ATTENTIVE READER.

free-form answers (see row *Abstractive* in Table 6.6). When the STANFORD ATTENTIVE READER is fed into PGNET, we empower both STANFORD ATTENTIVE READER and PGNET — STANFORD ATTENTIVE READER in producing free-form answers; PGNET in focusing on the rationale instead of the passage. This combination outperforms the PGNET and the STANFORD ATTENTIVE READER models by 21.0 and 12.5 points respectively.

**Models vs. Humans.** The human performance on the test data is 88.8 F1, a strong agreement indicating that the COQA's questions have concrete answers. Our best model is 23.7 points behind humans, suggesting that the task is difficult to accomplish with current models. We anticipate that using a state-of-the-art reading comprehension model (Devlin et al., 2018) may improve the results by a few points.

**In-domain vs. Out-of-domain.** All models perform worse on out-of-domain datasets compared to in-domain datasets. The best model drops by 6.6 points. For in-domain results, both the best model and humans find the literature domain harder than the others since literature's vocabulary requires proficiency in English. For out-of-domain results, the Reddit domain is apparently harder. This could be because Reddit requires reasoning on longer passages (see Table 6.1).

| Type | SEQ2SEQ | PGNET | SAR | HYBRID | HUMAN |
|---|---|---|---|---|---|
| **Answer Type** | | | | | |
| Answerable | 27.5 | 45.4 | 54.7 | 66.3 | 89.9 |
| Unanswerable | 33.9 | 38.2 | 55.0 | 51.2 | 72.3 |
| Extractive | 20.2 | 43.6 | 69.8 | 70.5 | 91.1 |
| Abstractive | 43.1 | 49.0 | 22.7 | 57.0 | 86.8 |
| Named Entity | 21.9 | 43.0 | 72.6 | 72.2 | 92.2 |
| Noun Phrase | 17.2 | 37.2 | 64.9 | 64.1 | 88.6 |
| Yes | 69.6 | 69.9 | 7.9 | 72.7 | 95.6 |
| No | 60.2 | 60.3 | 18.4 | 58.7 | 95.7 |
| Number | 15.0 | 48.6 | 66.3 | 71.7 | 91.2 |
| Date/Time | 13.7 | 50.2 | 79.0 | 79.1 | 91.5 |
| Other | 14.1 | 33.7 | 53.5 | 55.2 | 80.8 |
| **Question Type** | | | | | |
| Lexical Matching | 20.7 | 40.7 | 57.2 | 65.7 | 91.7 |
| Paraphrasing | 23.7 | 33.9 | 46.9 | 64.4 | 88.8 |
| Pragmatics | 33.9 | 43.1 | 57.4 | 60.6 | 84.2 |
| No coreference | 16.1 | 31.7 | 54.3 | 57.9 | 90.3 |
| Explicit coreference | 30.4 | 42.3 | 49.0 | 66.3 | 87.1 |
| Implicit coreference | 31.4 | 39.0 | 60.1 | 66.4 | 88.7 |

Table 6.6: Fine-grained results of different question and answer types in the development set. For the question type results, we only analyze 150 questions as described in Section 6.2.3.

While humans achieve high performance on children's stories, models perform poorly, probably due to the fewer training examples in this domain compared to others.[6] Both humans and models find Wikipedia easy.

### 6.4.3 Error Analysis

Table 6.6 presents fine-grained results of models and humans on the development set. We observe that humans have the highest disagreement on the unanswerable questions. Sometimes, people guess an answer even when it is not present in the passage, e.g., one can guess the age of *Annie* in Figure 6.1 based on her *grandmother*'s age. The human agreement on abstractive answers is lower than on extractive answers. This is expected because our evaluation metric is based on word overlap rather than on the meaning of words. For the question *did Jenny like her new room?*, human answers *she loved it* and *yes* are both accepted.

Finding the perfect evaluation metric for abstractive responses is still a challenging problem (Liu et al., 2016) and beyond the scope of our work. For our models' performance, SEQ2SEQ and PGNET perform well on the questions with abstractive answers, and STANFORD ATTENTIVE READER performs well on the questions with extractive answers, due to their respective designs. The combined model improves on both categories.

Among the lexical question types, humans find the questions with lexical matches the easiest followed by paraphrasing, and the questions with pragmatics the hardest — this is expected since questions with lexical matches and paraphrasing share some similarity with the passage, thus making them relatively easier to answer than pragmatic questions. The best model also follows the same trend. While humans find the questions without coreferences easier than those with coreferences (explicit or implicit), the models behave sporadically. It is not clear why humans find implicit coreferences easier than explicit coreferences. A conjecture is that implicit coreferences depend directly on the previous turn whereas explicit coreferences may have long distance dependency on the conversation.

**Importance of conversation history.** Finally, we examine how important the conversation history is for the dataset. Table 6.7 presents the results with a varied number of previous turns used as conversation history. All models succeed at leveraging history but only up to a history of one previous turn (except PGNET). It is surprising that using more turns could decrease the performance.

[6]We collect children's stories from MCTest which contains only 660 passages in total, of which we use 200 stories for development and test.

| history size | SEQ2SEQ | PGNET | SAR | HYBRID |
|---|---|---|---|---|
| 0 | 24.0 | 41.3 | 50.4 | 61.5 |
| 1 | 27.5 | 43.9 | 54.7 | 66.2 |
| 2 | 21.4 | 44.6 | 54.6 | 66.0 |
| all | 21.0 | 45.4 | 52.3 | 64.3 |

Table 6.7: Results on the development set with different history sizes. History size indicates the number of previous turns prepended to the current question. Each turn contains a question and its answer. SAR: STANFORD ATTENTIVE READER.

We also perform an experiment on humans to measure the trade-off between their performance and the number of previous turns shown. Based on the heuristic that short questions likely depend on the conversation history, we sample 300 one or two word questions, and collect answers to these varying the number of previous turns shown.

When we do not show any history, human performance drops to 19.9 F1 as opposed to 86.4 F1 when full history is shown. When the previous question and answer is shown, their performance boosts to 79.8 F1, suggesting that the previous turn plays an important role in making sense of the current question. If the last two questions and answers are shown, they reach up to 85.3 F1, almost close to the performance when the full history is shown. This suggests that most questions in a conversation have a limited dependency within a bound of two turns.

## 6.5 Discussion

So far, we have discussed the COQA dataset and several competitive baselines based on conversational models and reading comprehension models. We hope that our efforts can enable the first step to building conversational QA agents.

On the one hand, we think there is ample room for further improving performance on COQA: our hybrid system obtains an F1 score of 65.1%, which is still 23.7 points behind the human performance (88.8%). We encourage our research community to work on this dataset and push the limits of conversational question answering models. We think there are several directions for further improvement:

- All the baseline models we built only use the conversation history by simply concatenating the previous questions and answers with the current question. We think that there should be better ways to connect the history and the current question. For the questions in Table 6.4, we should build models to actually understand that *his* in the question *What was his name?* refers to *the puppy*, and the question *Where?* means *Where will Sirisena be sworn in?*. Indeed, a recent model FLOWQA (Huang et al., 2018a) proposed a solution to effectively stack single-turn models along the conversational flow and demonstrated a state-of-the-art performance on COQA.

- Our hybrid model aims to combine the advantages from the span prediction reading comprehension models and the pointer-generator network model to address the nature of abstractive answers. However, we implemented it as a pipeline model so the performance of the second component depends on whether the reading comprehension model can extract the right piece of evidence from the passage. We think that it is desirable to build an end-to-end model which can extract rationales while also rewriting the rationale into the final answer.

- We think the rationales that we collected can be better leveraged into training models.

On the other hand, COQA certainly has its limitations and we should explore more challenging and more useful datasets in the future. One clear limitation is that the conversations in COQA are only turns of question and answer pairs. That means the answerer is only responsible for answering questions while she can't ask any clarification questions or communicate with the questioner through conversations. Another problem is that COQA has very few (1.3%) unanswerable questions, which we think are crucial in practical conversational QA systems.

In parallel to our work, Choi et al. (2018) also created a dataset of conversations in the form of questions and answers on text passages. In our interface, we show a passage to both the questioner and the answerer, whereas their interface only shows a title to the questioner and the full passage to the answerer. Since their setup encourages the answerer to reveal more information for the following questions, their answers are as long as 15.1 words on average (ours is 2.7). While the human performance on our test set is 88.8 F1, theirs is 74.6 F1. Moreover, while COQA's answers can be abstractive, their answers are restricted

to only extractive text spans. Our dataset contains passages from seven diverse domains, whereas their dataset is built only from Wikipedia articles about people. Also, concurrently, Saeidi et al. (2018) created a conversational QA dataset for regulatory text such as tax and visa regulations. Their answers are limited to *yes* or *no* along with a positive characteristic of permitting to ask clarification questions when a given question cannot be answered.

# Chapter 7

# Conclusions

In this dissertation, we gave readers a thorough overview of neural reading comprehension: the foundations (PART I) and the applications (PART II), as well as how we contributed to the development of this field since it emerged in late 2015.

In Chapter 2, we walked through the history of reading comprehension, which dates back to the 1970s. At the time, researchers already recognized its importance as a proper way of testing the language understanding abilities of computer programs. However, it was not until the 2010s that, reading comprehension started to be formulated as a supervised learning problem by collecting human-labeled training examples in the form of (passage, question, answer) triples. Since 2015, the field has been completed reshaped, by the creation of large-scale supervised datasets, and the development of neural reading comprehension models. Although it has been only 3 years so far, the field has been moving strikingly fast. Innovations in building better datasets and more effective models have occurred alternately, and both contributed to the development of the field. We also formally defined the task of reading comprehension, and described the four most common types of problems: *cloze style*, *multiple choice*, *span prediction* and *free-form answers* and their evaluation metrics.

In Chapter 3, we covered all the elements of modern neural reading comprehension models. We introduced the STANFORD ATTENTIVE READER, which we first proposed for the CNN/DAILY MAIL cloze style task, and is one of the earliest neural reading comprehension models in this field. Our model has been studied extensively on other cloze style

and multiple choice tasks. We later adapted it to the SQuAD dataset and achieved what was then state-of-the-art performance. Compared to conventional feature-based models, this model doesn't rely on any downstream linguistic features and all the parameters are jointly optimized together. Through empirical experiments and a careful hand-analysis, we concluded that neural models are more powerful at recognizing lexical matches and paraphrases. We also discussed recent advances in developing neural reading comprehension models, including better *word representations*, *attention mechanisms*, *alternatives to LSTMs*, and other advances such as training objectives and data augmentation.

In Chapter 4, we discussed future work and open questions in this field. We examined error cases on SQuAD (for both our model and the state-of-the-art model which surpasses the human performance). We concluded that these models have been doing very sophisticated matching of text but they still have difficulty understanding the inherent structure between entities and the events expressed in the text. We later discussed future work in both models and datasets. For models, we argued that besides *accuracy*, there are other important aspects which have been overlooked that we will need to work on in the future, including *speed and scalability*, *robustness*, and *interpretability*. We also believe that future models will need more structures and modules to solve more difficult reading comprehension problems. For datasets, we discussed more recent datasets developed after SQuAD — these datasets either require more complex reasoning across sentences or documents, or need to handle longer documents, or need to generate free-form answers instead of extracting a single span, or predict when there is no answer in the passage. Lastly, we examined several questions we think are important to the future of neural reading comprehension.

In Part II, the key questions we wanted to answer are: Is reading comprehension only a task of measuring language understanding? If we can build high-performing reading comprehension systems which can answer comprehension questions over a short passage of text, can it enable useful applications?

In Chapter 5, we showed that we can combine information retrieval techniques and neural reading comprehension models to build an open-domain question-answering system: answering general questions over a large encyclopedia or the Web. In particular, we implemented this idea in the DrQA project, a large-scale, factoid question answering system over English Wikipedia. We demonstrated the feasibility of doing this by evaluating

the system on multiple question answering benchmarks. We also proposed a procedure to automatically create additional distantly-supervised training examples from other question answering resources and demonstrated the effectiveness of this approach. We hope that our work takes the first step in this research direction and this new paradigm of combining information retrieval and neural reading comprehension will eventually lead to a new generation of open-domain question answering systems.

In Chapter 6, we addressed the conversational question answering problem, where a computer system needs to understand a text passage and answer a series of questions that appear in a conversation. To approach this, we built CoQA: a Conversational Question Answering challenge for measuring the ability of machines to participate in a question-answering style conversation. Our dataset contains 127k questions with answers, obtained from 8k conversations about text passages from seven diverse domains. We also built several competitive baselines for this new task, based on conversational and reading comprehension models. We believe that building such systems will play a crucial role in our future conversational AI systems.

All together, we are really excited about the progress that has been made in this field for the past 3 years and have been glad to be able to contribute to this field. At the same time, we also deeply believe there is still a long way to go towards genuine human-level reading comprehension, and we are still facing enormous challenges and a lot of open questions that we will need to address in the future. One key challenge is that we still don't have good ways to approach deeper levels of reading comprehension — those questions which require understanding the reasoning and implications of the text. Often this occurs with *how* or *why* questions, such as *In the story, why is Cynthia upset with her mother?*, *How does John attempt to make up for his original mistake?* In the future, we will have to address the underlying science of what is being discussed, rather than just answering from text matching, to achieve this level of reading comprehension.

We also hope to encourage more researchers to work on the applications, or apply neural reading comprehension to new domains or tasks. We believe that it will lead us towards building better question answering and conversational agents and hope to see these ideas implemented and developed in industry applications.

# Bibliography

David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Müller, Maarten de Rijke, and Stefan Schlobach. 2004. Using Wikipedia at the TREC QA Track. In *Text REtrieval Conference (TREC)*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *North American Association for Computational Linguistics (NAACL)*, pages 1545–1554.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, pages 2425–2433.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735. Springer.

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

Ondrej Bajgar, Rudolf Kadlec, and Jan Kleindienst. 2016. Embracing data abundance: BookTest dataset for reading comprehension. *arXiv preprint arXiv:1610.00956*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Petr Baudiš. 2015. YodaQA: a modular question answering system pipeline. In *POSTER 2015—19th International Student Conference on Electrical Engineering*, pages 1156–1165.

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the YodaQA system. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 222–228. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics (TACL)*, 5:135–146.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

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

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 257–264.

Davide Buscaldi and Paolo Rosso. 2006. Mining knowledge from Wikipedia for the question answering task. In *International Conference on Language Resources and Evaluation (LREC)*, pages 727–730.

Eugene Charniak, Yasemin Altun, Rodrigo de Salvo Braz, Benjamin Garrett, Margaret Kosmala, Tomer Moscovich, Lixin Pang, Changhee Pyo, Ye Sun, Wei Wy, et al. 2000. Reading comprehension programs in a statistical-language-processing class. In *ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems*, pages 1–5.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Conference of the International Speech Communication Association (Interspeech)*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*, volume 1, pages 2358–2367.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1870–1879.

Kyunghyun Cho. 2015. Natural language understanding with distributed representation. *arXiv preprint arXiv:1511.07916*.

Kyunghyun Cho, Bart Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

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 *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2174–2184.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Association for Computational Linguistics (ACL)*, volume 1, pages 845–855.

Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. 2017. DAWNBench: An end-to-end deep learning benchmark and competition. In *NIPS ML Systems Workshop*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, Jose MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Conference on computer vision and pattern recognition (CVPR)*, pages 1080–1089.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W Cohen. 2017a. A comparative study of word embeddings for reading comprehension. *arXiv preprint arXiv:1703.00993*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017b. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V 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*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Association for Computational Linguistics (ACL)*, volume 1, pages 889–898.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1019–1027.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational AI. *arXiv preprint arXiv:1809.08267*.

Yoav Goldberg. 2017. *Neural network methods for natural language processing*, volume 10. Morgan & Claypool Publishers.

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics (ACL)*, pages 1631–1640.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2943–2952.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Conference on computer vision and pattern recognition (CVPR)*, pages 770–778.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks Principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR)*.

Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. 1999. Deep read: A reading comprehension system. In *Association for Computational Linguistics (ACL)*, pages 325–332.

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

Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. 2018a. FlowQA: Grasping flow in history for conversational machine comprehension. *arXiv preprint arXiv:1810.06683*.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2018b. FusionNet: Fusing via fully-aware attention with application to machine comprehension. In *International Conference on Learning Representations (ICLR)*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1821–1831.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2021–2031.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1601–1611.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? A critical investigation of popular benchmarks. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 5010–5015.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? Textbook question answering for multimodal machine comprehension. In *Conference on computer vision and pattern recognition (CVPR)*, pages 5376–5384.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *North American Association for Computational Linguistics (NAACL)*, volume 1, pages 252–262.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Walter Kintsch. 1998. *Comprehension: A paradigm for cognition.* Cambridge University Press.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. pages 67–72.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association of Computational Linguistics (TACL)*, 6:317–328.

Julian Kupiec. 1993. MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. In *ACM SIGIR conference on Research and development in information retrieval*, pages 181–190.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 785–794.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

Wendy Grace Lehnert. 1977. *The process of question answering*. Ph.D. thesis, Yale University.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 107–117.

Tao Lei, Yu Zhang, Sida I Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 4470–4481.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *North American Association for Computational Linguistics (NAACL)*, pages 110–119.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1736–1745.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike 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 in Natural Language Processing (EMNLP)*, pages 2122–2132.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1694–1704.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421.

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.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6297–6308.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2017. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. ParlAI: A dialog research software platform. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–84.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1400–1409.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics (ACL)*, pages 1003–1011.

Tom M Mitchell, Justin Betteridge, Andrew Carlson, Estevam Hruschka, and Richard Wang. 2009. Populating the semantic web by macro-reading internet text. In *International Semantic Web Conference (IWSC)*, pages 998–1002.

Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. 2000. The structure and performance of an open-domain question answering system. In *Association for Computational Linguistics (ACL)*, pages 563–570.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1253–1262.

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. *arXiv preprint arXiv:1611.09268*.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2230–2235.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL)*, pages 311–318.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2249–2255.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*, pages 1470–1480.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*, volume 1, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Martin Raison, Pierre-Emmanuel Mazaré, Rajarshi Das, and Antoine Bordes. 2018. Weaver: Deep co-encoding of questions and documents for machine reading. *arXiv preprint arXiv:1804.10490*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, volume 2, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association of Computational Linguistics (TACL)*. Accepted pending revisions.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 193–203.

Ellen Riloff and Michael Thelen. 2000. A rule-based question answering system for reading comprehension tests. In *ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems*, pages 13–19.

Pum-Mo Ryu, Myung-Gil Jang, and Hyun-Ki Kim. 2014. Open domain question answering using Wikipedia-based knowledge model. *Information Processing & Management*, 50:683–692.

Mrinmaya Sachan, Kumar Dubey, Eric Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Association for Computational Linguistics (ACL)*, volume 1, pages 239–249.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2087–2097.

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse

over linked question answer pairs with a knowledge graph. In *Conference on Artificial Intelligence (AAAI)*.

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

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1073–1083.

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

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Neural speed reading via Skim-RNN. In *International Conference on Learning Representations (ICLR)*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1683–1692.

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

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2377–2385.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 4208–4219.

Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. 2017. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Association for Computational Linguistics (ACL)*, volume 1, pages 806–817.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics (NAACL)*, volume 1, pages 641–651.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. MovieQA: Understanding stories in movies through question-answering. In *Conference on computer vision and pattern recognition (CVPR)*, pages 4631–4640.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Ellen M Voorhees. 1999. The TREC-8 question answering track report. In *Text REtrieval Conference (TREC)*, pages 77–82.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Association for Computational Linguistics (ACL)*, volume 2, pages 700–706.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using Match-LSTM and answer pointer. In *International Conference on Learning Representations (ICLR)*.

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

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *International Conference on Learning Representations (ICLR)*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Association for Computational Linguistics (ACL)*, volume 1, pages 189–198.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning (ICML)*, pages 1113–1120.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *3rd Workshop on Noisy User-generated Text*, pages 94–106.

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

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *International Conference on Learning Representations (ICLR)*.

Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270.

Pengtao Xie and Eric Xing. 2017. A constituent-centric neural architecture for reading comprehension. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1405–1414.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR)*.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2369–2380.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. 2014. Freebase QA: Information extraction or semantic parsing? In *ACL 2014 Workshop on Semantic Parsing*, pages 82–86.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations (ICLR)*.

Adams Wei Yu, Hongrae Lee, and Quoc Le. 2017. Learning to skim text. In *Association for Computational Linguistics (ACL)*, volume 1, pages 1880–1890.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Association for Computational Linguistics (ACL)*, volume 1, pages 2204–2213.