Towards a Question Answering View of Natural Language Processing

Submitted by

Andrej Žukov-Gregorič

for the degree of Doctor of Philosophy

of the

Royal Holloway, University of London

2018

Declaration

I, Andrej Žukov-Gregorič, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed .... (Andrej Žukov-Gregorič)
Date:

To my parents

Abstract

In recent years natural language processing research has tended to evolve through the successive modelling of various tasks. Examples include part-of-speech tagging, named entity recognition, co-reference resolution, various parsing tasks, language modelling, dialogue state tracking, extractive question answering (also called reading comprehension), answer sentence selection, text summarization, machine translation, and even visual question answering. The goal of this thesis is threefold: to provide new architectures for a subset of the above; to propose a new model or task capable of generalising some of the above; finally, to build a dataset suitable for extractive question answering. We present results close to or above the state-of-the-art in named entity recognition (91.48 F1 on the CoNLL-2003 English dataset) and answer sentence selection (a 70.1% and 67.4% accuracy on the two test sets of InsuranceQA). Additionally, we provide a detailed ablation study of the well-known Bi-directional Attention Flow (BiDAF) model which we then use to suggest improvements to it. Finally, we introduce a new question answering dataset of labelled news data from armed-conflict events in the Iraq war.

Acknowledgement

I would like to thank my supervisor Zhiyuan Luo for initiating my interest in NLP. I started my PhD working on conformal prediction, followed by work on the probabilistic modelling of the spread of information across networks. I did not make much headway in these topics until I, by chance, was introduced to conditional random fields by Zhiyuan. This thesis is a testament to that moment. I would also like to thank our Department of Computer Science at Royal Holloway, as well as the Slovene Scholarship Fund for supporting me throughout my PhD.

In no particular order I would also like to thank my friends and colleagues: Dmitry Adamskiy, Sam Coope, Haotian Chen, Mayur Ladwa, Yoram Bachrach, Ed Tovall, Conan McMurtrie, Pasha Minkovsky, Attila Brozik, David Li, James Smith, Gregory Chockler, Jamie Alnasir, Michael Spagat, Florentyna Farghly, Pauline Heinrichs, Tim Scarfe, Han-Yi Chen, Ali Ahmed, Sam Robinson, Špela Drnovšek-Zorko, Filip Drnovšek-Zorko, Joonas Karjaleinen, Raphael Reuben, Bryn Adams, Daniel Partington, Sahil Wadhwa, Sebastian Janisch, Gabriel Hochstatter, Guy Doza, Nahom Shewaybel, Sandford Sun, Maja Hakalo, Borjana Flipovska, Jasa Andrenšek, Luka Pilko, Volodya Vovk, James Smith, Asiana Jurca Avci and many, many, others for interesting technical conversations and for keeping me sane throughout my degree.

To my parents, Bojana and Miloš: none of this would have been possible without you. All your traits; your curiosity, your intellect, your strength and your love, you have selflessly, and through more sacrifice than most, given to me. For this, I am and forever will be, grateful.

Finally, to Samantha Si. To goose. Without you, I would be like a ship lost at sea.

Contents

List of Figures

List of Tables

List of Abbreviations

ACL	Association of Computational Linguistics
ADAM	Adaptive Moment Estimation
AMTCL	Association for Machine Translation and Computational Linguistics
BiDAF	Bi-Directional Attention Flow
BoW	Bag of Words
CBOW	Continuous Bag of Words
CBT	Children's Book Test
CNN	Convolutional Neural Network
CRF	Conditional Random Field
EM	Exact Match
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
IBC	Iraq Body Count
IE	Information Extraction
IR	Information Retrieval
KB	Knowledge Base
LSTM	Long Short Term Memory
MUC	Message Understanding Conference
MRR	Mean Reciprocal Rank
MT	Machine Translation
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part of Speech
QA	Question Answering
RNN	Recurrent Neural Network
SEA	Semantically Equivalent Adversary
SQuAD	Stanford Question Answering Dataset
SVM	Support Vector Machine
TF	Term Frequency
TREC	Text Retrieval Conference

Chapter 1

Introduction

Language is fundamentally inquisitive. Not only do we use it to *convey* information, we also use it to *inquire* for new information. Historically, most written text was of the former kind but with today's popularity of messaging applications, we have seen an explosion in the amount of written data we have of the latter. This in turn has spurred research into *question answering* (QA) which is the study of models capable of answering questions across data.

The *natural language processing* (NLP) community divides question answering into two main categories depending on what the questions are being asked across. If across structured data, it is called *structured QA*, whereas if across unstructured data, it is called *text QA*. The two divisions have developed a separate focus.

Text QA research has focused heavily on attention models capable of detecting affinities between text and question. Much work has been done on detecting, mostly contiguous, answer spans in text. This has led to claims of human-level performance on certain well-known datasets. Recently however, there has been mounting evidence that suggests these models often learn to answer in a wrong way. They tend to focus on wrong parts of the question which makes them brittle in new contexts. There have been two main ways of remedying this. One has been through new datasets, the other through better inductive biases in the models themselves. Both strategies serve to help prevent models from learning spurious forms of reasoning.

Structured QA research has focused on two main data sources: *knowledge bases* (KBs) and tables. In the case of KBs, affinities between question and the contents of the KB (usually in the form of KB triplets) are modeled. Work on QA across tables on the other hand has seen attention from the neural semantic parsing community. Here, the mapping between question and logical form (such as a SQL query which then gets run across a table) is modeled. The logical form can be thought of as a blueprint for how to reason your way to the answer. Its generation can even be constrained to be both syntactically and semantically correct. However, it is unclear how this could be applied across unstructured data.

Many well known NLP tasks can be turned into a question answering task. For example, *named entity recognition* (NER) can be seen as a multi-span QA task. Similarly, much of *information extraction* (IE) can be seen as just QA across documents. Even seemingly unrelated tasks such as parsing can be thought of as questions answering across text where the goal is to generate a parse. This unifying aspect of QA has helped open the door for the study of *transfer* and *multi-task learning* across different domains in NLP. For a long time NLP was forced to focus on small independent tasks; different models were developed and tailored for specific tasks like co-reference resolution, part-of-speech tagging, or constituency parsing. They were then pitted against each other on open datasets until they were "solved" only to have new, harder, datasets appear. Over time new datasets introduced harder and harder problems such as tasks involving dialogue or elements of computer vision. Concurrently, the unsupervised learning of word and sentence-level embeddings has armed downstream QA models with better inputs that have led to spikes in performance. Nevertheless, NLP is yet to have its ImageNet moment and although at least in question answering, the predominant mood now seems to be that of taking a step back and re-evaluating what it is our models are trying to learn.

All this, of course, matters, not only in and of itself, but also because of its use in application. Written questions nowadays appear in two main contexts: search and, more recently, dialogue (whether with an automated customer support agent or a home assistant). The proliferation of human-computer interaction is undoubtedly set to continue and the task of NLP research is to help equip it with ever better tools. This is not to say that the relationship between research and application is a one-way street, for surely it has been application which has over years heavily dictated the interests and directions of researchers in NLP. One need not look further than tasks which have preoccupied NLP for decades such as *machine translation* (MT) or more recent tasks such as . *sentiment analysis*, both driven by applicative need. In the former case, as a means of translating Cold War documents from Russian into English en masse; in the latter, as a way of predicting market sentiment in financial modeling or as a gauge of the quality of a product. Similarly, with question answering, it is clear that until we one day have fully automated customer service centers or home assistants with more human-like capabilities, applications such as these will continue to guide research in QA.

The goal of this thesis is to explore and extend our current understanding of text

QA by focusing on three things: (1) the pitfalls of existing QA models, (2) entities and how they relate to QA, and (3) a new dataset created for QA across armed-conflict related texts. The thesis is divided into eight chapters and starts with chapter two where question answering, its history, and how it is viewed in linguistics is described. This is followed by a discussion of various different neural models used in the field in chapter three. The first substantive chapter with contributions is chapter four where we present a detailed ablation study of one of the most popular QA models and propose a new named entity QA task. Chapter five focuses on named entity recognition. Chapter six introduces a new answer selection model. Chapter seven introduces a new dataset for armed conflict analysis and presents results on it. Finally, chapter eight concludes this thesis with proposals for future work.

1.1 Contributions

The contributions in this thesis are contained in Chapters 4, 5, 6, and 7. We describe them briefly in turn:

Chapter 4

Chapter four presents an ablation study on the BiDAF reading comprehension model. We show that: (1) the model relies on its strong attention mechanisms to achieve good results (2) it is not sensitive to the complexity or simplicity of the question representation (3) we propose a new form of attention which does not prematurely compress the question (4) we introduce a new dataset which combines reading comprehension with NER (5) we introduce an extension of the BiDAF model to the multi-span setting which we call BiDAF-MS.

Chapter 5

Chapter five introduces a new multi-LSTM model which is an end-to-end neural architecture comprised of multiple ensemble-like bi-directional LSTMs. We show that our models achieves state of the art results on the CoNLL 2003 NER dataset.

Chapter 6

Chapter six focuses on another QA task called answer selection and introduces a new answer selection model which achieves state-of-the-art results on the InsuranceQA answer selection dataset. We achieve this by using a siamese architecture augmented with attention and a global view of the question and candidate answer.

Chapter 7

Chapter seven introduces a new dataset of annotated armed-conflict news data. Using human-coded incident-level data about casualties in the ongoing conflict in Iraq we construct a base dataset which can be posited as either a NER, relationship extraction, event de-duplication, or question answering dataset.

1.2 List of Papers

The thesis is in part based on the following papers:

- **Named Entity Recognition With Parallel Recurrent Neural Networks.** [39].
- **An Attention Mechanism for Neural Answer Selection Using a Combined Global and Local View** [7, 6].
- **Neural Named Entity Recognition Using a Self-Attention Mechanism** [40].
- **IBC-C: A Dataset for Armed Conflict Analysis** [41].

Part I

Background

Chapter 2

Question Answering

This chapter provides a thorough background on the topic of the thesis. The literature on this particular topic is huge, but we will try our best.

Question answering has a long history in both computing and natural language processing. This chapter begins by introducing the history behind question answering, followed by a review of what the areas of focus are today.

2.1 The History of Question Answering

Question can be thought of as utterances which we employ to retrieve information. Whether from or own memory or from some external context. As such, it is easy to see the central role question answering plays in natural language processing.

2.1.1 A brief history of NLP

Natural language processing is a very young field. We can trace its history back to the late 1940s. World War II had just ended and academia was spoiled for research directions to pursue. An entire generation of mathematicians, linguists, and engineers, on both sides, had just spent years contributing towards the war effort of their respective sides. Many of them focused on code breaking and the machinery which made it feasible at scale - early computers. Code breakers developed a stable of techniques, runnable on the electromechanical or early electronic computers of the time or by so-called *human computers* - groups of primarily women whose job it was to operate and program the computers as well hand-calculate or transcribe things when automated methods were found to be unfeasible. The code breaking techniques involved counting character, word, or n-gram frequencies; comparing documents; and through other means statistically analyzing cyphertext and how it relates to plaintext. With the war over, questions naturally arose over whether the computers and techniques developed in code breaking could be applied to *natural* language instead of code.

This was a time of unprecedented international cooperation. International organizations such as the World Bank, the International Monetary Fund, the United Nations and its various agencies, had just been formed. The Nuremberg trials were ongoing, the Marshall plan was about to begin, and suspicions between the West and the Soviet Union festered. What a burning need, all of a sudden, to understand the world's languages! And so, between roughly 1946 and 1949, Warren Weaver, a departmental director at the Rockefeller Foundation and a mathematician who had spent the war in operations research but who was familiar with communication theory and cryptography, began thinking about what would later become *machine translation* (MT). The result of his thoughts was a memorandum, entitled *Translation*, which he completed in 1949. In it he cites a wonderful exchange between him and Norbert Wiener of MIT:

> *One thing I wanted to ask you about is this. A most serious problem, for UNESCO and for the constructive and peaceful future of the planet, is the problem of translation, as it unavoidably affects the communication between peoples. Huxley has recently told me that they are appalled by the magnitude and the importance of the translation job.*
>
> *Recognizing fully, even though necessarily vaguely, the semantic difficulties because of multiple meanings, etc., I have wondered if it were unthinkable to design a computer which would translate. Even if it would translate only scientific material (where the semantic difficulties are very notably less), and even if it did produce an inelegant (but intelligible) result, it would seem to me worth while.*
>
> *Also knowing nothing official about, but having guessed and inferred considerable about, powerful new mechanized methods in cryptographymethods which I believe succeed even when one does not know what language has been codedone naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."*
>
> *Have you ever thought about this? As a linguist and expert on computers, do you think it is worth thinking about?*

Wiener replied in the negative:

> *Second - as to the problem of mechanical translation, I frankly am afraid the boundaries of words in different languages are too vague and the emotional and international connotations are too extensive to make any quasi-mechanical translation scheme very hopeful. I will admit that basic English seems to indicate that we can go further than we have generally done in the mechanization of speech, but you must remember that in certain respects basic English is the reverse of mechanical and throws upon words a burden which is much greater than most words carry in conventional English. At the present time, the mechanization of language, beyond such a stage as the design of photoelectric reading opportunities for the blind, seems very premature. . . .*

Despite not having convinced Wiener, Weaver convinced many others including policy makers which responded with plentiful funds to fund research into the topic. In tandem with Weaver, disparate efforts elsewhere were ongoing. In particular, Andrew D. Booth of Birkbeck College and Richard H. Richens (a botanist turned computational linguist) of the Commonwealth Bureau of Plant Breeding and Genetics were by then working on a mechanized dictionary capable of translating individual words using a rule-based method. Similar efforts were being worked on, on one of the computers in California.

Research into machine translation soon opened other areas of focus which needed to be addressed such as syntactics and semantics. From this, new grammars, parsers, and what today we would call lexical databases were created. As these sub-problems turned into sub-fields, modern NLP slowly began to be born. In 1954 the ability to translate simple technical Russian into English was showcased in the IBM-Georgetown demonstration. In 1962 the *Association for Machine Translation and Computational Linguistics* (AMTCL) was formed - later to be renamed to today's well known *Association for Computational Linguistics* (ACL) in 1968.

Progress in MT was nevertheless slow. Despite early optimism that machine translations would be solved within years, the end was nowhere in sight. In 1964 the US government set up the *Automatic Language Processing Advisory Committee.* Their task: to evaluate progress in machine translation and computational linguistics. Their 1966 report titled *Computers in Translation and Linguistics* was critical of the progress achieved and as a result funding for MT was heavily curtailed. The report emphasized the need for more basic research in computational linguistics and as a result NLP became more diversified.

The 1970s and 80s saw researchers focus on ways of representing contextual knowledge and systems capable of building and answering questions about these early knowledge bases. Lexical databases such as WordNet and other knowledge bases based on semantic frames, were developed. Work in these areas complemented and intersected with work being done on expert systems. This was the era of great optimism that software systems could help humans solve tasks faster. Systems exposed natural language interfaces to be able to query structured data. SHRDLU (1970) could answer questions and enact prompts in a simple block world; PARRY (1972) was an early chatbot which attempted to simulate a person with paranoid schizophrenia; KL-ONE (1974) reasoned across knowledge bases, SAM (1975) and PAM (1978) understood stories and could launch procedures as a result or answer questions about them. These systems in turn spurred research into more basic NLP research such as statistical parsing, part-of-speech tagging *POS tagging*, knowledge-base creation, and intent recognition.

In the early 1980s it was hoped that natural language interfaces would be the *de facto* way of interfacing with computers (instead of having to type complicated commands into the terminal). Entire companies such as Symantec were founded on this premise. Instead, with the advent of GUI operating systems (Mac OS 1.0 in 1984 quickly followed by Windows 1.0 and AmigaOS 1.0 in 1985), curtailed research into expert systems and natural language interfaces.

Running in parallel with the work being done on the above, research into the statistical analysis of language started gaining traction. *Hidden Markov models*, which had been developed in the late 1950s and throughout the 1960s by the likes of Ruslan Stratonovich and Leonard E. Baum (known for the *Baum-Welch* algorithm), saw their application first in speech recognition and later in tasks which could be posited as sequence classification problems such as POS tagging and eventually *named entity recognition* (NER).

The 1990s saw an explosion in statistical NLP methods as the machine learning and NLP communities edged ever closer. Structured *support vector machines* (SVMs), *conditional random fields* (CRFs), recurrent neural networks (RNNs) [1] all started to be applied to language by the end of the decade. The standardization and proliferation of NLP tasks allowed the community to more easily compete on solving them. Conferences such as the first Text Retrieval Conference (TREC, 1992), SemEval-1 (1998), and Conll-1999, promoted shared tasks which saw researchers around the world compete on *information retrieval* (IR) and computational semantics analysis tasks.

The 2000s saw a maturing of these methods as researchers focused on an ever-growing list of shared tasks, various flavors of models, and perhaps what looking back would mark the decade - hand-engineered features. By the end of the decade, intricate models, complicated pipeline systems, and baroque hand-engineered features had

[1]Not *long-short term memory* (LSTM) networks yet, although they were already around.

achieved measurable progress in most tasks. Simple tasks such as POS tagging and NER were considered close to being solved and much progress had been made on tasks such as co-reference resolution, semantic role labeling, word sense disambiguation, and sentiment analysis. Important caveats where the then state-of-the-art models failed remained. Pointedly, the way this was tackled was usually by adding more and better hand-engineered features.

An important shift happened in the early 2010s with a few key incursions of deep learning into the field. Distributed word representations in vector space appeared in 2011 [2] and were shown to substantially improve the performance of downstream models. As a direct result of this research, study into distributed word vector space models has blossomed and researchers are now trying to encode as much information into word embeddings as possible. In 2012 the now widely used sequence-to-sequence model was developed, immediately improving results in machine translation and semantic parsing. It was in the 2010s that mobile phones went from being devices used to call and text people to so-called *smartphones* - devices with which users could book flights, bank, check cycle routes, video call, book a cab, and, towards the end of the decade, even begin to use as smart personal assistants. Despite it being easier than ever before to place a call, smart phones users in the 2010s turned unexpectedly to messaging each other. This has reignited hopes that natural language, whether through voice or text, could again be used to interface with complex systems. As a result, study into dialogue systems saw increased interest, as did some of its sub-tasks, including semantic parsing and question answering.

At the time of writing it is late 2018 and we are slowly edging towards the end of another decade. Looking forward we can probably expect work in the next decade to continue to push the boundaries of deep learning NLP models. We can expect to see word and sentence or document level embedding models to mature as we find better ways of encoding information in them. This will lead to significant improvements in downstream tasks. Just as in the computer vision community, we are beginning to see more attention being paid to the robustness of our models and their brittleness to adversarial examples. Dialogue systems will improve as households acquire smart home assistants en masse, as a result we can expect continued interest in dialogue state tracking, question answering, and semantic parsing. A tougher nut to crack will be problems which we have only skirted the surface of such as true transfer learning and how memories are stored, retrieved, and used.

And what of machine translation? The original problem which helped spark re-

[2]The first such models was actually introduced by Rumelhart, Hinton, and Williams in 1986 but distributed word representations first really saw their use in the early 2010s.

search into what would become modern NLP? As of 2018, the strongest neural machine translations systems perform close to par with humans.

2.1.2 A brief history of QA

Question answering traces its history back to the early days of NLP. From the very beginning there was a distinction between *structured QA* and *text QA*. The two branches distinguished themselves not only by the types of data questions were asked across but also by their application. *Structured QA* was seen as a means of providing an interface to complex structured data whereas text QA was initially seen as a frontend to *information retrieval*. Later, text QA would become an object of separate study as part of *reading comprehension*.

In 1961 researchers at MIT revealed BASEBALL [38], a system capable of automatically answering questions across a structured baseball dataset. Its introduction confidently stated that:

> Men typically communicate with computers in a variety of artificial, stylized, unambiguous languages that are better adapted to the machine than to the man. For convenience and speed, many future computer-centered systems will require men to communicate with computers in natural language. The business executive, the military commander, and the scientist need to ask questions of the computer in ordinary English, and to have the computer answer questions directly. Baseball is a first step toward this goal.

In 1965 a survey by Simmon et al. [128] mentioned no fewer than fifteen English language QA systems built over the preceding five years including early conversational agents. In the mid 1970s work by Wendy Lehnert influenced by Roger Schank and conceptual dependency theory introduced *QUALM* [74], a question answering program capable of answering questions about text. It was later used as a part of the *SAM* and *PAM* story understanding systems. Lehnert was one of the first NLP researchers to focus on *text QA* as an area of independent study. QUALM worked by first classifying the question into one of thirteen conceptual categories before using heuristics to find the answer.

Despite having studied text QA *in situ*, QUALM failed to spark research into the new field. Perhaps one of the most decisive reasons for this was the lack of a standardized dataset, or task, to help research coalesce around a shared goal. This at last happened with the inclusion of a QA task into the 1999 *Text Retrieval Conference*

(TREC) [139] where the task was to retrieve a ranked list of five documents and corresponding answer snippets (short passage of text which contain the answer). As a result, research into QA has continued to blossom over the past 20 years.

Initially, researchers used pipeline approaches to try and tackle these new shared QA tasks. A common pipeline was:

1. **Question Analysis** questions would first be analyzed and classified as belonging to a particular type using either heuristics or through some machine learning method (see [78]).
2. **Candidate Document Selection** A limited subset of candidate documents believed to contain the answer would be selected.
3. **Answer Extraction** Items at varying granularities depending on the task would be ranked or classified in terms of their probability of containing the answer. Items here could range from the individual documents themselves down to sub-strings of the documents, individual sentences, or even short sequences of words.
4. **Response Generation** An appropriate response to be returned to the user was generated or selected.

With continued adoption of statistical and later deep learning methods in NLP the above pipeline began shedding heuristics and adopting a more end-to-end approach.

In what follows we cover some of the recent work done in the various sub-tasks of QA that have matured over the past two decades with a special focus on reading comprehension and sentence selection - the two main topics of this thesis.

2.2 Question Answering Research Today

Question answering, just like the rest of NLP research today, slots almost wholly within the prevailing deep learning paradigm. Deep learning methods have consistently achieved state-of-the-art results in the field and progress has been rapid.

The reason behind the move in NLP from the more traditional, statistical, approaches to deep learning is for the most part because deep learning methods allow for the more expressive modelling of textual inputs and as a result have achieved much better results on NLP tasks. As an example, modeling the long and short distance sequential dependencies in variable length text is a lot easier to achieve using recurrent neural networks than more traditional approaches which would have either disregarded these dependencies or modelled them using models not as expressive or as general as recurrent neural networks.

2.2.1 Open-Domain Question Answering and Sentence Selection

Open-domain QA is the task of finding answers in collections of unstructured documents. Today, it roughly corresponds to steps 2. and 3. of the previous section. With emphasis on the lots of documents! It is the most closely related sub-field of QA with information extraction and as such also closely related to some of the technology underpinning modern search engines. It is also the branch of QA championed by TREC-QA between 1999 and 2007. The emphasis was always on having to handle large numbers of documents which makes it a great problem to think of in terms of learning to rank all the documents. In 2007, the now somewhat confusingly named TrecQA, or sometimes QASent, dataset was created using some of the original TREC data. It contained a total of 227 questions and 8,478 candidate answers. Significantly, the dataset contained only question-answer pairs with overlapping words.

Recently, research has coalesced around a number of frequently used datasets which try to address some of the deficiencies observed in the original TREC data.

WikiQA [150], released in 2015, is an order of magnitude larger than the datasets released by TREC-QA and focuses on the candidate document selection step (now more commonly called sentence selection or answer selection) from a set of pre-selected sentences. A total of 3,047 questions and 29,258 candidate answer sentences are collected from Bing search queries and Wikipedia summaries respectively. Notably, some questions *do not* have any corresponding correct answer sentences. Further, unlike many instances in TREC, WikiQA data significantly reduces the bias towards lexical overlap (shared words) between question and candidate answer text.

InsuranceQA [26], also released in 2015, expanded the size of sentence selection datasets again this time including many more questions (17,487) and a comparable number of candidate answers (24,981). Released by IBM, the dataset is noteworthy for focusing on a single domain - insurance - using actual questions searched for by users and featuring answers written by experts. Put otherwise, the dataset came from a real-world use case.

Finally, the SemEval-2016 cQA challenge [30], introduced three selection sub-tasks: (1) *question-comment similarity* where the task is to rank the comments left beneath a question according to their relevance (2) *question-question similarity* where the task is to rank a question to a set of other related questions according to their similarity, and (3) *question-external comment similarity* where the task is to, given a new question, rank the 100 comments associated with the 10 most related questions to the new question.

The above datasets have all contributed towards a range of deep learning models which divide roughly into three architectural categories:

- **Siamese Architectures.** First introduced in 1993 by [13], siamese networks work by encoding input into embeddings using the same encoder before comparing the embeddings using some scoring function. The idea is to make the model score correct question-candidate pairs above wrong (so called distractor) pairs by learning an encoder which brings correct pair embeddings closer together in space. Recent examples include the non-attentive model in [134] as well as [108], [44] which even outperform some attentive architectures.
- **Attentive Architectures.** Learning an encoder function can be hard. To make the computation of an embedding more dependent on other inputs, we may attend across them. Attention mechanisms were first described in 2014 in [8] and have seen wide adoption in NLP since. Recent works on answer selection include the attentive model in [134] and the work by [117].
- **Compare-Aggregate Architectures.** The motivation behind compare-aggregate architectures is to delay the aggregation into embeddings to occur as late as possible and to compare as much as possible along the way. Each contextual representation of one input (such as a word) is compare to all contextual representations of the other input thus forming an outer product matrix of representation comparisons. These are then reasoned across using various aggregation steps thus allowing for more fine-grained inter-dependence than with normal attentive architectures. There has been a lot of work very recently on these types of models including initially in work by [45] followed by [144] and [136].

An excellent review of the above methods can be found in [69].

2.2.2 Reading Comprehension

Reading comprehension is the task of reasoning across, usually a single passage, of unstructured text to answer a question. The answer can either be a (1) span extracted from the passage (2) a set of multiple extracted spans (3) an answer picked from a set of multi-choice candidate answers, or (4) generated.

The history of modern reading comprehension dates back to Hirschmann et al. [51] and their work on - the fashionably named - *DEEP READ*, an automated reading comprehension system which learned to find the sentence containing the right answer using a combination of heuristic pattern-matching techniques on hand-engineered features. The data their system was tested against comprised of easy to understand simulated news stories with associated questions of a 3rd-6th grade comprehension difficulty. The system was tested on a small development set made up of 60 passages and applied on a similarly sized test set. Each passage was associated with 5 questions giving a total of 600 questions. *DEEP READ* could identify the sentence containing the correct answer 30-40% of the time. These scores were later to be improved on by others. A year later Ng et al. [95] published the first statistical model (logistic regression) to be used on the dataset achieving results competitive with previous heuristic approaches (this was still a time in NLP when it wasn't clear whether statistical approaches would prevail).

New datasets much like the above were released throughout the 2000s, Breck et al. [12] released 75 stories from the Canadian Broadcasting Corporation's website together with 650 associated questions. This data was then enriched in 2003 by Leidner et al. [75]. On the modelling side, efforts remained much the same and were either heuristics or simple machine learning methods (both of which used hand-engineered features as input).

With increased computational power and the adoption of deep learning methods in NLP towards the end of the 2000s came the need for new datasets and new models. MCTest [114], released in 2013, contains 500 stories and 2000 questions and is a multi-choice reading comprehension task with 4 candidate answers per question.

Neural models continued to struggle on reading comprehension tasks and the best of models only recently began to perform on-par with the best handcrafted ones. Notably, [138] presented at ACL 2016, was the first neural model to outperform handcrafted ones on MCTest by a small margin.

2016 was also the year in which the *Stanford question answering dataset* (SQuAD) was released. Considered somewhat of a watershed moment for reading comprehension, it catapulted reading comprehension into the limelight and there have been over 70 proposed models which have been submitted to its leader board where models compete against one another. Version 1.1 of the dataset contained over 100,000 questions and 536 passages of (so called) context text. The recently released version 2.0 of the dataset [104] adds unanswerable questions into the mix.

The vast majority of SQuAD models are very similar. Roughly speaking they all divide into three layers:

- **Input Layer.** The input layer's responsibility is to take input representations (usually of words, or characters) and transform them into contextual embeddings which better represent the input passage and question. A very common approach is to combine pre-trained word embeddings with character-level embeddings by appending the output of an LSTM or CNN across the characters which make up the corresponding word embeddings. The combined word-character embeddings are then passed through an LSTM to contextualize them. We are left with a sequence of embeddings of the same length as the context (or question depending on what is being embedded) which contain information about their surrounding context and thus represent the inputs better.

- **Body Layer.** The body layer is where reading comprehension models model the dependencies between passage and question. This is done through various attention steps using attention, co-attention and/or self-attention modules. Notable examples include BiDAF [123], Dynamic CoAttention Networks [148], Stochastic Answer Networks [80], QANet [151], FusionNet [55]. We explore BiDAF in greater detail in a subsequent chapter.
- **Answer Layer.** The answer layer takes the outputs of the body layer and converts them into predictions. Predictions in reading comprehension come in two main flavours: (1) span predictions; where the goal is to predict one or more spans of passage text by either predicting the start and end index (or indexes) or by enumerating across all spans in order to classify across them or classify each one of them or (2) generative predictions; where the goal is to generate the correct answer.

Since SQuAD 1.0's release there has been work criticizing the dissecting the dataset's structure and the models which compete on it. My work on analyzing BiDAF can be looked at in this vein. Three recent works stand out here. Jia and Lang (2017) [58] show that simple adversarial examples such as adding extra characters to questions can successfully trick models. Even more recently, [113] formalize adversarial examples for reading comprehension through their semantically equivalent adversarial examples framework which shows that models perform poorly on simple paraphrases. Finally, [141] perform a comparative error study of various reading comprehension models on SQuAD.

Another popular machine comprehension dataset which has seen continuing focus recently is the set of bAbI tasks [147] originally introduced in 2015 and since expanded. Whereas SQuAD tries to cover as broad of a set of topics and questions as possible and to make sure that all text is human generated, bAbI on the other hand is a synthetic dataset with a focus of exploring how comprehension models perform on simple well structured synthetic datasets allowing researchers to see where specifically these models fail.

2.2.3 Cloze-style Tasks

Cloze-style datasets, where the the goal is to predict the missing word in a passage of text, have become a popular way of automatically generating large reading comprehension datasets. Often, named entities are blotted out and the task is to fill them in.

The most popular dataset in the field is the Children's Book Test (CBT) [49] where the goal is to predict a blanked-out word of a sentence given 20 previous sentences. Other examples include the cloze-style CNN / Daily Mail corpus created by [46]. However, as discussed in [105], it is difficult to create hard cloze-style datasets because the blanked out words are usually entities or single words which are heavily dependent on a short context.

2.3 Datasets

In this section, I describe in more detail the datasests used in the thesis.

2.3.1 CoNLL 2003

The CoNLL 2003 NER shared task dataset [137] is a collection sentences where each word is assigned an entity label which identifies it as a *organization*, *person*, *location*, or as *miscellaneous*. The task is most commonly posited as a sequence labelling task. Traditionally, models such as HMMs and CRFs would have been used to solve this task but more recently, research has focused on studying the performance of various recurrent neural architectures.

English data	Articles	Sentences	Tokens
Training set	946	14,987	203,621
Development set	216	3,466	51,362
Test set	231	3,684	46,435

Table 2.1: Number of news articles, sentences, and tokens (words) in the CoNLL-2003 English dataset.

English data	LOC	MISC	ORG	PER
Training set	7140	3438	6321	6600
Development set	1837	922	1341	1842
Test set	1668	702	1661	1617

Table 2.2: Number of named entity types in the CoNLL-2003 English dataset.

U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

Table 2.3: An example sentence from the CoNLL-2003 English dataset.

2.3.2 WikiQA

The WikiQA dataset is an early answer selection dataset [150]. The task is to classify or learn to rank across a number of candidate answers given a question. Questions are usually short, whereas candidate answers are usually multi-sentence paragraphs. In the case of WikiQA, the former are human-coded whereas the latter are collected from Wikipedia.

	Train	Dev	Test	Total
Questions	2,118	296	633	3,047
Sentences	20,360	2,733	6,165	29,258
Answers	1,040	140	293	1,473
Average ques. length	7.16	7.23	7.26	7.18
Average sent. length	25.29	24.59	24.95	25.15
Questions w/o ans.	1,245	170	390	1,805

Table 2.4: WikiQA dataset statistics.

Question type	Counts	%
Location	373	12
Human	494	16
Numeric	658	22
Abbreviation	16	1
Entity	419	14
Description	1,087	36

Table 2.5: WikiQA question type statistics.

Question	Who wrote second Corinthians?
Answer	**Second Epistle to the Corinthians** The Second Epistle to the Corinthians, often referred to as Second Corinthians (and written as 2 Corinthians), is the eighth book of the New Testament of the Bible. Paul the Apostle and Timothy our brother wrote this epistle to the church of God which is at Corinth, with all the saints which are in all Achaia.

Table 2.6: An example question-answer pair from the WikiQA dataset.

2.3.3 InsuranceQA

Similar to WikiQA, InsuranceQA is an answer selection dataset where answers must be classified across or ranked [26].

	Train	Validation	Test1	Test2	Total
Questions	12,887	1000	1,800	1,800	17,487
Answers	18,540	1,454	2,616	2,593	25,203

Table 2.7: InsuranceQA dataset statistics.

Question	Does Medicare cover my spouse?
Answer	If your spouse has worked and paid Medicare taxes for the entire required 40 quarters, or is eligible for Medicare by virtue of being disabled or some other reason, your spouse can receive his/her own medicare benefits. If your spouse has not met those qualifications, if you have met them, and if your spouse is age 65, he/she can receive Medicare based on your eligibility.

Table 2.8: An example question-answer pair from the InsuranceQA dataset.

2.3.4 SQuAD

The Stanford Question Answering Dataset [105] is a machine reading comprehension dataset where the task is to find the correct answer span of words in the *context* text given a *question*. In the case of SQuAD, there is only ever one correct answer span in every *context-question* pair and it is always contiguous (i.e. a sequence of neighbouring words).

	Train	Validation	Test (hidden)
Question-answer pairs	87,599	10,570	9,616
Articles	442	48	46
Context paragraphs	18,891	2,067	2,257

Table 2.9: SQuAD dataset statistics.

Answer type	Percentage	Example
Date	8.9%	19 October 1512
Other Numeric	10.9%	12
Person	12.9%	Thomas Coke
Location	4.4%	Germany
Other Entity	15.3%	ABC Sports
Common Noun Phrase	31.8%	property damage
Adjective Phrase	3.9%	second-largest
Verb Phrase	5.5%	returned to Earth
Clause	3.7%	to avoid trivialization
Other	2.7%	quietly

Table 2.10: SQuAD answer types.

Context paragraph	Nikola Tesla (Serbian Cyrillic: ; 10 July 1856 7 January 1943) was a Serbian American inventor, electrical engineer, mechanical engineer, physicist, and futurist best known for his contributions to the design of the modern alternating current (AC) electricity supply system.
Question	What does AC stand for?
Answer	alternating current

Table 2.11: An example taken from the SQuAD dataset.

2.3.5 IBC-C

Please see Chapter 7 for a further description of IBC-C.

2.4 Evaluation

A few common evaluation metrics are used throughout this thesis. We briefly go over them here.

2.4.1 Precision@k

Precision at k (often denoted as $p@k$) corresponds to the number of relevant results in the k retrieved results. As a result, $p@1$ can be thought of as accuracy whereas, for example, $p@5$ is the ratio of times the correct answer appears in the top 5 retrieved results. We use $p@k$ as a measure of accuracy in answer selection.

2.4.2 MRR

Mean reciprocal rank (MRR) is another metric used to score retrieval models. Retrieval models (including answer selection models which learn to rank) output a sorted list of predictions. The *rank* is the index of the first correct answer to appear in that list. The *reciprocal rank* is just the $rank^{-1}$. And the *mean reciprocal rank* is just the mean of reciprocal ranks.

For example, in a test set made up of three examples where our model ranked the correct answers as $1, 3, 6$; the *reciprocal ranks* would be $1, \frac{1}{3}, \frac{1}{6}$ and the MRR would be $\frac{1}{2}$

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{2.1}$$

2.4.3 F1 Score

The F1 score is defined as the harmonic mean of precision and recall:

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.2}$$

It is probably the single most used metric in NLP. The motivation behind that is that it takes a look at both the recall and precision of a model and not just the accuracy. In cases where there are class imbalances, the discrepancy between the two can be very large and accuracy results can be misleading.

2.4.4 Micro and Macro Scores

Micro- and macro-score on any of the above metrics will compute two different things. A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally), whereas a micro-average will aggregate the contributions of all classes, before computing the average score.

2.5 Summary

We have now reviewed the history of question answering and how it fits in the wider NLP context. Further, we have gone over some of the most recent research in question answering. Finally, we introduce the datasets used in this thesis. In the next chapter we will look at our first question answering model.

Chapter 3

A Deep Learning Primer

This chapter provides yet more background information.

3.1 Views of Language

3.1.1 A Compositional View of Language

It is generally agreed that language is compositional and follows a hierarchy ranging from words and characters to clauses, sentences, and entire documents - the understanding any of which might even be dependent upon knowledge of some external context. Linguists and NLP researchers have long explicitly modelled the compositional structure of language. Perhaps the two best examples of this are constituency and dependency parses of sentences (Figure 3.1 and Figure 3.2, respectively).

Parses of language model it as being made up of *terminals* (words) and *non-terminals* (in the case of Figure 3.1 Penn Treebank tags) which can operate on terminals or on other non-terminals themselves. Work on recursive neural nets which function across such parse trees (as popularized [129])has been a promising recent avenue of research but has been hampered by the difficulty of efficiently training such neural architectures.

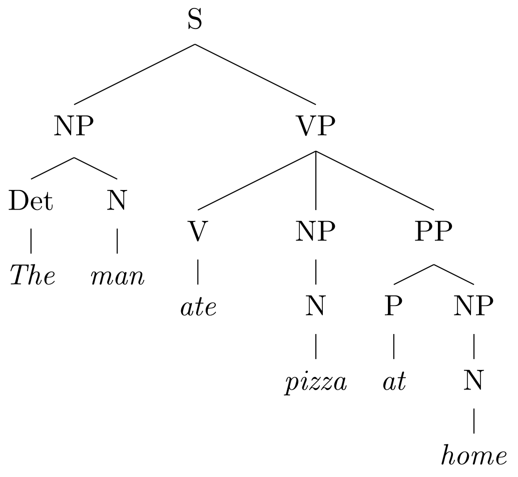

Figure 3.1: An example of a constituency parse tree.

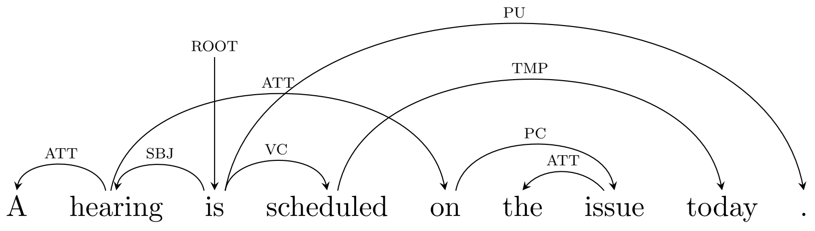

Figure 3.2: An example of a dependency parse tree.

3.1.2 A Sequential View of Language

Another way of looking at language is just as a sequence of words. The task is then to model language using models capable of reasoning across sequences. This view on language traces its history back to early sequential models such as hidden Markov models and conditional random fields. In deep learning, the most popular models have become recurrent neural networks starting with vanilla RNNs, followed by LSTMs, GRUs, and other recurrent neural network variants.

The sequential modelling of language does not fully ignore the compositional view, but instead, models it implicitly and argues that its explicit modelling is unnecessary. This thesis follows the sequential view of language.

3.2 Recurrent Neural Networks

Recurrent neural network architectures are a family of neural architectures capable of handling sequntial input. Their names originate from the recurrent application of the same set of parameters at each time-step and their dependence at each time-step on not only the present input but also the networks previous state.

3.2.1 Vanilla RNNs

The simplest form of recurrent neural network is the vanilla RNN shown here:

$$\begin{aligned} h_t &= \sigma(W_h h_{t-1} + W_i x_t + b_h) \\ o_t &= \sigma(W_o h_t + b_o) \end{aligned} \tag{3.1}$$

Where W_i, W_h and W_o are the input, recurrence, and output parameters; b_h and b_o the bias terms; h_t the hidden states (or network states); and, x_t and o_t the inputs and outputs. Notice that one can *unroll* the above across time-steps and think of it as a deep feed-forward neural network where inputs are fed to it at every layer.

The problem with the above vanilla RNN model is that it comes with a set of deficiencies which make it hard to train. Above all else, the above model is unstable because during optimization, its gradients may explore or vanish. It is easy to intuitively see why that is. Thinking of Equation 3.1 in its unrolled form and taking its derivative with respect to any of its parameters it is easy to see how one must borough through the function using the chain rule. This leads to a product of Jacobians as the derivative passes through the hidden layers. Just as the product of numbers just smaller than 1 collapse to 0 and just above 1 explode to infinity, so too do the gradients of vanilla RNNs behave. There are many hacks which to varying degrees solve this problem, such as clipping the gradients if they pass a certain threshold, but there has been much research since on architectures which do not experience this problem.

3.2.2 LSTMs

Motivated by the exploding and vanishing gradients problems of vanilla RNNs, *long-short term memory* networks (LSTMs) were invented as a result.

$$\begin{aligned} f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t &= \sigma_g\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ o_t &= \sigma_g\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c\left(W_c x_t + U_c h_{t-1} + b_c\right) \\ h_t &= o_t \circ \sigma_h\left(c_t\right) \end{aligned} \tag{3.2}$$

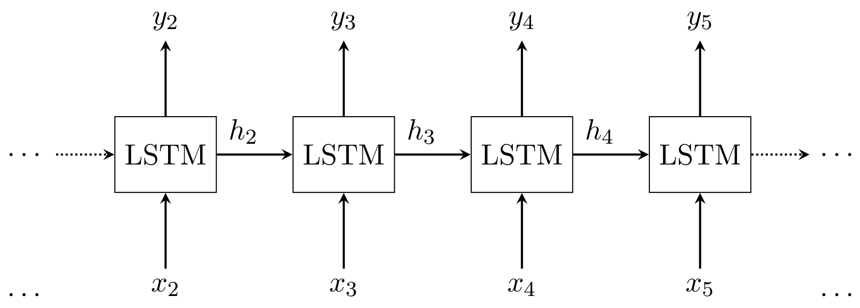

Figure 3.3: An LSTM network

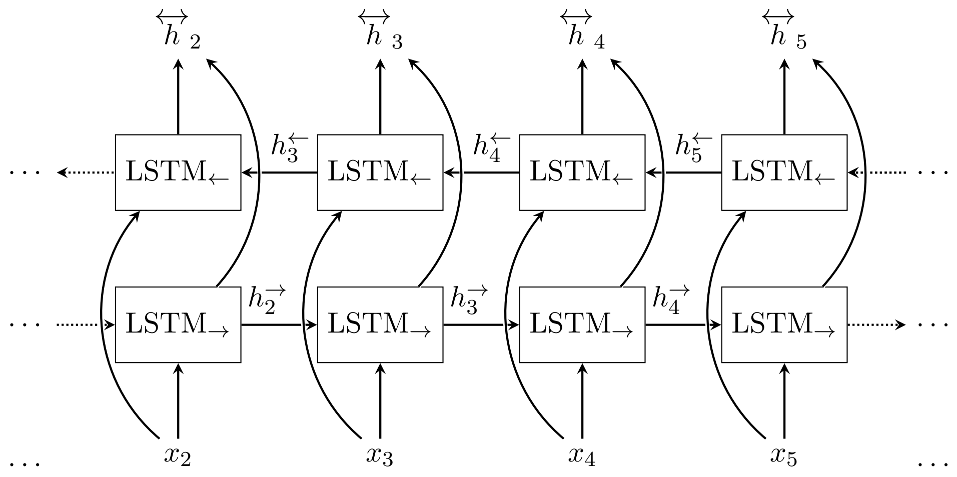

Figure 3.4: A bi-directional LSTM network

An LSTM is a gated recurrent neural network architecture which is designed to prevent the gradient from vanishing. Looking at Equation 3.2 we see the LSTM has three so-called *gates*: the forget gate f_t, input gate i_t, output gate o_t are all functions of the networks previous hidden state and the current input. These gates modulate the input flowing into $i_t \circ \sigma_c (W_c x_t + U_c h_{t-1} + b_c)$, the output flowing out of $o_t \circ \sigma_h (c_t)$ and the previous state flowing into $f_t \circ c_{t-1}$ the cell state c_t. By gating the cell state, vanishing derivatives are avoided. An intuitive way of seeing why this is by noticing that to take derivatives with respect to parameters will involve taking derivatives through cell states c_t across time. Since the derivative of $\partial c_t / \partial c_{t-1}$ is f_t, the network can learn to keep $f_t = 1$ when it wants the gradient to flow through it unimpeded.

3.2.3 Bi-directional RNNs

Bi-directional RNNs (incluidng bi-directional LSTMs as seen in Figure 3.4) are a special form of RNN where the data is modelled in both directions. This allows bi-directional

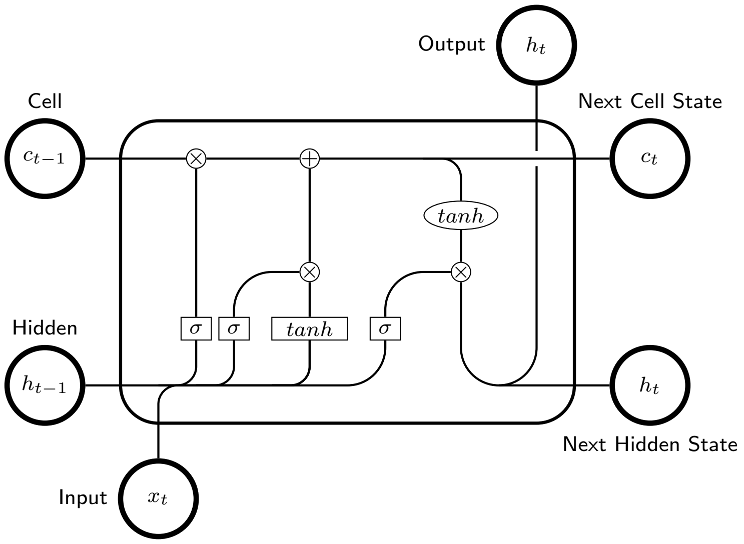

Figure 3.5: The inside of an LSTM cell

models to capture features of the input a normal RNN wouldn't have. In practice, creating a bi-directional RNN is simple. Two normal RNNs are initialized, data is then fed in correct order through one of them, and in reverse order through the other. The outputs are then concatenated.

3.2.4 Other architectures

There are many other RNN architectures in addition to vanilla RNNs and LSTMs. Notably, *gated recurrent units* or GRUs [20] have become popular as a more efficient and simpler alternative to LSTMs. Nevertheless, vanilla LSTMs have become standard in NLP.

3.3 Attention Mechanisms

In RNNs, information from previous inputs is compressed within the current most network state. Attention mechanisms instead allow to *attend* to the entire sequence and model it as a whole.

In general, an attention function computes a weighted sum over some values V, where the weights are a normalized *compatibility* function of two values: a query Q and a key K.

$$\text{Attention}\left(Q, K, V\right) = \text{softmax}\left(QK^T\right) V \tag{3.3}$$

This general form of attention allows us to introduce a few specific forms of it:

3.3.1 Self-Attention

In sel-attention the keys, values, and queries are all the same giving us:

$$\text{Attention}\ (V) = \text{softmax}\left(VV^T\right) V \tag{3.4}$$

3.3.2 Parametarized Attention

Parameterized attention parametarizes the compatibility function with some function f:

$$\text{Attention}\ (Q, K, V) = \text{softmax}\left(f(QK^T)\right) V \tag{3.5}$$

3.3.3 Multi-head Attention

Multi-head attention takes multiple *heads* of parametarized attention (Equation 3.5) on the same input and concatenates the outputs of the *heads*.

3.4 Word and Sentence Representations

The question of how text should be represented as input to models is a longstanding one. In the simplest case, a sequence of words could be represented using a sequence of one-hot vectors. Concretely, imagine a vocbulary of size three containing the words: *the*, *ate*, and *dog*. Then we may form vectors of size three to represent the three words as:

$$\begin{aligned} \text{the} &= (1,0,0) \\ \text{ate} &= (0,1,0) \\ \text{dog} &= (0,0,1) \end{aligned} \tag{3.6}$$

The sentence *the dog ate* can then be represented as a sequence of these one-hot embeddings:

$$[(1,0,0),(0,0,1),(0,1,0)] \tag{3.7}$$

Or as an input matrix of stacked one-hot column vectors:

$$X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.8}$$

Adding up the one-hot vectors of a sentence's words gives us its *bag of words* (BoW), also called *term frequency* (TF), representation.

The problem with such simple embeddings is that they fail to capture any form of similarity or relatedness between words. One-hot embeddings at the very least still preserve order whereas a TF representation loses information even about that.

3.4.1 Distributed Word Embeddings

One way of elucidating a word's meaning is to look at the company it keeps. This so-called *distributional* view of semantics is what inspired neural word embedding models such as word2vec [89]. These work by encoding compressed word co-occurrence counts. It is perhaps best to understand word2vec through example. The word *book* often co-occurs with words such as *flight*, *library*, *buy*, *room*, *hotel*, and *cover*.

To learn a good embedding for the word *book* using the *skip-gram* model we sample *word-context* pairs from within some window (in our case *book-flight, book-library, book-room*, and so on) and try and predict the context word from the input word for all word-context pairs. To do this for the example pair *book-flight*, we take the vocabulary-sized one-hot encoding of *book*, multiply it by the input embedding parameter matrix W_{in} to get its compressed representation, then decode it back using the output matrix W_{out} into a vocabulary-sized vector. We then softmax across this vector and expect the output to approach the one-hot embeddings for the word *flight*. After training is complete, W_in becomes our embedding matrix with every row being a different word embedding (corresponding to the index of the 1 in the one-hot embeddings of our words).

The CBOW model learns word embeddings in the opposite way. It takes *context-word* pairs, converts the *contexts* into a bag-of-words representation, and then tries to predict the correct *word*.

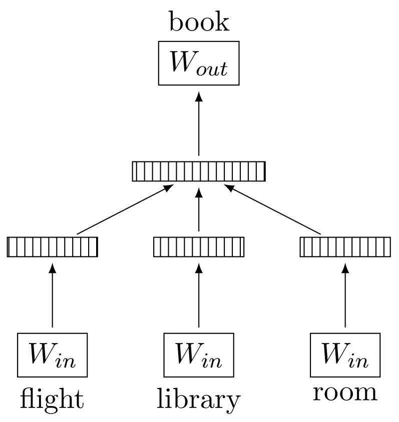

Figure 3.6: CBOW model

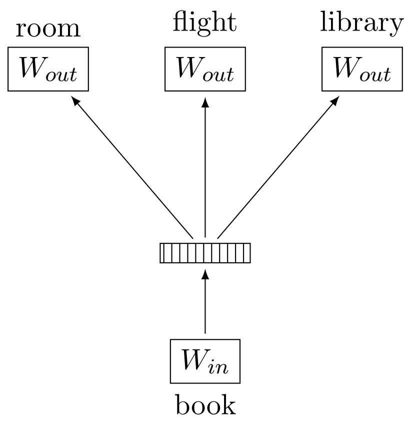

Figure 3.7: Skip-gram model

Models such word2vec [89] (described above), GloVe [99], and more recently fast-Text [60], all attempt to learn such compressed word embeddings. However, notice that that (1) the sampled *word-context* pairs all come from a fixed-sized window, and (2) context words are modelled independently and at no point are contexts considered as a sequence.

3.4.2 Sentence and Document Embeddings

Recently, word and sentence-level embeddings which capture information from the context in its entirety have begun to appear. Notably, a string of publications in 2018 [102, 54, 103] were the first to introduce such models and show that their representations are transfer well to downstream tasks. We describe one of these, ELMo [102], because we use it in Chapter 4 as part of our ablation study of BiDAF.

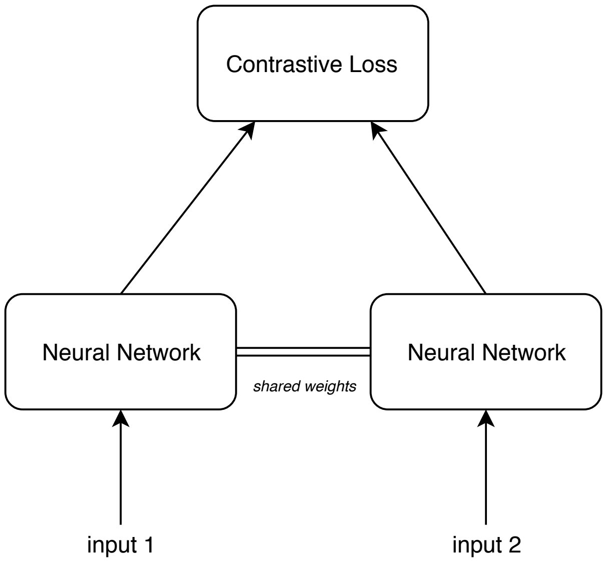

Figure 3.8: An example of a Siamese Network

3.4.3 ELMo

Embeddings from language models (ELMo) is a recently released models which uses a bi-directional language model to learn word embeddings. A language model is a model which models the probability of a sentence by continiously trying to predict the next word given the already seen ones, i.e.:

$$P\left(w_1, \ldots, w_m\right) = \prod_{i=1}^{m} P\left(w_i | w_1, \ldots, w_{i-1}\right) \tag{3.9}$$

One way of modelling the above is by using an LSTM to, at every time-step, predict the next word. ELMo does just that but instead of using a single LSTM it uses multiple stacked bi-directional LSTMs to model at every timestep model the next word given past *and* future words. It has been shown [102, 100] that ELMo seems to improve results in many NLP tasks across the board.

3.5 Siamese Networks

Siamese networks are networks which follow a specific kind of architecture. They consist of two identical networks with shared weights which are used to embed usually two inputs. The embeddings of these inputs and then compared in the objective function using a contrastive loss. The goal is not to classify, but rather to differentiate between inputs. For example, in Chapter 6 we look into training siamese-like networks for answer selection where we train the model to differentiate between correct context-candidate pairs and wrong (distractor) context-candidae pairs.

3.6 Summary

The above brief introduction to some of the most important deep learning and NLP concepts used throughout this thesis will help us as we present new models in subsequent chapters. Throughout, I will refer the reader back to this chapter whenever we use, with little description, any of the techniques described above. Already in the next chapter, we will make heavy use of almost all that has been introduced here.

Part II

Question Answering

Chapter 4

A Detailed Ablation Study of the BiDAF Model

This chapter contains our first contribution.

4.1 Introduction

Since the release of the Stanford Question Answering Dataset (SQuAD) [105] in 2016, dozens of models have competed on it. Although all different, they are comprised of many of the same or similar components. Just as LSTMs have become a staple component of many NLP models, so too has the *bi-directional attention flow* (BiDAF) [123] model become a popular model within reading comprehension research. Subsequent publications have all either analyzed it, based their proposed models on it, or compared themselves against it. A review of these can be found in Section 4.7. Like other question answering models, BiDAF divides into three sections. The input layers, the body layers, and the output (or answer) layers. A general description of these can found in Section 2.2.2. We describe how exactly BiDAF implements these layers in Section 4.2.

In this chapter we are interest in exploring which sections of neural QA models matter most to their performance. We take BiDAF as a canonical example of a neural QA model and SQuAD as a canonical dataset and (1) explore BiDAF's performance by conducting a detailed ablation study of its components. In so doing we attempt to isolate parts of it which contribute most to end performance; (2) We propose an extension of the BiDAF model to multiple spans.

We begin by describing the model in detail, followed by describing our modifications and ablations of it. Our changes touch on four sections of the model: (1) we analyze different attention mechanisms in the body layer used to fuse information about the passage and query; (2) we analyze different fusion functions used to do this; (3) we look at how important the question representation is to the success of the model in the input layer; finally (4) we look at the new multi-span setting.

4.2 The BiDAF Model

4.2.1 The Input Layers

The goal of the input layers is to vectorize the question and the passage (also referred to as the context) and represent them in such a way so as to lighten the modeling burden of downstream layers. Put simply, the better the representations produced by the input layers, the easier it is for the body and output layers to make sense of the question and passage.

All this is achieved by using fusing word- and character-level representations and passing them through recurrent models to 'contextualize' so as to be able to model inter-word dependencies.

To form contextualized embeddings of passage words $\boldsymbol{h}_i$ and contextualized embeddings of question words $\boldsymbol{u}_j$ we model them at different granularities:

- **Character-level.** Every word is made up of individual characters. Characters can tell us a lot about a word. For example, by just looking at a word's characters we may quickly get a sense of whether it denotes a name or place, or whether it is an acronym. Thus, encoding character-level features has become standard in modern NLP models. BiDAF opts for encoding characters using a *convolutions neural network* (CNN) as done in [62]. Denote by $\mathbf{x_{i:i+h}}$ a window of characters (of some single word). A convolution operation takes a *filter* parameter $w \in \mathbb{R}^h$ and applies it to a window of h characters:

$$c_i = f\left(\boldsymbol{w} \cdot \boldsymbol{x}_{i:i+h-1} + b\right) \tag{4.1}$$

  Where b is a bias term and f a non-linear function. The filter is applied to a rolling window of character representations (taking pad values if boundaries are crossed) to produce a *feature map*:

$$\boldsymbol{c} = [c_1, c_2, \ldots, c_{n-h+1}] \tag{4.2}$$

  where $\boldsymbol{c} \in \mathbb{R}^{n-h+1}$. The feature-map is then max-pooled to produce the final feature $\hat{c} = \{\boldsymbol{c}\}$. By applying many such *filters* to character representations, a character *vector* representation can be come up with.

- **Word-level.** Words are represented as concatenations between their pre-trained GloVe word embeddings [99] and their character vector representations. These are then passed through a two-layer highway network [131] to produce the final word-level embeddings.

- **Context-level.** The word-level embeddings from the previous layer are passed through a bi-directional LSTM [52] to produce contextualized passage and question word embeddings $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$, respectively. Packed together in matrix form we denote these by $\boldsymbol{H} \in \mathbb{R}^{2d \times T}$ and $\boldsymbol{U} \in \mathbb{R}^{2d \times J}$. where d is the dimension of the word embeddings from the previous layer and $2d$ is the size of the contextualized embeddings (their dimension is $2d$ instead of d because of the bi-direcitonal LSTM). T and J denote the length of the context and question, in number of words, respectively.

The above sequence of input layers have become popular in many different NLP models. In short, the character layers encode character-level features such as capitalization and they are especially useful when a corresponding pre-trained word embedding does not exist. The word-level layer appends to the character-level features a pre-trained word embedding which captures information learned about the word from large **external** unsupervised text corpus by learning about the contexts in which the word appears and which words it co-occurs with. Finally, the context-level layer contextualizes these disparate representations to make them co-dependent on each other by passing them through an LSTM.

Very recently, the word-level and context-level layers are being replaced by pre-trained language models [102] in what is widely expected to be NLP's ImageNet moment. Unfortunately, these are beyond the scope of this thesis.

4.2.2 The Body Layers

The body is what usually differentiates many NLP models. In cases where we are modelling two inputs (a passage of a text and a corresponding question) the obvious thing we want to do is to model the dependencies between these inputs. BiDAF proposes a particular flavour of doing this.

- **Attention Flow.** What makes BiDAF's body different from many other models is that it avoids aggregating the result of attention into vectors prematurely thus reducing the loss incurred by early summarization. The inputs to the layer are the context-level embbedings of the context $\boldsymbol{H}$ and the query $\boldsymbol{U}$ from the previous layer. The layer's outputs are the query-aware vector representations of the context words, $\boldsymbol{G}$, and the contextual embeddings from the previous layer which are passed through.

We now describe how the query-aware context word representations are formed:

1. We first create a similarity matrix $\boldsymbol{S} \in \mathbb{R}^{T \times J}$ where every component is a function of a context and question word:

   $$\boldsymbol{S}_{tj} = \alpha\left(\boldsymbol{H}_{:t}, \boldsymbol{U}_{:j}\right) \in \mathbb{R} \tag{4.3}$$

   In the original model α is set to be:

   $$\alpha(\boldsymbol{h}, \boldsymbol{u}) = \boldsymbol{w}_{(\boldsymbol{S})}^{\top}[\boldsymbol{h}; \boldsymbol{u}; \boldsymbol{h} \circ \boldsymbol{u}] \tag{4.4}$$

2. Next, we apply attention to the similarity matrix $\boldsymbol{S}$. Two attentions are applied:

   Context to Query. The focus here is on the context words. For each context word, the model attends across all question words to see how relevant they are to the context word. First, an attention distribution $\boldsymbol{a}_t$ is computed by:

   $$\boldsymbol{a}_t = \text{softmax}\left(\boldsymbol{S}_{t:}\right) \in \mathbb{R}^J \tag{4.5}$$

   Next, for every t^{th} word in the context, the corresponding attention weights $\boldsymbol{a}_t$ are used to attend across the question:

   $$\tilde{\boldsymbol{U}}_{:t} = \sum_j \boldsymbol{a}_{tj}\boldsymbol{U}_{:j} \tag{4.6}$$

   where $\tilde{\boldsymbol{U}} \in \mathbb{R}^{2d \times T}$.

   Query to Context. The second attention mechanism we apply flow the opposite way. The focus here is on the query words. For each query word, the model now attends across all context words to see how relevant they are to the query word. We begin by forming the attention weights:

   $$\boldsymbol{b} = \text{softmax}\left(\max_{col}(\boldsymbol{S})\right) \in \mathbb{R}^T \tag{4.7}$$

   where the *max* is taken column-wise, i.e. for every row in $\boldsymbol{S} \in \mathbb{R}^{T \times J}$, we pick its maximum value. Next we apply the attention weights to every word in the context:

$$\tilde{\boldsymbol{h}} = \sum_t \boldsymbol{b}_t \boldsymbol{H}_{:t} \in \mathbb{R}^{2d} \tag{4.8}$$

We now copy this vector T times and tile it to form $\tilde{\boldsymbol{H}} \in \mathbb{R}^{2d \times T}$.

A natural question one might have is why *context-to-query* attention is so different from *query-to-context* attention. Ultimately, it boils down to a design choice. Were the same attention mechanism as seen in *context-to-query* to have been used in *query-to-context* it would result in a $\tilde{\boldsymbol{H}} \in \mathbb{R}^{2d \times J}$. This would lead to a miss-match in dimensions between the new $\tilde{\boldsymbol{H}}$ and $\tilde{\boldsymbol{U}} \in \mathbb{R}^{2d \times T}$.

3. **Combining Attention Flow Results**

   Given the results from the previous two sections we combine them into a single matrix:

$$\boldsymbol{G}_{:t} = \boldsymbol{\beta}\left(\boldsymbol{H}_{:t}, \tilde{\boldsymbol{U}}_{:t}, \tilde{\boldsymbol{H}}_{:t}\right) \in \mathbb{R}^{d_{\boldsymbol{G}}} \tag{4.9}$$

   where $\boldsymbol{G} \in \mathbb{R}^{d_G \times T}$, β is some fusion function, and d_G its output dimension. In the original paper, the fusion function is set to be:

$$\boldsymbol{\beta}(\boldsymbol{h}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{h}}) = [\boldsymbol{h}; \tilde{\boldsymbol{u}}; \boldsymbol{h} \circ \tilde{\boldsymbol{u}}; \boldsymbol{h} \circ \tilde{\boldsymbol{h}}] \in \mathbb{R}^{8d}. \tag{4.10}$$

   The matrix $\boldsymbol{G}$ now contains T query-aware context embeddings. These are now further contextualized by passing them through a bi-directional LSTM of total size $2d$ giving us a matrix $\boldsymbol{M} \in \mathbb{R}^{2d \times T}$.

4.2.3 The Answer (Output) Layers

- **Finding answer spans.** To find the correct answer span, the model learns to point to its correct start and end index. This involves calculating two probability distributions across all context words. The start index distribution is

$$\boldsymbol{p}^1 = \text{softmax}\left(\boldsymbol{w}_{(\boldsymbol{p}^1)}^\top [\boldsymbol{G}; \boldsymbol{M}]\right). \tag{4.11}$$

  And the end index distribution is

$$\boldsymbol{p}^2 = \text{softmax}\left(\boldsymbol{w}_{(\boldsymbol{p}^2)}^\top \left[\boldsymbol{G}; \boldsymbol{M}^2\right]\right). \tag{4.12}$$

  Where the two $\boldsymbol{w}$ are trainable parameters and $\boldsymbol{M}^2$ is the result of $\boldsymbol{M}$ being passed through another bi-directional LSTM. The concatenation $\boldsymbol{G}; \boldsymbol{M}$ is done across the token, T, dimension so that $[\boldsymbol{G}; \boldsymbol{M}] \in \mathbb{R}^{(8d+2d)\times T}$.

4.2.4 Model Optimization

During training we minimize the following objective function:

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N} \log\left(\boldsymbol{p}^1_{y_i^1}\right) + \log\left(\boldsymbol{p}^2_{y_i^2}\right) \tag{4.13}$$

where θ denotes all the trainable parameters used in the model. To recap, these include:

- The CNN filter parameters used for character embeddings.
- The various LSTM parameters. BiDAF uses 4 LSTMs (two input LSTMs, a fusion LSTM, and an LSTM to differentiate the input for predicting $\boldsymbol{p}_2$).
- Output layers parameters used for computing $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$.

Importantly, notice that no parameters are used in the attention mechanism.

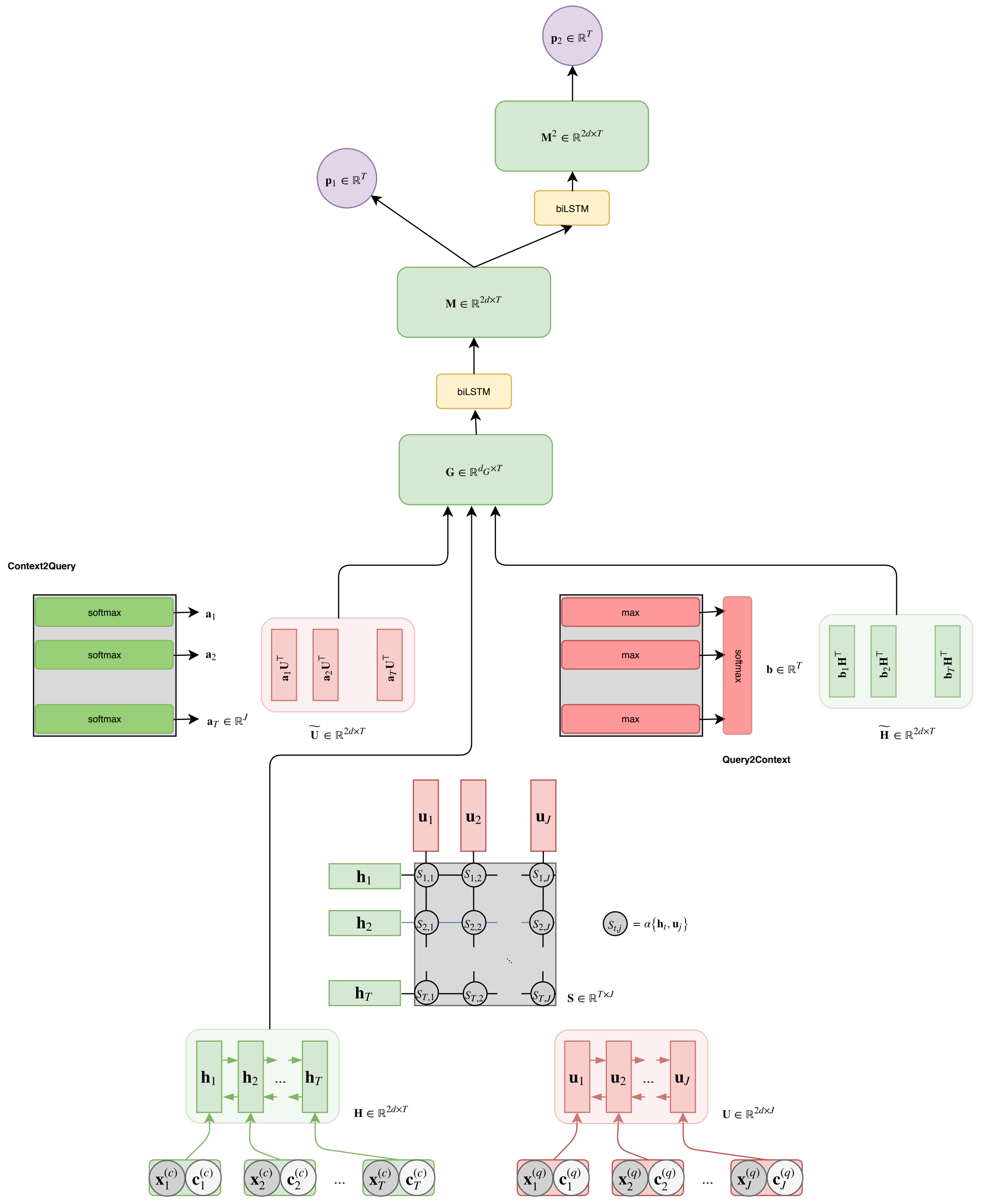
$\mathbf{p}_2 \in \mathbb{R}^T$
$\mathbf{M}^2 \in \mathbb{R}^{2d \times T}$
$\mathbf{p}_1 \in \mathbb{R}^T$
biLSTM
$\mathbf{M} \in \mathbb{R}^{2d \times T}$
biLSTM
$\mathbf{G} \in \mathbb{R}^{d_G \times T}$
Context2Query
softmax
softmax
softmax
$\mathbf{a}_1$
$\mathbf{a}_2$
$\mathbf{a}_T \in \mathbb{R}^J$
$\mathbf{a}_1\mathbf{U}^\top$
$\mathbf{a}_2\mathbf{U}^\top$
$\mathbf{a}_T\mathbf{U}^\top$
$\widetilde{\mathbf{U}} \in \mathbb{R}^{2d \times T}$
max
max
max
softmax
$\mathbf{b} \in \mathbb{R}^T$
$\mathbf{b}_1\mathbf{H}^\top$
$\mathbf{b}_2\mathbf{H}^\top$
$\mathbf{b}_T\mathbf{H}^\top$
$\widetilde{\mathbf{H}} \in \mathbb{R}^{2d \times T}$
Query2Context
$\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_J$
$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_T$
$S_{1,1}$ $S_{1,2}$ $S_{1,J}$
$S_{2,1}$ $S_{2,2}$ $S_{2,J}$
$S_{T,1}$ $S_{T,2}$ $S_{T,J}$
$S_{t,j} = \alpha\{\mathbf{h}_t, \mathbf{u}_j\}$
$\mathbf{S} \in \mathbb{R}^{T \times J}$
$\mathbf{h}_1$ $\mathbf{h}_2$ ... $\mathbf{h}_T$
$\mathbf{H} \in \mathbb{R}^{2d \times T}$
$\mathbf{u}_1$ $\mathbf{u}_2$ ... $\mathbf{u}_J$
$\mathbf{U} \in \mathbb{R}^{2d \times J}$
$\mathbf{x}_1^{(c)}$ $\mathbf{c}_1^{(c)}$ $\mathbf{x}_2^{(c)}$ $\mathbf{c}_2^{(c)}$ ... $\mathbf{x}_T^{(c)}$ $\mathbf{c}_T^{(c)}$
$\mathbf{x}_1^{(q)}$ $\mathbf{c}_1^{(q)}$ $\mathbf{x}_2^{(q)}$ $\mathbf{c}_2^{(q)}$ ... $\mathbf{x}_J^{(q)}$ $\mathbf{c}_J^{(q)}$

Figure 4.1: The BiDAF model.

)

4.3 Ablations

In this section we first go over a worked example of BiDAF's attention mechanism. We then introduce *Uncompressed query-to-context attention* in Section 4.3.2 and propose four of its variants. In Section 4.3.3 we propose a simpler question representation input. Finally, in Section 4.3.4 we propose four new fusion functions.

4.3.1 Understanding Context2Query and Query2Context

Most machine reading comprehension models hinge on using *context2query* and *query2context* attention. We first go over a worked example to gain an intuition over how these two attention mechanisms work.

Consider the context passage:

Nikola Tesla was born in Smiljan.

And consider the question:

Where was Nikola Tesla born?

Assume the similarity matrix for this context-question pair to be $\boldsymbol{S} \in \mathbb{R}^{T \times J}$ (where $T = 6$ and $J = 5$):

	Where	**was**	**Nikola**	**Tesla**	**born**
Nikola	1	1	5	4	1
Tesla	1	1	4	5	2
was	3	5	1	1	2
born	2	1	2	2	5
in	3	1	1	1	4
Smiljan	4	1	4	4	2

Context2Query. In *context2query* we attend across all question words *for each context words* where the attention weights are a function of similarity score between the current context word and *all* question words.

To get the attention weights we take $\boldsymbol{S}$ and apply a softmax across it column-wise giving us the attention matrix $\boldsymbol{A} = softmax_{col}(S)$:

	Where	was	Nikola	Tesla	born
$\boldsymbol{a}_1$ **Nikola**	0.01	0.01	0.7	0.26	0.01
$\boldsymbol{a}_2$ **Tesla**	0.01	0.01	0.25	0.69	0.03
$\boldsymbol{a}_3$ **was**	0.11	0.82	0.01	0.01	0.04
$\boldsymbol{a}_4$ **born**	0.04	0.02	0.04	0.04	0.86
$\boldsymbol{a}_5$ **in**	0.24	0.03	0.03	0.03	0.66
$\boldsymbol{a}_6$ **Smiljan**	0.31	0.02	0.31	0.31	0.04

We now use the attention weights $\boldsymbol{a}_t$ to attend across the the question embeddings $\boldsymbol{U} \in \mathbb{R}^{2d \times J}$. If we consider our attention weight vectors in matrix form $\boldsymbol{A}^{(c2q)} \in \mathbb{R}^{T \times J}$ then the application of our attention weights (as in Equation 4.5) is:

$$\tilde{\boldsymbol{U}} = softmax(\boldsymbol{S})\boldsymbol{U}^\top = \boldsymbol{A}^{(c2q)}\boldsymbol{U}^\top \tag{4.14}$$

Query2Context. The way *query2context* is computed is different. We first take a column-wise max, i.e. $max_{col}(\boldsymbol{S})$ giving us:

	Question Max
Nikola	5
Tesla	5
was	5
born	5
in	4
Smiljan	4

Taking the softmax of this we are left with:

$$\boldsymbol{b} = softmax(max_{col}(\boldsymbol{S})) = \begin{pmatrix} 0.21 \\ 0.21 \\ 0.21 \\ 0.08 \\ 0.08 \end{pmatrix} \tag{4.15}$$

Which we use to attend across question words (just as in Equation 4.8):

$$\tilde{\boldsymbol{h}} = \boldsymbol{b}^\top \boldsymbol{H}^\top \tag{4.16}$$

We now tile $\tilde{\boldsymbol{h}}$, T times to get, $\tilde{\boldsymbol{H}} \in \mathbb{R}^{2d \times T}$.

Notice how *context2query* and *query2context* attentions are different. The latter compresses information contained in the similarity matrix $\boldsymbol{S}$ by taking a column-wise max across it whereas the former doesn't. In the next subsection we propose forgoing this compression.

4.3.2 Uncompressed Query2Context Attention

Instead of the compressed attention proposed in the original BiDAF (see Equation 4.7), we study what happens when uncompressed attention (see Equation 4.5) is applied during *query2context*. Equation 4.7 collapses the similarity matrix using a column-wise *max* function. This throws away any information contained in most components of $\mathbf{S}$. Our motivation here is to study whether by forgoing this compression, we may achieve better overall results.

Just as in Equation 4.7 but in reverse we attend across all context embeddings for every question word:

$$\boldsymbol{V} = softmax(\boldsymbol{S}^\top)\boldsymbol{H}^\top \tag{4.17}$$

Since $\boldsymbol{V} \in \mathbb{R}^{J \times 2d}$, we have to find a way of merging it into what later becomes $\boldsymbol{G}$ (see Equation 4.9). To achieve this we need it to be the same dimension as $\tilde{\mathbf{H}} \in \mathbb{R}^{2d \times T}$. We study four variants of achieving this:

Variant 1

Our first attempt is to re-apply generalized attention to the new $\boldsymbol{V}$ in much the same way as in *context2query*. Notice that under this variation, $\tilde{\mathbf{H}}$ is a function of both *context-to-query* attention and uncompressed *query-to-context* attention.

$$\tilde{\boldsymbol{H}} = (softmax(\boldsymbol{S})\boldsymbol{V})^\top = (\boldsymbol{A}^{(c2q)}\boldsymbol{V})^\top \tag{4.18}$$

Variant 2

Our second attempt is to map $\boldsymbol{V}$ into a shape resembling $\tilde{\boldsymbol{H}}$. This is done by passing $\boldsymbol{V}$ through a feed-forward neural network. Under this variation, $\tilde{\mathbf{H}}$ is a function of uncompressed *query-to-context* attention and a non-linear mapping which maps it into the correct form.

$$\tilde{\boldsymbol{H}} = FFNN(\boldsymbol{V}) \tag{4.19}$$

Variant 3

Our third attempt is to compress $\boldsymbol{V}$ into a single vector using a feed-forward neural network and then tile it. In a way this is similar to the original compressed attention. The key distinction is that all similarities in $\mathbf{S}$ are retained and it is up to the feed-forward neural network to correctly compress them into a single vector which is than tiled rather than a *max* function as in Equation 4.7.

Variant 4

The fourth variant on the original architecture is to take $\boldsymbol{S}$ and apply both column-wise and row-wise softmax followed by applying the original *query2context* attention to it. Under this variation $\boldsymbol{V}$ isn't used but a similar effect is reached since the softmax is taken in the same way as in Equation 4.17.

4.3.3 Using Different Question Representations

Variant 5

There has been continued criticism of existing SQuAD models failing to generalize well (see related work section). We set the question representations to be independent word embeddings which *are not* passed through an LSTM to contextualize them.

4.3.4 Different Fusion Functions

We study four different fusion functions which replace Equation 4.10.

Variant 6

Under this variation we only bring forward the result of the *query-to-context* attention branch.

$$\boldsymbol{\beta}(\boldsymbol{h}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{h}}) = [\tilde{\boldsymbol{h}}] \in \mathbb{R}^{2d \times T} \tag{4.20}$$

Variant 7

Under this variation we only bring forward the result of the *context-to-query* attention branch.

$$\boldsymbol{\beta}(\boldsymbol{h}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{h}}) = [\tilde{\boldsymbol{u}}] \in \mathbb{R}^{4d \times T} \tag{4.21}$$

Variant 8

Under this variation we only bring forward the results of the two attention branches and *not* the original input $\mathbf{h}$.

$$\boldsymbol{\beta}(\boldsymbol{h}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{h}}) = [\tilde{\boldsymbol{h}}; \tilde{\boldsymbol{u}}] \in \mathbb{R}^{4d \times T} \tag{4.22}$$

Variant 9

Under this variation we bring forward everything present in Equation 4.10 apart from $\boldsymbol{h} \circ \tilde{\boldsymbol{h}}$ - i.e. the interaction between the input and the result of *query-to-context*.

$$\boldsymbol{\beta}(\boldsymbol{h}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{h}}) = [\boldsymbol{h}; \tilde{\boldsymbol{u}}; \boldsymbol{h} \circ \tilde{\boldsymbol{u}}] \in \mathbb{R}^{8d \times T} \tag{4.23}$$

4.4 Multi-Span Answers

This section proposes an extension of the BiDAF architecture to the multi-span setting where an answer can belong to multiple spans. We wish to classify each span j in a passage of text D as being correct or not given a question Q. We denote by T_D and T_Q the number of words in the passage of text and question, respectively. The total number of spans in a passage of text is $M = \frac{T_D(T_D+1)}{2}$.

We denote the start and end word indexes of a span s_j by START(j) and END(j). We assume an ordering of spans based on START(j); spans with the same start index are ordered by END(j).

As the goal is to figure out the correct set of spans given an input we can think of this as a multilabel classification problem.

For further notation, assume we have a set of inputs $\mathbf{x_i} \in \mathbf{X}$ (where every input is comprised of D and Q) and that we denote our neural network model by $f(\cdot)$. The output of $f(\cdot)$ is a scoring function that produces a vector of activations.

4.4.1 Multilabel Loss

The loss layer specifies how network training penalizes the deviation between the predicted and true answer spans. There has been a lot of study into different multilabel loss layer types, particularly in the computer vision community. A nice review can be found in [36]. In this thesis, we opt for the multilabel softmax loss.

The posterior probability of input $\mathbf{x}_i$ and span j can be expressed as

$$p_{ij} = \frac{\exp(f_j(\mathbf{x}_i))}{\sum_{k=1}^{M} \exp(f_k(\mathbf{x}_i))} \tag{4.24}$$

where $f_j(\mathbf{x}_i)$ is the output activation of the model for input $\mathbf{x}_i$ and span j. We can now minimize the KL-divergence between the predictions and the ground-truth probabilities. Since each input can have multiple correct spans (labels), we can form a label vector $\mathbf{y} \in \mathbf{R}^{1\times M}$ for each input where $y_j = 1$ denotes the presence of a label and $y_j = 0$ its absence. Ground truth probabilities can then be obtained by normalizing $\mathbf{y}$ as $\mathbf{y}/||\mathbf{y}||_1$. If the ground truth probability for input i and span j is defined as $\bar{p}_{ij}$, the cost function to be minimized is

$$J = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\bar{p}_{ij}\log(p_{ij}) \tag{4.25}$$

where N is the number of training examples.

4.4.2 BiDAF-MS

We propose a modified version of BiDAF which we call BiDAF-MS ('MS' for *multi span*). Using the contextual vectors $\{\mathbf{h}_i\}$ we form recursive span representations, just as in [72], by taking the $\{\mathbf{h}_l\}$ and forming span representations $\mathbf{s}_j = [\mathbf{h}_{\mathrm{START}(j)}; \mathbf{h}_{\mathrm{END}(j)}]$. The span representations are passed through a final feed-forward neural network which collapses them to a single value. These values are then fed to Equation 4.23.

4.4.3 Large Span Spaces

The total number of spans in a passage of text might seem exceedingly large. For example, the longest passage of text in SQuAD is made up of 300 tokens. That equates to 45,510 possible spans. Can this number be reduced? One possible way is to look at how this number behaves as we limit span lengths. The number of span lengths up to and including some length d can be expressed as:

$$g_{doc}(d) = \frac{T_D(T_D+1) - (T_D-d)(T_D-d+1)}{2} \tag{4.26}$$

where $d \leq T_D$. For span lengths up to and including ten, $g(10) = 2955$ which is a much more manageable sum. Further reductions in the total number of spans to be considered can be achieved by disregarding cross-sentence spans. In this case, the number of total spans can be calculated using:

$$g_{sent}(d) = \sum_{s}\frac{T_s(T_s+1) - (T_s-d)(T_s-d+1)}{2} \tag{4.27}$$

which reduces the total number of spans further.

4.4.4 A CoNLL QA dataset

We create a new dataset by repositing the CoNLL 2003 dataset as a multi-span reading comprehension problem where the questions are synthetically generated based on the entity type. For example, given a sentence such as:

London's Royal Opera House opens up its doors to toddlers with the aim of opening up opera to a younger generation.

London is tagged as a *LOCATION* and *Royal Opera House* as an *ORGANIZATION*. Out of the above we form two context-question pairs where the context is the above and questions (or rather *prompts*) are *Mark all people* and *Mark all organizations*, respectively. In so doing we end up with the below dataset:

	Training Set	Development Set	Test Set	Total
PER	634	159	937	1,730
ORG	748	160	215	1,123
LOC	937	181	228	1,346
Total	2,319	500	1,380	4,199

Table 4.1: CoNLL QA dataset statistics.

Note that wheras the original CoNLL dataset is usually considered as a collection tagged sentences (and the documentation demarcation boundaries are ignored), in our case, we consider it as a list of documents, each of which becomes a context passage in our dataset.

4.5 Results

The results sections covers the two main branches of this chapter: the ablation study with and the multi-span extension. We discuss them in turn.

4.5.1 Ablation Results

We run our model across the SQuAD dataset. The original BiDAF ablation study can be found in Table 4.2. Our ablation results with its variations can be found in Table 4.3. A quick glance at Table 4.3 shows that under certain variations we beat the original BiDAF model's F1 and EM scores. We go over a few key takeaways from the results next.

Ablation	EM	F1
No char embedding	65.0	75.4
No word embedding	55.5	66.8
No C2Q attention	57.2	67.7
No Q2C attention	63.6	73.7
Dynamic attention	63.5	73.6
Original BiDAF	67.7	77.3

Table 4.2: Existing BiDAF ablation results on the SQuAD dataset. [123] The **Exact Match** (EM) score measures the percentage of predictions that match any one of the ground truth answers exactly.

Ablation variant	EM	F1
v1	67.9	78.2
v2	59.2	76.3
v3	57.3	75.8
v4	66.5	76.5
v5	65.8	72.3
v6	55.1	65.3
v7	64.5	68.6
v8	65.2	69.9
v9	64.3	67.3
Original BiDAF	67.7	73.3

Table 4.3: Our BiDAF ablation results on the SQuAD dataset. [123]

Loss of Information In ***Query2Context*** Attention

By using an uncompressed form of *query2context* we are able to achieve results which beat the original model's score. This indicates that by column-wise maxing the similarity matrix we degrade performance by ignoring important information relating question words to context words. This can be seen by looking at v1-v4 in Table 4.3. Interestingly, our v1 proposal does best out of the four.

Performance Degradation With Simple Question Embeddings

Perhaps most interestingly, performance *did not* substantially degrade in *v*5 which points to the form of the question embedding not being that important. This further corroborates recent work which shows that even with complex question embeddings, models like BiDAF tend to model simple patterns between context and question instead of somehow modeling more general reading comprehension reasoning.

Importance of Various Embeddings

We note that other than using *query2context* embeddings alone (v6), results are remarkably stable as soon as *context2query* embeddings $\tilde{\boldsymbol{u}}$ are included which points to their importance.

What Does This Mean for Reading Comprehension Models?

Most recent reading comprehension models follow a structure similar to BiDAF's. Because of the evidence that the complexity of the question embeddings matters little and that *context2query* attention is more important than *query2context* attention we can hypothesize that most BiDAF-like models heavily rely on the *context2query* attention mechanisms across short questions which originate from a comparatively small vocabulary. This helps explain in part why existing reading comprehension models are so prone to making mistakes on adversarial examples.

4.5.2 Multi-Span Results

	F1
BiDAF-MS	85.1
BiDAF-MS (no Q2C)	85.3
BiDAF-MS (no Q2C + simple Q embedding)	85.3

Table 4.4: CoNLL QA test set results using the BIDAF-MS multi-span model.

We run the BiDAF-MS model on our new CoNNL QA dataset. The first thing discernible from the results is that, because the questions are synthetic, the model does not rely heavily on the question embedding branch. This can be seen by looking at how performance does not degrade when *query2context* is turned off, nor when the question is represented using just word embeddings and without contextualization using an LSTM. Despite the complexity of the output space, the model still performs competitively compared to the sequence-tagging models discussed in the next chapter which indicates it is capable of learning good span representations for NER.

4.6 Implementation Details

Individual experiments were run on NVIDIA P100 GPUs. Training time is around 7h for SQuAD and 1h for the CoNLL QA dataset. 100 dimensional Glove vectors are used as the input embedding representations, 16 dimensional character embeddings are used (the CNN which generates them uses 100 filters of size 5). All LSTMs in the various models are of size 100 (making the biLSTMs of size 200), the span encoder FFNNs are also of size 100. ADAM is used to optimize the models.

4.7 Related Work

The BiDAF model was introduced in November 2016, only six months after the release of SQuAD. Since then it has consistently been shown to perform well on various QA tasks. The original model presented in [123] was shown to achieve the then state-of-the-art in reading comprehension and on the cloze-style QA task on the CNN and Daily Mail corpus [46]. Since then it has become one of the most popular QA models.

Works exploring BiDAF's performance under various settings have recently been studied. Min et al. [90] trained a modified BiDAF model capable of sentence selection on SQuAD and then fine-tuned it on sentence selection tasks achieving state-of-the-art results on WikiQA and SemEval-2016 (task 3A).

After dozens of model submissions to the SQuAD leaderboard, more in-depth error analysis and criticisms of the SQuAD dataset appeared. What unified many of these works is their use of BiDAF model. Jia and Liang [58] showed that random ungrammatical or non-sensical sentence insertion into the SQuAD dataset can lead to a catastrophic collapse in BiDAF's performance, from 75.1 F1 down to 34.3 and 4.8 F1 respectively. More recently, [112] have shown that performance can be improved from 4.8 F1 to 52.3 F1 by first selecting the sentence most likely to contain the correct answer and only then running BiDAF.

Another recent work by Ribeiro et al. [113] studies the performance of BiDAF on paraphrases of development set questions. The paper introduces the *Semantically*

Equivalent Adversary (SEA), defined to be a paraphrase x' of input text x which leads to changed predictions $f(x) \neq f(x')$. Paraphrases are created by translating x into multiple pivot languages to and taking the score of their back-translations to be proportional to $P(x'|x)$. For these score to be consistent they are normalized into what the paper calls a semantic score $S(x, x')$. Semantic equivalence is then defined as $SemEq\,(x, x') = 1\,[S\,(x, x') \geq \tau]$.

The paper also introduces *Semantically Equivalent Adversarial Rules* (SEARs) which are rule-based system which generate SEAs. A rule is taken to be of the form $r = (a \rightarrow c)$ where the first instance of the antecedent a is replaced by the consequent c for every instance that includes a. For example, $r = (movie \rightarrow film)$ would lead to r("Great movie!") = "Great film!". More general rules can be formed such as *(What NOUN → Which NOUN)*.

Using SEAs and SEARs, the work shows that paraphrases of questions can change a model's understanding of it. Simple change like *What* to *What's* or *What was* to *So what was* flip up to 2% of SQuAD dev set instances.

Another line of work [92] has used integrated gradients introduced in [132] to see which input features reading comprehension model performance can be attributed to. Instead of analyzing BiDAF, the paper analyses QANet [151], a newer model closely related to BiDAF which replaces its recurrent layers with self-attention or convolution layers. They analyse QANet's performance on the non-sensical sentence insertion dataset of [58] (called ADDSENT) by looking which question words the model focuses on most. They find two types of ADDSENT examples which when added to the context successfully trick the model: (1) ones where a contentful word in the question gets low attribution but the adversarially added sentence modifies it; (2) ones where a contentful word in the question that is not present in the context is added in the context.

4.8 Summary

We show BiDAF works well even with simple question representations. Given the biggest degradation in performance come when the attention mechanism relating every context word to every question word is switched off, we hypothesize that what the model learns is to match keywords in the question with keywords in the context - a very shallow form of reasoning. By incorporating more advanced attention from question words to context words we show a small gain in performance can be observed. Additionally, we create a new dataset by re-positing the CoNNL-2003 English NER task as a multi-span reading comprehension problem. To solve this dataset, we introduce a new version of BiDAF capable of extracting multiple spans (BiDAF-MS). Because we use synthetic questions in the dataset, results show that BiDAF-MS mostly relies on the context embedding branch of the model.

Chapter 5

A New NER Model

This chapter presents our second contribution.

5.1 Introduction

The ability to reason about entities in text is an important element of natural language understanding. Named entity recognition (NER) concerns itself with the identification of such entities. Given a sequence of words, the task of NER is to label each word with its appropriate corresponding entity type. Examples of entity types include *Person*, *Organization*, and *Location*. A special *Other* entity type is often added to the set of all types and is used to label words which do not belong to any of the other entity types.

Recently, neural network based approaches which use no language-specific resources, apart from unlabeled corpora for training word embeddings, have emerged. There has been a shift of focus from handcrafting better features to designing better neural architectures for solving NER.

In this thesis, we propose a new parallel recurrent neural network model for entity recognition. We show that rather than using a single LSTM component, as many other recent architecture have, we instead resort to using multiple smaller LSTM units. This has the benefit of reducing the total number of parameters in our model. We present results on the CoNNL 2003 English dataset and achieve the new state of the art results for models without help from an outside lexicon.

5.2 Named Entity Recognition

Named Entity Recognition can be posited as a standard sequence classification problem where the dataset $D = \{(\mathbf{X}_i, \mathbf{y}_i)\}_{i=1}^{k}$ consists of example label pairs where both the examples and the labels are themselves sequences of word vectors and entity types, respectively.

Specifically, an input example $\mathbf{X}_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,|X_i|})$ is a variable-length sequence of word vectors $\mathbf{x}_{i,j} \in \mathbb{R}^d$; the example's corresponding label $\mathbf{y}_i = (y_{i,1}, ..., y_{i,|X_i|})$ is a equal-length sequence of entity-type labels $y_{i,j} \in Y$ where Y is the set of all entity type labels and includes a special other 'O'-label with which all words that are not entities are labeled.

The goal is then to learn a parametrized mapping $f_\theta : \mathbf{X} \to \mathbf{y}$ from input words to output entity labels. One of the most commonly used class of models that handle this mapping are recurrent neural networks.

5.2.1 LSTM complexity

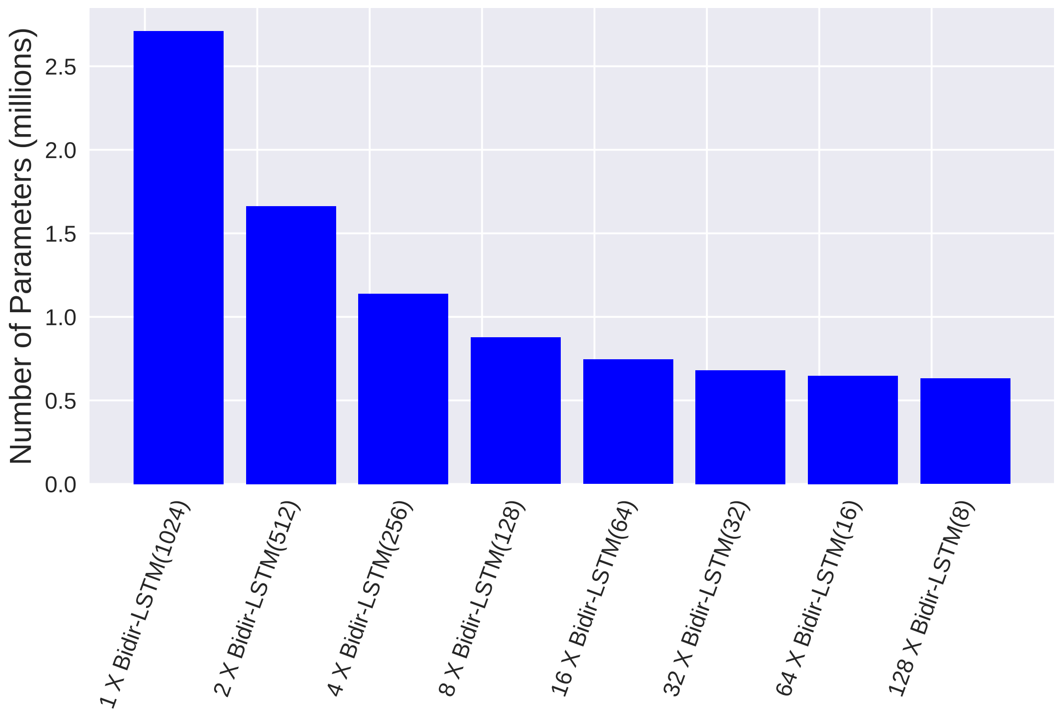

Figure 5.1: A chart showing how multi-LSTM parameter complexity decreases as we shrink n and increase K.

Long short term memory (LSTM) models belong to the family of recurrent neural network (RNN) models. They are often used as a component of much larger models, particularly in many NLP tasks including NER.

Classically, an LSTM cell is defined as follows (biases excluded for brevity):

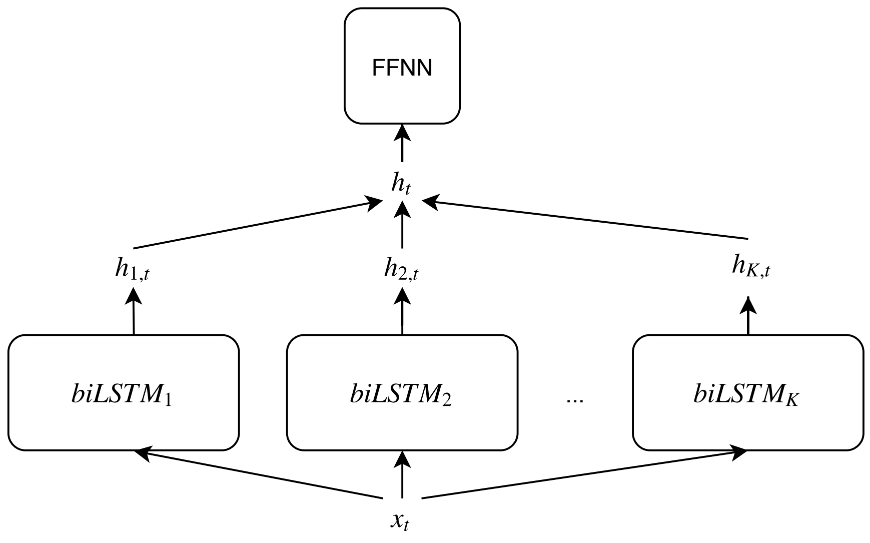

Figure 5.2: An example of a parallel LSTM. FFNN is a *feed-forward neural network*.

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\boldsymbol{W}_i\mathbf{h}_{t-1} + \boldsymbol{U}_i\mathbf{x}_t) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{h}_{t-1} + \boldsymbol{U}_f\mathbf{x}_t) \\
\mathbf{o}_t &= \sigma(\boldsymbol{W}_o\mathbf{h}_{t-1} + \boldsymbol{U}_o\mathbf{x}_t) \\
\tilde{\mathbf{c}}_t &= \tanh(\boldsymbol{W}_c\mathbf{h}_{t-1} + \boldsymbol{U}_c\mathbf{x}_t) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{5.1}$$

One way of measuring the complexity of a model is through its total number of parameters. Looking at the above, we note there are two parameter matrices, $\mathbf{W}$ and $\mathbf{U}$, for each of the three input gates and during cell update. If we let $\mathbf{W} \in \mathbb{R}^{n\times n}$ and $\mathbf{U} \in \mathbb{R}^{n\times m}$ then the total number of parameters in the model (excluding the bias terms) is $4(nm + n^2)$ which grows quadratically as n grows. Thus, increases in LSTM size can substantially increase the number of parameters.

5.3 Parallel RNNs

To reduce the total number of parameters we split a single LSTM into multiple equally-sized smaller ones:

$$h_{k,t} = \mathrm{LSTM}_k(h_{k,t-1}, \mathbf{x}) \tag{5.2}$$

where $k \in \{1, ..., K\}$. This has the effect of dividing the total number of parameters by a constant factor. The final hidden state h_t is then a concatenation of the hidden states of the smaller LSTMS:

$$h_t = [h_{1,t}; h_{2,t}; ...; h_{K,t}] \tag{5.3}$$

5.3.1 Promoting Diversity

To promote diversity amongst the constituent smaller LSTMs we add a orthogonality penalty *across* the smaller LSTMs. Recent research has used similar methods but applied to single LSTMs [140].

We take the cell update recurrence parameters $\mathbf{W}_i$ across LSTMs (we omit the c in the subscript for brevity; the index i runs across the smaller LSTMs) and for any pair we wish the following to be true:

$$\langle \text{vec}(W_c^{(i)}), \text{vec}(W_c^{(j)}) \rangle \approx 0 \tag{5.4}$$

To achieve this we pack the vectorized parameters into a matrix:

$$\Phi = \begin{pmatrix} \text{vec}(W_c^{(1)}) \\ \text{vec}(W_c^{(2)}) \\ \vdots \\ \text{vec}(W_c^{(N)}) \end{pmatrix} \tag{5.5}$$

and apply the following regularization term to our final loss:

$$\lambda \sum_i \|\Phi\Phi^\top - I\|_F^2 \tag{5.6}$$

5.3.2 Output and Loss

The concatenated output h_t is passed through a fully connected layer with bias before being passed through a final softmax layer:

$$o_t = \text{softmax}(\mathbf{W}_{\text{out}}\hat{h}_t + b_{\text{out}}) \tag{5.7}$$

To extract a predicted entity type $\hat{y}_t$ at time t, we select the entity type corresponding to the most probable output:

$$\hat{y}_t = argmax(o_t) \tag{5.8}$$

The loss is defined as the sum of the softmax cross-entropy losses along the words in the input sequence. More precisely, we denote by $y_t^j \in 0, 1$ a binary indicator variable indicating whether word x_t truly is an entity of type j. The loss at time t is then defined to be $\mathcal{L}_t = -\sum_j y_t^j \log(o_t^j)$. Thus the overall loss is:

$$\mathcal{L} = -\sum_t \sum_j y_t^j \log(o_t^j) \tag{5.9}$$

5.3.3 Relation to Ensemble Methods

Our model bears some resemblance to ensemble methods [31, 25], which combine multiple "weak learners" into a single "strong learner"; One may view each of the parallel recurrent units of our model as a single "weak" neural network, and may consider our architecture as a way of combining these into a single "strong" network.

Despite the similarities, our model is very different from ensemble methods. First, as opposed to many boosting algorithms [31, 119, 25] we do not "reweight" training instances based on the loss incurred on them by a previous iteration. Second, unlike ensemble methods, our model is trained *end-to-end*, as a single large neural network. All the subcomponents are co-trained, so different subparts of the network may focus on different aspects of the input. This avoids redundant repeated computations across the units (and indeed, we encourage diversity between the units using our inter-module regularization). Finally, we note that our architecture does not simply combine the *prediction* of multiple classifiers; rather, we take the final *hidden layer* of each of the LSTM units (which contains more information than merely the entity class prediction), and combine this information using a feedforward network. This allows our architecture to examine inter-dependencies between pieces of information computed by the various components.

Model	**F1**
[17]	88.31
[29]	88.76
[5]	89.31
[23]‡	89.59
[56]‡	90.10
[18]‡	90.77
[109]	90.80
[79]	90.90
[98]‡*	90.90
[70]‡	90.94
[81]‡	91.20
[83]‡	91.21
[118]	91.28
[18]‡*	91.62
[101]‡*	91.93
Our Model‡	**91.48** ±0.22

Table 5.1: English NER F1 score of our model on the test set of CoNLL-2003 (English). During training we optimize for the development set and report test set results for our best performing development set model. The bounded F1 results we report (±0.22) are taken after 10 runs. For the purpose of comparison, we also list F1 scores of previous top-performance systems. ‡ marks the neural models. * marks model which use external resources.

# RNN units	F_1
1	90.53 ±0.31
2	90.79 ±0.18
4	90.64 ±0.24
8	91.09 ±0.28
16	91.48 ±0.22
32	90.68 ±0.18

Table 5.2: Performance as a function of the number of RNN units with a fixed unit size of 64; averaged across 5 runs apart from the 16 unit model (averaged across 10 runs).

# RNN units	Unit size	F_1
1	1024	87.54
2	512	91.25
4	256	91.29
8	128	91.31
16	64	91.48 ±0.22
32	32	90.60
64	16	90.79
128	8	90.41

Table 5.3: Performance of our model with various unit sizes resulting in a fixed final output size h_t. Single runs apart from 16 unit.

Unit size	F_1
8	89.78
16	89.77
32	90.26
64	91.48 ±0.22
128	89.28

Table 5.4: Performance as a function of the unit size for our best performing model (16 biLSTM units). Single runs apart from with size 64.

Component	F_1
No character embeddings	90.39
No orthogonal regularization	90.79
No Xavier initialization	91.09
No variational dropout	91.03
Mean pool instead of concat	90.49

Table 5.5: Impact of various architectural decisions on our best performing model (16 biLSTM units, 64 unit size). Single runs.

5.4 Results

We achieve state-of-the-art results on the CoNNL 2003 English NER dataset (see Table 5.1). Although we do not employ additional external resources (language specific dictionaries or gazetteers), our model is competitive even with some of the models that do. To gain a better understanding of the performance of our model including how its various components affect performance we prepared four additional tables of runs.

Table 5.2 shows performance as a function of the number of RNN units with a fixed single unit LSTM size of 64. The number of units is a hyperparameter which must be tuned. We find good performance across the board and there is no catastrophic collapse in results. This means the model is stable and not over-sensitive to changes in this particular hyperparameter. We notice our top score is achieved when using 16 units.

In Table 5.3 the output size is fixed and the number of units and their size is changed to match. Results are mostly stable but an interesting outlier is the single unit, large (1024), LSTM which achieves a much poorer F1 score. This points to the benefit of our architecture which mimics ensemble models but in an end-to-end fashion. Even by increasing the number of RNN units to two, the result improves by more than 3.5 F1 points. Interestingly, even with 128 tiny 8-sized units, the performance remains stable.

Table 5.4 showcases what happens if we keep the number of LSTM units fixed. We pick our best model (16 units) and then change the unit sizes. We see that performance degrades more so than in the other tables when we deviate from the best model's combination. By looking at final output sizes across the tables (by multiplying the number of units by unit size) we see that performance degrades at its extremes: 16-units / 128-size (2048 output size) performs poorly as does 16/8 (128 output size). However, somewhere between these output extremes, it seems results are mostly stable and can be fine-tuned by adjusting the number of units and their size with the caveat that here too we find a substantial degradation in performance if we choose to only use a single LSTM unit.

Finally, Table 5.5 presents ablation results on our best performing model. We notice that not using character level embeddings followed by no otrhogonal regularization has the largest impact on results. Not using Xavier initialization or variational dropout also degrades performance but not as much. Mean pooling the outputs of the units into unit-sized output as opposed to concatenating them also achieves poorer results - but not by much. It would be interesting to further explore why this is since with an architecture such as ours, using a mean-pooled final layer would lead to efficiency gains. Instead of having to concatenated sixteen 64-dimensional vectors as is the case in our best model, we could just mean pool them into a single 64-dimensional vector.

5.5 Implementation Details

We use bidirectional LSTMs as our base recurrent unit and use pretrained word embeddings of size 100. These are the same embeddings used in [70]. We concatenate to our word embeddings character-level embeddings similar to [70] but with a max pooling layer instead. Unlike with the parallel LSTMs, we only use a single character embedding LSTM.

Parameters are initialized using the method described by [35] [35]. This approach scales the variance of a uniform distribution with regard to the root of the number of parameters in a layer. This approach has been found to speed up convergence compared to using a unit normal distribution for initialization.

Our model uses variational dropout [32] between the hidden states of the parallel LSTMs. Recent work has shown this to be very effective at training LSTMs for language models [85]. In our experiments, we use $p = 0.1$ as our dropping probability.

We experiment with different values of the regularization term parameter but settled on $\lambda = 0.01$.

Although vanilla stochastic gradient descent has been effective at training RNNs on language problems [85], we found that using the ADAM optimizer [64] to be more effective at training our model. We experimented with different values for the learning rate α, increasing α from 10^{-3} to as high as 5×10^{-3} and still obtained good results.

Similarly, we kept a constant size for the character-level embeddings, using a unit bidirectional LSTM output size of 50. As previously discussed, we trained the network parameters using the ADAM optimizer [64].

5.6 Related Work

Various approaches have been proposed to NER. Many of these approaches rely on hand-crafted feature engineering or language-specific or domain-specific resources [154, 17, 29, 124, 94]. While such approaches can achieve high accuracy, they may fail to generalize to new languages, new corpora or new types of entities to be identified. Thus, applying such techniques in new domains requires making a heavy engineering investment.

Over time neural methods such as [18, 83, 81, 70] emerged. More recently [101, 111, 118] have set the top benchmarks in the field.

Architecturally, our model is similar to those of [155, 48] with the most pronounced difference being that we (1) apply our parallel RNN units across the same input (2) explore a new regularization term for promoting diversity across what features our parallel RNNs extract and (3) explicitly motivate the architecture with a discussion about parameter complexity.

The need for a wider discussion on parameter complexity in the deep learning community is being pushed by the need to make complex neural models runnable in constrained environment such as field-programmable gate arrays (FPGAs) - for a great discussion relating to running LSTMs on FPGAs see [43]. Additionally, complex models have proven difficult to use in certain domains such as embedded systems or finance due to their slowness. Our architecture lends itself to parallelization and attempts to tackle this problem.

5.7 Summary

We achieve state-of-the-art results on the CoNLL 2003 English dataset and introduce a new model motivated primarily by its ability to be easily distributable and reduce the total number of parameters. Further work should be done on evaluating it across different classification and sequence classification tasks to study its performance. Additionally, a run-time analysis should be conducted to compare speedups if the model is parallelized across CPU cores.

Chapter 6

A New Answer Selection Model

This chapter presents our third contribution.

6.1 Introduction

Question answering (QA) relates to the building of systems capable of automatically answering questions posed by humans in natural language. Various frameworks have been proposed for question answering, ranging from simple information-retrieval techniques for finding relevant knowledge articles or webpages, through methods for identifying the most relevant sentence in a text regarding a posed question, to methods for querying structured knowledge-bases or databases to produce an answer [14, 139, 68, 50, 110]

A popular QA task is *answer selection*, where, given a question, the system must pick correct answers from a pool of candidate answers [149, 59, 65, 71, 125].

Answer selection has many commercial applications. Virtual assistants such as Amazon Alexa and Google Assistant are designed to respond to natural language questions posed by users. In some cases such systems simply use a search engine to find relevant webpages; however, for many kinds of queries, such systems are capable of providing a concise specific answer to the posed question.

Similarly, various AI companies are attempting to improve customer service by automatically replying to customer queries. One way to design such a system is to curate a dataset of historical questions posed by customers and the responses given to these queries by human customer service agents. Given a previously unobserved query, the system can then locate the best matching answer in the curated dataset.

Answer selection is a difficult task, as typically there is a large number of possible answers which need to be examined. Furthermore, although in many cases the correct answer is lexically similar to the question, in other cases semantic similarities between words must be learned in order to find the correct answer [66, 4]. Additionally, many of the words in the answer may not be relevant to the question.

Consider, for example, the following question answer pair:

How do I freeze my account?

Hello, hope you are having a great day. You can freeze your account by logging into our site and pressing the freeze account button. Let me know if you have any further questions regarding the management of your account with us.
Intuitively, the key section which identifies the above answer as correct is "[...] you can freeze your account by [...]", which represents a small fraction of the entire answer.

Earlier work on answer selection used various techniques, ranging from information retrieval methods [22] and machine learning methods relying on hand-crafted features [142, 143]. Deep learning methods, which have recently shown great success in many domains including image classification and annotation [67, 153, 76], multi-annotator data fusion [3, 33], NLP and conversational models [37, 9, 77, 61, 127] and speech recognition [37, 3], have also been successfully applied to question answering [27]. Current state-of-the-art methods use recurrent neural network (RNN) architectures which incorporate attention mechanisms [133]. These allow such models to better focus on relevant sections of the input [9].

Our contribution: We propose a new architecture for question answering. Our high-level approach is similar to recently proposed QA systems [27, 133], but we augment this design with a more sophisticated attention mechanism, combining the *local* information in a specific part of the answer with a *global* representation of the entire question and answer.

We evaluate the performance of our model using the recently released *InsuranceQA dataset* [27], a large open dataset for answer selection comprised of insurance related questions such as: "what can you claim on Medicare?". [1]

We beat state-of-the-art approaches [27, 133], and achieve good performance even when using a relatively small network.

6.2 Preliminaries

Our approach is similar to the *Answer Selection Framework* of Tan et al. [133], but we propose a different network architecture and a new attention mechanism. We first provide a high level description of this framework (see the original paper for a more detailed discussion), then discuss our proposed attention mechanism.

[1]As opposed to other QA tasks such as answers extraction or machine text comprehension and reasoning [147, 105], the InsuranceQA dataset questions do not generally require logical reasoning.

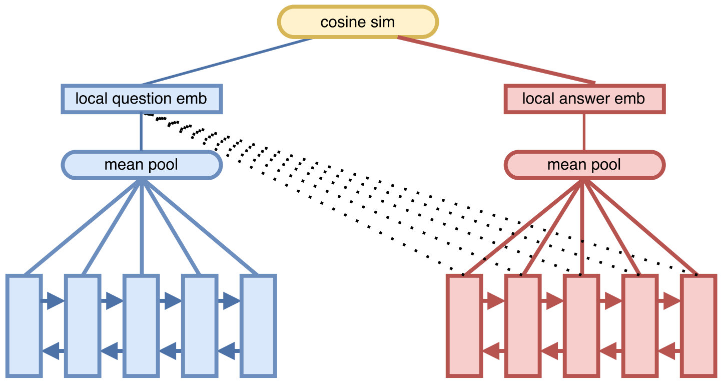

Figure 6.1: Model architecture using answer-localized attention [133]. The left hand side used for the question. The right side of the architecture is used for both the answer and distractor.

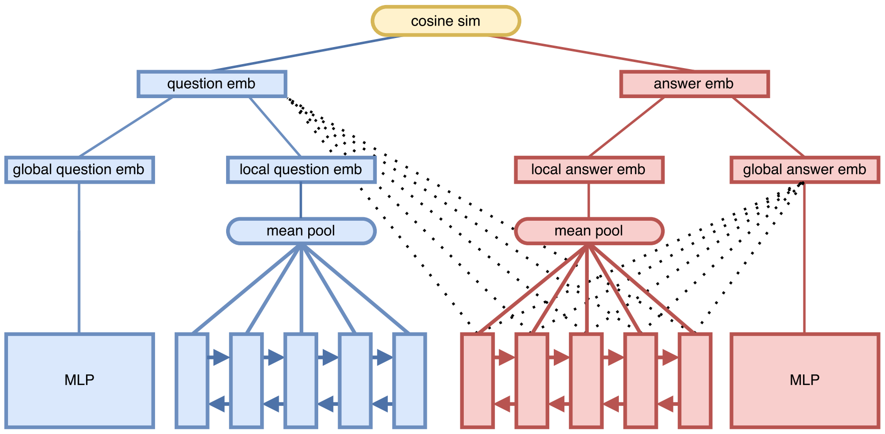

Figure 6.2: Our proposed architecture with augmented attention. As in Figure 6.1, the right side of the model is used to embed answers and distractors.

The framework is based on a neural network with parameters θ which can embed either a question q or a candidate answer a into low dimensional vectors $r \in R^k$. The network can embed a question with no attention, which we denote as $f_\theta(q)$, and embed a candidate answer with attention to the question, denoted as $g_\theta(a, q)$. We denote the similarity function used as $s(x, y)$ (s may be the dot product function, the cosine similarity function or some other similarity function).

Given a trained network, we compute the similarity between question and answer embeddings:

$$s_i = s(f_\theta(q), g_\theta(A_i, q)) \tag{6.1}$$

for any $i \in 1, 2, \ldots, k$ with A_i being the ith candidate answer in the pool. We then select the answer yielding the highest similarity $\arg\max_i s_i$.

The embedding functions, f_θ and g_θ, depend on the architecture used and the parameters θ. The network is trained by choosing a loss function $\mathcal{L}$, and using stochastic gradient descent to tune the parameters given the training data. Each training item consists of a question q, the correct answer a^* and a distractor d (an incorrect answer). A prominent choice is using a shifted hinge loss, designating that the correct answer must have a higher score than the distractor by at least a certain margin M, where the score is based on the similarity to the question.

$$\mathcal{L} = \max\Big\{0, M - \sigma_{a^*} + \sigma_d\Big\} \tag{6.2}$$

where:

$$\sigma_{a^*} = s\Big(f_\theta(q), g_\theta(a^*, q)\Big) \tag{6.3}$$

$$\sigma_d = s\Big(f_\theta(q), g_\theta(d, q)\Big) \tag{6.4}$$

The above expression has a zero loss if the correct answer has a score higher than the distractor by at least a margin M, and the loss linearly increases in the score difference between the correct answer and the distractor.

Any reasonable neural network design for f_θ can be used to build a working answer-selection systems using the above approach; however, the network design can have a big impact on the system's accuracy.

6.2.1 Embedding Questions and Answers

Earlier work examined multiple approaches for embedding questions and answers, including convolutional neural networks, recurrent neural networks (RNNs) (sometimes augmented with an attention mechanism) and hybrid designs [27, 133].

An RNN design "digests" the input sequence, one element at a time, changing its internal state at every timestep. The RNN is based on a cell, a parametrized function mapping a current state and an input element to the new state [146]. A popular choice for the RNN's cell is the Long Short Term Memory (LSTM) cell [52].

Given a question comprised of words $q = (x_1, x_2, \ldots, x_m)$, we denote the i'th output of an LSTM RNN digesting the question as q_i; similarly given an answer $a = (y_1, y_2, \ldots, y_n)$ we denote the j'th output of an LSTM RNN digesting the question as a_j.

One simple approach is to have the embeddings of the question and answer be the last LSTM output, i.e. $f_\theta(q) = q_m$ and $f_\theta(a) = a_n$. Note that q_i, a_i are vectors whose dimensionality depends on the dimensionality of the LSTM cell; we denote by $q_{i,j}$ the j'th coordinate of the LSTM output at timestep i.

Another alternative is to aggregate the LSTM outputs across the different timesteps by taking their coordinate-wise mean (mean-pooling):

$$f_\theta(q)_r = \frac{1}{m}\sum_{i=1}^{m} q_{i,r} \tag{6.5}$$

Alternatively, one may aggregate by taking the or coordinate-wise max (max-pooling):

$$f_\theta(q)_r = max_{i=1}^{m} q_{i,r} \tag{6.6}$$

We use another simple way of embedding the question and answer, which is based on term-frequency (TF) features. Given a vocabulary of words $V = (w_1, \ldots, w_v)$, and a text p we denote the TF representation of p as $p^{\text{tf}} = (d_1, \ldots, d_v)$ where $d_j = 1$ if the word w_j occurs in p and otherwise $d_j = 0$. [2]

A simple overall embedding of a text p is $p' = Wt(p)$ where W is an $v \times d$ matrix, and where d determines the final embedding's dimensionality; the weights of W are typically part of the neural network parameters, to be learned during the training of the network. Instead of a single matrix multiplication, one may use the slightly more elaborate alternative of applying a feedforward network, in order to allow for non-linear embeddings.

We note that a TF representation loses information regarding the *order* of the words in the text, but can provide a good global view of key topics discussed in the text.

Our main contribution is a new design for the neural network that ranks candi-

[2]Another alternative is setting d_j to the *number* of times the word w_j appears in p. A slightly more complex option is using TF-IDF features [107] or an alternative hand-crafted feature scheme; however we opt for the simpler TF representation, letting the neural network learn how to use the raw information.

date answers for a given question. Our design uses a TF-based representation of the question and answer, and includes a new attention mechanism which uses this global representation when computing the attention weights (in addition to the local information used in existing approaches). We describe existing attention designs (based on local information) in Section 6.2.2, before proceeding to describe our approach in Section 6.3.

6.2.2 Local Attention

Early RNN designs were based on applying a deep feedforward network at every timestep, but struggled to cope with longer sequences due to exploding and diminishing gradients [53]. Other recurrent cells such as the LSTM and GRU cells [53, 21] have been proposed as they alleviate this issue; however, even with such cells, tackling large sequences remains hard [87]. Consider using an LSTM to digest a sequence, and taking the final LSTM state to represent the entire sequence; such a design forces the system to represent the entire sequence using a single LSTM state, which is a very narrow channel, making it difficult for the network to represent all the intricacies of a long sequence [9].

Attention mechanisms allow placing varying amounts of emphasis across the entire sequence [9], making it easier to process long sequences; in QA, we can give different weights to different parts of the answer while aggregating the LSTM outputs along the different timesteps:

$$f_\theta(a) = \sum_{i=1}^{m} \alpha_i a_{i,r} \tag{6.7}$$

where α_i denotes the weight (importance) placed on timestep i and $a_{i,r}$ is the rth value of the ith embedding vector.

Tan et al. [133] proposed a very simple attention mechanism for QA, shown in Figure 6.1:

$$m_{a,q}(i) = W_{ad} a_i + W_{qd} f_\theta(q) \tag{6.8}$$

$$\alpha_i \propto exp(w_{ms}^T \tanh(m_{a,q}(i))) \tag{6.9}$$

$$\hat{h}_a(i) = h_a(i)\alpha_i \tag{6.10}$$

$$\hat{a} = \sum_{i=1}^{m} \alpha_i a_i \tag{6.11}$$

where $\alpha_i a(i)$ is the weighted hidden layer, W_{ad} and W_{qd} are matrix parameters to be learned, and w_{ms} is a vector parameter to be learned.

6.3 Global-Local Attention

A limitation of the attention mechanism of Tan et al. [133] is that it only looks at the the embedded question vector and one candidate answer word embedding at a time. Our proposed attention mechanism adds a *global* view of the candidate, incorporating information from *all* words in the answer.

6.3.1 Creating Global Representations

One possibility for constructing a global embedding is an RNN design. However, RNN cells tend to focus on the more recent parts of an examined sequence [87]. We thus opted for using a term-frequency vector representing the entire answer, as shown in Figure 6.2. We denote this representation as:

$$a^{\text{tf}} = (d_1, d_2, \ldots, d_v) \tag{6.12}$$

where d_i relates to the i'th word in our chosen vocabulary, and $d_i = 1$ if this word appears in the candidate answer, and $d_i = 0$ otherwise.

Consider a candidate answer $a = (y_1, \ldots, y_n)$, and let $(a_1, \ldots, a_n)$ denote its sequence of RNN LSTM outputs, i.e. a_i denotes the i'th output of a RNN LSTM processing this sequence (so a_i is a vector whose dimensionality is as the hidden size of the LSTM cell). We refer to a_i as the local-embedding at time i. [3]

6.3.2 Combining Local and Global Representations to Determine Attention Weights

The goal of an attention mechanism is to construct an overall representation of the candidate answer a, which is later compared to the question representation to determine how well the candidate answers the question; this is achieved by obtaining a set of weights $w_1, \ldots, w_n$ (where $w_i \in \mathbb{R}^+$), and constructing the final answer representation as a weighted average of the LSTM outputs, with these weights.

Given a candidate answer a, we compute the attention coefficient w_i for timestep i as follows.

First, we combine the local view (the LSTM output, more heavily influenced by the words around timestep t) with the global view (based on TF features of all the words in the answer). We begin by taking linear combinations of the TF features then passing

[3]Note that although we call a_i a local embedding, the i'th LSTM state does of course take into account other words in the sequence (and not only the i'th word). By referring to it as "local" we simply mean to say that it is more heavily influenced by the i'th word or words close to it in the sequence.

them through a tanh nonlinearity (so that the range of each dimension is bounded in $[-1,1]$):

$$b^{\text{tf}} = \tanh(W_1 a^{\text{tf}}) \tag{6.13}$$

The weights of the matrix W_1 are model parameters to be learned, and its dimensions are set so as to map the sparse TF vector a^{tf} to a dense low dimensional vector (in our implementation b^{tf} is a 50 dimensional vector).

Similarly, we take a linear combination of the different dimensions of the local representation a_i (in this case there is no need for the *tanh* operation, as the LSTM output is already bounded):

$$b_i^{\text{loc}} = W_2 a_i \tag{6.14}$$

where the weights of the W_2 are model parameters to be learned (and with dimensions set so that b_i^{loc} would be a 140 dimensional vector).

Given a TF representation of a text x^{tf}, whose dimensionality is the size of the vocabulary, and an RNN representation of the text x^{rnn}, with a certain dimentionality h, we may wish construct a normalized representation of the text. As the norms of these two parts may differ, simply concatenating these parts may result in a vector dominated by one side. We thus define a joint representation $h(x^{\text{tf}}, x^{\text{rnn}})$ as follows.

We normalize each part so as to have a desired ratio of norms $\frac{\alpha}{\beta}$ between the RNN and TF representations; this ratio reflects the relative importance of the RNN and TF embeddings in the combined representation (for instance when settings both α, β to 1 both parts would have a unit norm, giving them equal importance):

$$c^{\text{tf}} = \frac{\alpha}{||x^{\text{tf}}||} \cdot x^{\text{tf}} \tag{6.15}$$

$$c^{\text{rnn}} = \frac{\beta}{||x^{\text{rnn}}||} \cdot x^{\text{rnn}} \tag{6.16}$$

We then concatenate the normalized TF and RNN representations to generate the joint representation:

$$h(x^{\text{tf}}, x^{\text{rnn}}) = c^{\text{tf}} \| c^{\text{rnn}} \tag{6.17}$$

where $\|$ represents vector concatenation.

We construct the local attention representation at the i'th word of the answer as:

$$a_i^{\text{glob-loc}} = h(b^{\text{tf}}, b_i^{\text{loc}}) \tag{6.18}$$

using values of $\alpha = 0.5, \beta = 1$.

The raw attention coefficient of the i'th word in the answer is computed by measuring the similarity of a vector representing the question, and a local-global representation of the answer at word i. We build these representations, of matching dimensions, by taking the same number of linear combinations from $a_i^{\text{glob-loc}}$ (the raw global-local representation of the answer at word i). Thus the attention weight for the i'th word is:

$$\alpha_i' = sim\Big(W_3 a_i^{\text{glob-loc}}, W_4 f_\theta(q)\Big) \tag{6.19}$$

where W_2, W_3 are matrices whose weights are parameters to be learned (and whose dimensions are set so that $W_3 a_i^{\text{glob-loc}}$ and $W_4 f_\theta(q)$ would be vectors of identical dimensionality, 140 in our implementation), and where sim denotes the cosine similarity between vectors:

$$sim(u,v) = \frac{u \cdot v}{||u|| \cdot ||v||} \tag{6.20}$$

with the $\cdot$ symbol in the nominator denoting the dot product between two vectors.

Finally, we normalize the attention coefficients with respect to their exponent to obtain the final attention weights, by applying the softmax operator on the raw attention coefficients. We take the raw attention coefficients, $\alpha' = (\alpha_1', \alpha_2', \ldots, \alpha_m')$ and define the final attention weights $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ where $\alpha_i \propto exp(\alpha_i')$ and α is the result of the softmax operator applied on α:

$$\alpha_i = \frac{\exp(\alpha_i')}{\sum_{j=1}^{m} \exp(\alpha_j')} \tag{6.21}$$

6.3.3 Building the Final Attention Based Representation

The role of the attention weights is building a final representation of a candidate answer; different answers are ranked based on the similarity of their final representation and a final question representation. Similarly to the TF representation of the answer, we denote the TF representation of the question as: $q^{\text{tf}} = (r_1, r_2, \ldots, r_v)$, where r_i relates to the i'th word in our chosen vocabulary, and $r_i = 1$ if this word appears in the question, and $r_i = 0$ otherwise. Our final representation of the question is a joining of the TF representation of the question and the mean pooled RNN question representation (somewhat similarly to how we join the TF and RNN representation when determining the attention weights):

$$f_\theta'(q) = h(q^{\text{tf}}, f_\theta(q)) \tag{6.22}$$

Our final representation of the answer is also a joining two parts, a TF part a^{tf} (as defined earlier) and an attention weighted RNN part $\hat{a}$. We construct $\hat{a}$ as the weighted average of the LSTM outputs, where the weights are the attention weights defined above:

$$\hat{a} = \sum_{i=1}^{m} \alpha_i a_i \tag{6.23}$$

The final representation of the answer is thus:

$$f'_\theta(a) = h(a^{\mathrm{tf}}, \hat{a}) \tag{6.24}$$

Figure 6.2 describes the final architecture of our model, showing how we use a TF-based global embedding both in determining the attention weights and in the overall representation of the questions and answers. The dotted lines in the figures indicate that our model's attention weights depend not only on the local embedding but also on the global embedding.

6.3.4 Tuning Parameters to Minimize the Loss

The loss function $\mathcal{L}$ we use is the shifted hinge loss defined in Section 6.2. We compute the score of an answer candidate a as the similarity between its final representation $f'_\theta(a)$ and the final representation of the question $f'_\theta(q)$ [4] :

$$sim(f'_\theta(q), f'_\theta(a)) \tag{6.25}$$

Given the score of the correct answer candidate $\sigma_{a^*} = sim(f'_\theta(q), f'_\theta(a))$ and the score of a distractor (incorrect) candidate d, $\sigma_d = sim(f'_\theta(q), f'_\theta(d))$, our loss is $\mathcal{L} = \max\Big\{0, M - \sigma_{a^*} + \sigma_d\Big\}$.

The above loss relates to a single training item (consisting of a single question, its correct answer and an incorrect candidate answer). Training the neural network parameters involves iteratively examining items in a dataset consisting of many training items (each containing a question, its correct answer and a distractor) and modifying the current network parameters. We train our system using variant of stochastic gradient descent (SGD) with the Adam optimization [64].

6.4 Results

We evaluate our proposed neural network design in a similar manner to earlier evaluations of Siamese neural network designs [150, 126], where a neural network is trained

[4] We use the cosine similarity as our similarity function for the loss, though other similarity functions can also be used.

to embed both questions and candidate answers as low dimensional vectors.

Question: how much do The Average American pay for Health Insurance

the average American pay between $0 and $2000 per month for their health insurance yes that be a huge range for example if an employer pay the entire premium your out of pocket cost be $0 conversely if someone be receive cobra benefit and they be in their 60s the rate can possibly be in the $2,000 per month range and of course rate will vary from 1 state to another begin in 2,014 federal tax subsidy will become available which can potentially reduce premium for many American

Question: do Medicare d cover shingle shot

some vaccine be cover Medicare , some not the herpes zoster chicken pox vaccine be 1 that be cover and not if the vaccine be administer in your doctor office Medicare will not cover the actual cost of the vaccine if bill your doctor you must purchase the vaccine under part d at a pharmacy you can then transport the vaccine to your doctor so they can inject you make complete sense right

Figure 6.3: A visualization of the attention weights for each word in a correct answer to a question. These examples show how the attention mechanism is focusing on relevant parts of the correct answer (although the attention is still quite noisy).

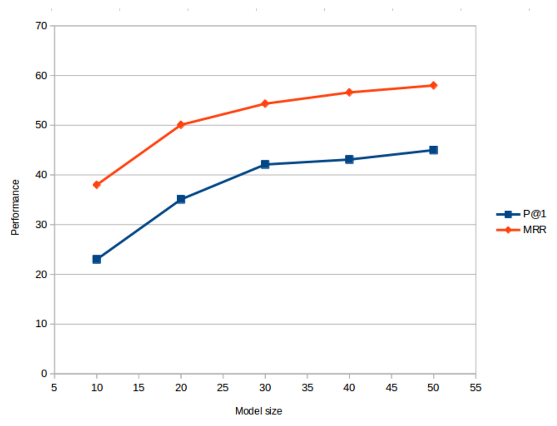

Figure 6.4: Performance of our system on InsuranceQA for various model sizes h (both the LSTM hidden layer size and embedding size)

Table 6.1 presents the results of our model and the various baselines for InsuranceQA. The performance metric used here is P@1, the proportion of instances where a correct answer was ranked higher than all other distractors in the pool. The table shows that our model outperforms the previous baselines.

We have also examined the performance of our model as a function of its size (determining the system's runtime and memory consumption). We used different values $h \in \{10, 20, 30, 40, 50\}$ for both the size of the LSTM's hidden layer size and embedding size, and examined the performance of the resulting QA system on InsuranceQA.

Model	Test1	Test2
Bag-of-words	32.1	32.2
Metzler-Bendersky	55.1	50.8
Arch-II [27]	62.8	59.2
Arch-II GSED [27]	65.3	61.0
Attention LSTM [133]	69.0	64.8
TF-LSTM Concatenation	62.1	61.5
Local-Global Attention	**70.1**	**67.4**

Table 6.1: Performance of various models on InsuranceQA

Our results are given in Figure 6.4, which shows both the P@1 metric and the mean reciprocal rank (MRR) [24, 16] [5]

Figure 6.4 shows that performance improves as the model gets larger, but the returns on extending the model size quickly diminish. Interestingly, even relatively small models achieve a reasonable question answering performance.

To show our attention mechanism is necessary to achieve good performance, we also construct a model that simply concatenates the output of the feedforward network (on TF features) and the output of the bidirectional LSTM, called TF-LSTM concatenation. While this model does make use of TF-based features in addition to the LSTM state of the RNN, it does not use an attention mechanism to allow it to focus on the more relevant parts of the text. As the table shows, the performance of the TF-LSTM model is significantly lower than that of our model with the global-local attention mechanism. This indicates that the improved performance stems from the model's improved ability to focus on the relevant parts of the answer (and not simply from having a larger capacity and including TF-features).

Finally, we examine the the attention model's weights to evaluate it qualitatively. Figure 6.3 visualizes the weights for two question-answer pairs, where the color intensity reflects the relative weight placed on the word (the α_i coefficients discussed earlier). The figure shows that our attention model can focus on the parts of the candidate answer that are most relevant for the given question.

6.5 Implementation Details

We use the InsuranceQA dataset and its evaluation framework [27]. The InsuranceQA dataset contains question and answer pairs from the insurance domain, with roughly 25,000 unique answers, and is already partitioned into a training set and two test sets,

[5]The MRR metric assigns the model partial credit even in cases where the highest ranking candidate is an incorrect answer, with the score depending on the highest rank of a correct answer.

called test 1 and test 2.

The InsuranceQA dataset has relatively short questions (mean length of 7). However, the answers are typically very long (mean length of 94).

At test time the system takes as input a question q and a pool of candidate answers $P = (a_1, a_2, \ldots, a_k)$ and is asked to select the best matching answer a^* to the question from the pool. The InsuranceQA comes with answer pools of size $k = 500$, consisting of the correct answers and random distractors chosen from the set of answers to other questions.

State-of-the-art results for InsuranceQA were achieved by Tan et al [133], which also provide a comparison with several baselines: Bag-of-words (with IDF weighted sum of word vectors and cosine similarity based ranking), the Metzler-Bendersky IR model [11], and [27] - the CNN based Architecture-II and Architecture-II with Geometricmean of Euclidean and Sigmoid Dot product (GESD).

We implemented our model in TensorFlow [2] and conducted experiments on NVIDIA Tesla K80s.

We use the same hidden layer sizes and embedding size as Tan et al. [133]: $h = 141$ for the bidirectional LSTM size and an embedding size of $e = 100$; this allows us to investigate the impact of our proposed attention mechanism. [6]

6.6 Related Work

Answer selection systems can be evaluated using various datasets consisting of questions and answers. Early answer selection models were commonly evaluated against the QASent dataset [143]; however, this dataset is very small and thus less similar to real-world applications. Further, its candidate answer pools are created by finding sentences with at least one similar (non-stopword) word as compared to the question, which may create a bias in the dataset.

Wiki-QA [150] is a dataset that contains several orders of magnitude more examples than QASent, where the candidate answer pools were created from the sentences in the relevant Wikipedia page for a question, reducing the amount of keyword bias in the dataset compared to QASent.

Our analysis is based on the InsuranceQA [27] dataset, which is much larger, and similar to real-world QA applications. The answers in InsuranceQA are relatively long (see details in Section **??**), so the candidate answers are likely to contain content that

[6] As is the case with many neural networks, increasing the hidden layer size or embedding size can improve the performance on our InsuranceQA models; we compare our performance to the work of Tan et al. [133] with the same hidden and embedding sizes; similarly to them we use embeddings pre-trained using Word2Vec [88] and avoid overfitting by applying early stopping (we also apply Dropout [130, 152]).

does not relate directly to the question; thus, a good QA model for InsuranceQA must be capable of identifying the most important words in a candidate answer.

Early work on answer selection was based on finding the semantic similarity between question and answer parse trees using hand-crafted features [142, 143]. Often, lexical databases such as WordNet were used to augment such models [15]. Not only did these models suffer from using hand-crafted features, those using lexical databases were also often language-dependent.

Recent attempts at answer selection aim to map questions and candidate answers into n-dimensional vectors, and use a vector similarity measure such as cosine similarity to judge a candidate answer's affinity to a question. In other words, the similarity between a question and a candidate is high if the candidate answers the question well, low if the candidate is not a good match for the question.

Such models are similar to Siamese models, a good review of which can be found in Muller et al's paper [93]. Feng et al. [27] propose using convolutional neural networks to vectorize both questions and answers before comparing them using cosine similarity. Similarly, Tan et al. [133] use a recurrent neural network to vectorize questions and answers. Attention mechanisms have proven to greatly improve the performance of recurrent networks in many tasks [9, 133, 115, 116, 82], and indeed Tan et al. [133] incorporate a simple attention mechanism in their system.

6.7 Summary

We proposed a new neural design for answer selection, using an augmented attention mechanism, which combines both local and global information when determining the attention weight to place at a given timestep. Our analysis shows that our design outperforms earlier designs based on a simpler attention mechanism which only considers the local view.

Several questions remain open for future research. First, the TF-based global view of our design was extremely simple; could a more elaborate design, possibly using convolutional neural networks, achieve better performance?

Second, our attention mechanism joins the local and global information in a very simple manner, by normalizing each vector and concatenating the normalized vectors. Could a more sophisticated joining of this information, perhaps allowing for more interaction between the parts, help further improve the performance of our mechanism?

Finally, can the underlying principles of our global-local attention design improve the performance of other systems, such as machine translation or image processing systems?

Part III

A New Dataset

Chapter 7

A New Dataset for Conflict Incident Question Answering

This chapter presents our fourth contribution.

7.1 Introduction

Many reports about armed conflict related incidents are published every day. However, these reports on the deaths and injuries of civilians and combatants often get forgotten or go unnoticed for long periods of time. Automatically extracting casualty counts from such reports would help better track ongoing conflicts and help us understand past ones.

One popular approach of discovering incidents is to identify them from textual reports and extract casualty, and other, information from them. This can either be done by hand or automatically. The Iraq Body Count (IBC) project has been directly recording casualties since 2003 for the ongoing conflict in Iraq [57, 47]. IBC staff collect reports, link them to unique incidents, extract casualty information, and save the information on a per incident basis as can be seen in Table 7.1.

Direct recording by hand is a slow and tedious process and notable efforts to do so have tended to lag behind the present. Information extraction systems capable of automating this process must explicitly or implicitly successfully solve three tasks: (1) find and extract casualty information in reports (2) detect events mentioned in reports (3) deduplicate detected events into unique events which we call *incidents*. The three tasks correspond to named entity recognition, slot filling, and de-duplication. The former two (NER and slot filling) can be posited as a question answering problem.

In this work we introduce the report based IBC-C dataset. Each report can contain

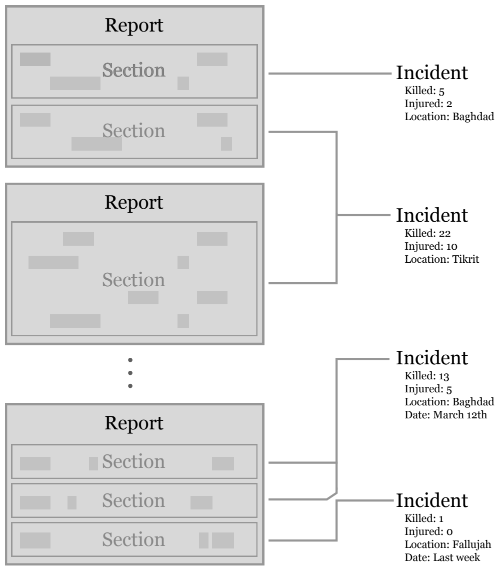

Figure 7.1: The IBC-C dataset visualised. A report is split into one or more non overlapping sections. A section is comprised of sentences which are comprised of words. Each section is linked to exactly one incident which in turn can be linked to one or more sections.

one or more sections; each section, one or more sentences; each sentence, one or more words. Each word is tagged with one of nine entity tags in the inside-outside-beginning (IOB) style. A visual representation of the dataset can be seen in Figure 7.1 and its statistics in Table 7.9.

To the best of our knowledge apart from the significantly smaller MUC-3 and MUC-4 datasets (which aren't casualty-specific) there are no other publicly available datasets made specifically for tasks (1), (2) or (3). The IBC-C dataset can be used to train supervised models for all three tasks.

We provide baseline results for task (1) which we posit as a sequence-classification problem and solve using a HMM, a CRF, and an RNN.

Since the 1990s the conflict analysis and NLP/IE communities have diverged. With the IBC-C dataset we hope to bring the two communities closer again.

7.2 Creating the IBC-C Dataset

7.2.1 Preprocessing

The Iraq Body Count project (IBC) has been recording conflict-related incidents from the Iraq war since 2003. An incident is a unique event related to war or other forms of

Incident ID	**Start date**	**End date**
d3473	22 Mar 2003	22 Mar 2003
Min killed	**Max killed**	**Min injured**
2	2	8
Max injured	**Location**	**Cause of death**
9	Khurmal	Suicide car bomb
Sources	**Town**	**Province**
BBC 23 Mar DPA 23 Mar	Khurmal	Sulaymaniyah
Alt. province	**District**	**Alt district**
/	Halabja	/
Killed Subjects		
Person 1, Person 2, ...		
Injured Subjects		
Person 3, Person 4, ...		
Report Sections		
BBC: "On Saturday **Person 1** died in **Khurmal** ..." DPA: "**2** people died yesterday afternoon..."		

Table 7.1: An example of an incident hand coded by IBC staff. Min and max values represent the minimum and maximum figures quoted in report sections linked to the incident.

violence which led to the death or injury of people. An example can be seen in Table 7.1.

The recording of incidents by the IBC works as follows: IBC staff first collect relevant *reports* before highlighting *sections* of them which they deem relevant to individual incidents. Parts of the report outside the highlighted sections are discarded. Sections can be seen in Figure 7.1. Because of the way IBC staff highlight sections there are no overlapping sections in the IBC-C dataset. Events are then recognised from the highlighted sections and de-duplicated into incidents. A final description of the incident (e.g. death and injury counts, location and date) is agreed upon after multiple rounds of human checking.

In the preprocessing step we gathered all incidents which occurred between March 20th, 2003 and December 31st, 2013. We removed spurious incidents (e.g. where the minimum number killed is larger than the maximum number killed) and cleaned the section text by removing all formatting and changing all written-out numbers into their numeric form (e.g. 'three' to 3).

7.2.2 Annotation

Using the information extracted by the IBC (see Table 7.1) we annotated each section word with one of ten tags: *KNUM* and *INUM* for numbers representing the number killed and injured respectively; *KSUB* and *ISUB* for named individuals were killed or injured; *KOTHER* and *IOTHER* for unnamed people who were killed or injured (for example "The doctor was injured yesterday."); *LOCATION* for the location in which

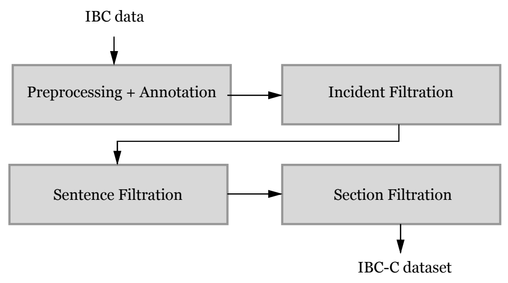

Figure 7.2: A visualisation of the different steps taken to create the dataset.

an incident occurred; *WEAPON* for any weapons used in an attack; *DATE* for words which identify when the incident happened; and, *O* for all other words.

Our data generation process can be thought of as a form of *distant supervision* [91] where we use agreed upon knowledge about an incident to label words contained within its sections instead of having hand-labeled individual words. This inevitably introduces errors which we try to mitigate using a filtration step where we remove ambiguous data.

7.2.3 Filtration

Simply annotating words based on the information in Table 7.1 can lead to wrong annotations. For example, if two people were recorded as having died in an incident, then, if another number two appears in the same sentence, this might lead to a wrong annotation. The sentence, "2 civilians were killed after 2 rockets hit the compound" could lead to the second '2' being annotated as a KNUM. Importantly, the actual *cardinality* of a number makes little difference to a sequence classifier compared to the difference a *misannotated* number would make. To minimise such misannotations we remove sentences and reports which do not pass all filtration criteria. Our filtration criteria consist of boolean functions over sentences, sections and incidents which return false if a test isn't passed.

The goal of filtration is to remove as much ambiguously labelled data as possible without biasing against any particular set of linguistic forms. There is thus a tradeoff which must be struck between linguistic richness and the quality of annotation. In our case we found that simple combinations of pattern matching and semantic functions worked well. No syntactic functions were used.

Incident Filtration

Incidents are filtered using a single criterion: if the minimum number of people killed or injured does not equal the maximum number of people killed or injured, respectively,

hasKSub	hasPerson	isKillSentence	toConsider	#
+	+	+	+	9774
+	+	-	-	7338
+	-	+	+	1527
+	-	-	-	1381
-	+	+	-	21715
-	+	-	+	34623
-	-	+	+	138307
-	-	-	+	169313
			+	353544
			-	30434
			Total	383978

Table 7.2: KSUB consideration rules

(Table 7.1) then the incident is removed. We do this so as to minimise any ambiguity in our named entity tagging (the only task for which we provide baseline results). This has the adverse effect of removing any incidents where reports mention different casualty counts. To compile a dataset which disregards this criterion, or considers a permissible window of casualties, a parameter in our dataset generating program may be changed.

Sentence Filtration

Filtering sentences is by far the hardest step. It is here where we must be careful to not bias against any linguistic forms. A separate set of boolean functions are applied to each sentence for each of the seven tags. These boolean functions or *rules*, as we call them, can be seen in Tables 7.3 - 7.9. We describe them in turn.

Table 7.2 and Table 7.3 define the filtration rules for KSUBs and ISUBs, respectively. Both rules eliminate sentences which contained named, killed or injured, subject but which aren't related to death or injury and instances where the sentence is related to death or injury and has persons tagged but now the ones we expect.

Tables 7.4 and 7.5 define the filtration rules for KNUMs and INUMs, respectively. These are the numbers mentioned in sentences which denote the number of people killed. We convert all numbers to their numerical form: for example, every *one* turns into a *1*. We then apply the filtration rules. Most of the boolean combinations in the case of KNUMs and INUMs are removed. The most decisive row for removing sentences in both cases is the $(- + - + -)$ row where 67,402 and 42,391 sentences are removed, respectively. This is the case where no KNUMs/INUMs were found despite

hasISub	hasPerson	isInjureSentence	toConsider	#
+	+	+	+	1386
+	+	-	-	2341
+	-	+	+	216
+	-	-	-	372
-	+	+	-	12300
-	+	-	+	57424
-	-	+	+	81597
-	-	-	+	228342
			+	368965
			-	15013
			Total	383978

Table 7.3: ISUB consideration rules

the sentences being connected to killings/injuries and other numbers being present. Since we only have incident-level annotations and reports tied to an incident often cite different numbers we are stuck with only being able to consider cases where the two match. In addition to this we take out any sentence which has a number and where no KNUM/INUM numbers were found elsewhere in the report.

Finally, tables 7.6, 7.7 and 7.8 define filtration rules for locations, dates, and weapons (usually the cause of death), respectively.

Report Filtration

Report filtering is simple and again done using only one rule. If any sentence a report contains fails to pass a single sentence-level test, then the whole report is removed.

7.2.4 Tasks

The above annotation and incident-, sentence-, and report-level filtration rules can all be used to construct different IBC-C datasets. We construct one such dataset to use as a NER dataset and describe how the rules can be used to produce other types of datasets.

hasKNUM	isKillSentence	hasOneTaggedAsKNUM	hasNumber	otherKNUMsInReport	toConsider	#
+	+	+	+	+	-	2445
+	+	+	+	-	+	7526
+	+	+	-	+	-	0
+	+	+	-	-	-	0
+	+	-	+	+	-	14624
+	+	-	+	-	+	30204
+	+	-	-	+	-	0
+	+	-	-	-	-	0
+	-	+	+	+	-	2119
+	-	+	+	-	-	1498
+	-	+	-	+	-	0
+	-	+	-	-	-	0
+	-	-	+	+	-	4282
+	-	-	+	-	-	4648
+	-	-	-	+	-	0
+	-	-	-	-	-	0
-	+	+	+	+	-	0
-	+	+	+	-	-	0
-	+	+	-	+	-	0
-	+	+	-	-	-	0
-	+	-	+	+	+	2757
-	+	-	+	-	-	67402
-	+	-	-	+	+	3360
-	+	-	-	-	+	43006
-	-	+	+	+	-	0
-	-	+	+	-	-	0
-	-	+	-	+	-	0
-	-	+	-	-	-	0
-	-	-	+	+	+	7573
-	-	-	+	-	-	47736
-	-	-	-	+	+	19749
-	-	-	-	-	+	125010
					+	239185
					-	144754
					Total	383939

Table 7.4: KNUM consideration rules. The *hasOneTaggedAsKNUM* column indicates whether the number '1' is tagged as a KNUM (it could also be a pronoun). The *isKillSentence* is determined by searching for kill-related keywords in the sentence.

hasINUM	isInjureSentence	hasOneTaggedAsINUM	hasNumber	otherINUMsInReport	toConsider	#
+	+	+	+	+	-	988
+	+	+	+	-	+	1557
+	+	+	-	+	-	0
+	+	+	-	-	-	0
+	+	-	+	+	-	10875
+	+	-	+	-	+	23962
+	+	-	-	+	-	0
+	+	-	-	-	-	0
+	-	+	+	+	-	1053
+	-	+	+	-	-	681
+	-	+	-	+	-	0
+	-	+	-	-	-	0
+	-	-	+	+	-	2943
+	-	-	+	-	-	3372
+	-	-	-	+	-	0
+	-	-	-	-	-	0
-	+	+	+	+	-	0
-	+	+	+	-	-	0
-	+	+	-	+	-	0
-	+	+	-	-	-	0
-	+	-	+	+	+	954
-	+	-	+	-	-	42381
-	+	-	-	+	+	1561
-	+	-	-	-	+	13220
-	-	+	+	+	-	0
-	-	+	+	-	-	0
-	-	+	-	+	-	0
-	-	+	-	-	-	0
-	-	-	+	+	+	5400
-	-	-	+	-	-	98645
-	-	-	-	+	+	12347
-	-	-	-	-	+	163960
					+	222961
					-	160938
					Total	383899

Table 7.5: INUM consideration rules

hasNERLocation	hasELOC	isInjureOrKillSentence	otherELOCsInReport	toConsider	#
+	+	+	+	-	5669
+	+	+	-	+	89038
+	+	-	+	-	1558
+	+	-	-	-	29973
+	-	+	+	-	0
+	-	+	-	-	35858
+	-	-	+	-	0
+	-	-	-	+	24540
-	+	+	+	-	270
-	+	+	-	+	5499
-	+	-	+	-	223
-	+	-	-	-	4446
-	-	+	+	-	0
-	-	+	-	-	76516
-	-	-	+	+	0
-	-	-	-	+	110385
				+	229462
				-	154513
				Total	383975

Table 7.6: LOCATION consideration rules

hasNERDate	hasEDate	toConsider	#
+	+	+	19970
+	-	+	77032
-	+	-	25
-	-	+	286883
		+	383885
		-	25
		Total	383910

Table 7.7: DATE consideration rules

hasWeapon	hasInjuredOrKilledInfo	toConsider	#
+	+	+	132097
+	-	-	56066
-	+	+	80749
-	-	+	115075
		+	327921
		-	56066
		Total	383987

Table 7.8: WEAPON (usually the cause of death) consideration rules

Element	**Count**
incidents	9,184
sections	18,379
reports	16,405
sentences	35,295
words	857,465
KNUM	13,597
INUM	6,689
KSUB	14,395
ISUB	1,036
LOCATION	25,251
DATE	4,765
WEAPON	35,617

Table 7.9: NER dataset statistics. Fully capitalized words indicate named entity tags.

Named Entity Recognition

Each word in the IBC-C dataset is tagged with one of nine (excluding O) entity tags as can be seen in Table 7.9 and can be thought of as subsets of more common named entity tags such as person or location. The dataset can be used to train a supervised NER model for conflict-specific named entity recognition. Note that the reason why there are fewer sentences in the NER dataset than in the filtration rules is because we apply report filtering (as described in Section 7.2.3) which reduces the number of considered sentences.

Slot Filling and Relationship Extraction

Each IBC-C event can be thought of as a 7-slot *event* template where each slot is named after an entity tag. The important thing to keep in mind is that a report may contain more than one section so just correctly recognising the entities isn't enough to solve the slot filling task. Instead, if a report mentions two events then two separate templates must be created and their slots filled.

A common sub-problem of slot filling is relationship extraction. Because we know which incident every section refers to, generating ground-truth relationships is trivial because we may be sure that an entity which appears in one of the sections is related to every other entity in that same section. For example, finding a KSUB and a LOCATION means that we can build a *killed_in(KSUB, LOCATION)* relationship.

Event De-duplication

Since the IBC-C dataset preserves the links between sections and incidents it may be used as a ground-truth training set for training event de-duplication models.

Question Answering

Just as in the case of CoNLL QA, the IBC-C dataset can be thought of as a QA dataset. For example, to identify KSUBS, the question (or, rather, prompt) *Identify persons killed* could be asked. More interestingly, because we have incident-level data, we can ask questions such as *Where was KSUB killed?* where *KSUB* is a slot we pre-fill with a KSUB from the instance.

	HMM			CRF 13-window			RNN 13-window		
Tag	Precision	Recall	**F1**	Precision	Recall	**F1**	Precision	Recall	**F1**
KNUM	0.63	0.86	**0.73**	0.91	0.94	**0.92**	0.90	0.85	**0.88**
INUM	0.50	0.39	**0.44**	0.95	0.93	**0.94**	0.87	0.91	**0.89**
KSUB	0.73	0.68	**0.70**	0.82	0.76	**0.79**	0.86	0.53	**0.66**
ISUB	0.00	0.00	**0.00**	0.89	0.24	**0.38**	0.80	0.06	**0.12**
LOCATION	0.75	0.70	**0.73**	0.85	0.77	**0.80**	0.86	0.70	**0.77**
DATE	0.75	0.64	**0.69**	0.75	0.64	**0.69**	0.41	0.30	**0.35**
WEAPON	0.98	0.89	**0.93**	0.98	0.90	**0.94**	0.97	0.87	**0.92**
Overall	0.57	0.53	**0.55**	0.88	0.73	**0.78**	0.74	0.57	**0.61**

Table 7.10: Results for various models on the IBC-C NER dataset.

7.2.5 Results

The first thing which strikes us is how low the ISUB scores are. The CRF returns a recall score of 0.24. At the same time, the precision is relatively high at 0.89. Low recall indicates a lot of false negative classifications - i.e. there were many injured people who were mistakenly tagged as uninjured. A high precision rate means a low false positive rate - i.e. most uninjured people were correctly tagged as uninjured. In short, the classifier was too generous with tagging irrelevant subjects as having been injured. Looking at the dataset we realize that:

- In contrast to KSUBS, words which we associate with injury such as "wounded" or "injured" are often further away from an ISUB such as in: *The attack killed 12 and injured 3 including the commander of the unit <ISUB> who was airlifted to the nearest hospital.*. KSUBs on the other hand tend to be immediately followed by a keyword such as *killed.*
- The language used to describe injuries, especially in the case of ISUBs, is much richer. Examples such as: *airlifted to hospital*, *are being taken care by medics*, *the hospital accepted a further*, *was left disabled by the attack* are all euphemisms for injuries.
- None of the three models did well on extracting ISUBs and the HMM failed completely. We were surprised to see the RNN perform relatively poorly and expected it to be able to factor in long-distance dependencies. Overall the CRF did the best out of the three.

7.3 Implementation Details

Baseline results were computed for the named entity recognition task using an 80:20 tag split across sentences (we ignore report or section boundaries). We compare three different sequence-classification models as seen in Table 7.10: a Hidden Markov Model [154], a Conditional Random Field [84], and a Elman-style Recursive Neural Network similar to the one used in [86].

For the HMM we use bigram features in combination with the current word and the current base, named entity features[1]. We trained the HMM in CRF form using LBFGS.

For the CRF we find that using bigram features and a 13-word window, across words and base named entities, gives us the best result. We train the CRF using LBFGS. All CRF training, including the HMM, was done using CRFSuite [96].

For the Elman-style recurrent network we use randomly initialized 100 dimensional word vectors as input, the network has 100 hidden units, and we use a 13-word context window again. The RNN was implemented using Theano [10]. We train the RNN using stochastic gradient descent on a single GPU (NVIDIA K80).

7.4 Related Work

Extracting information from conflict related reports has been a topic of interest at various times for both the conflict analysis, information extraction, and natural language processing communities.

The 1990s saw a series of message understanding conferences (MUCs) of which MUC-3 and MUC-4 are closely related to our work and contain reports of terrorist incidents in Central and South America. MUC data is most often used for slot filling and although MUC-3 and MUC-4 contain more slots than IBC-C they are at the same time much smaller (MUC4 contains 1,700 reports) and cannot be used for incident de-duplication.

Although various ACE, CoNNL, and TAC-KBP tasks contain within them conflict-related reports, none of them are specific to conflict and haven't been studied for conflict-related information extraction specifically.

Studies more directly related to our dataset include work by Tanev and Piskorski [135] who use pattern matching to count casualties. They report a 93% accuracy on counting the wounded. However, they have access to only 29 unique conflict events.

[1]Base named entities such as PERSON and LOCATION were found using Stanford's named entity recognizer [28].

Other non-casualty conflict-related work in the domain also suffers from a lack of data, for example, [63] only deal with 711 reports.

Despite work in the NLP and IE communities, the conflict analysis community is still reliant on datasets created by hand. These include IBC [57], ACLED [106], EDACS [19], UCDP [34], and GTD [42].

To the best of our knowledge there are no efforts to fully automate casualty counting. However, efforts using NLP/IE tools to automate incident detection do exist but their ability to de-deduplicate incidents has been called into question [145].

Three notable such efforts originating in the conflict analysis community are GDELT [73], ICEWS [97], and OEDA [121]. All three use pattern matching software such as TABARI [120] and to categorise reports using the CAMEO coding scheme [122].

7.5 Summary

We present IBC-C, a new dataset for armed conflict analysis which can be used for entity recognition, slot filling, incident de-duplication and question answering. We plan to make more information about the dataset available on `http://andrejzg.github.io/ibcc/`.

Part IV

Conclusion

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This thesis has explored question answering in its various guises. We started by describing a well known reading comprehension model (BiDAF) and conducted an ablation study of it by analyzing its performance as we removed components or replaced them with new ones. Through this we learn the importance of BiDAF's attention mechanisms and the nature of its reliance on the question representation. We re-posit the CoNLL-2003 (English) NER task as a reading comprehension problem and (1) create a new dataset called CoNLL QA and (2) extend BiDAF to this multi-span setting.

Next, we develop a new NER model which uses an ensemble-like architecture of multiple LSTMs. Using this we achieve state-of-the-art results on the CoNLL-2003 (English) NER dataset.

We also turn to the problem of answer selection. We tackle the InsuranceQA answer selection dataset using a new siamese architecture with attention to which we also concatenate global representations of the context and answer candidates.

Finally, we present a new dataset called IBC-C which amenable to being posited as a NER, relationship extraction, event deduplication and question answering dataset. We provide baseline results on it using three standard NER models.

8.2 Future Work

During the course of research for this thesis it wasn't at all clear that what I was working towards were various forms of question answering. Question answering it now seems to me is more of a format rather than necessarily a traditional NLP task. What NLP research into QA is trying to do is to place most (all?) NLP tasks under a single umbrella. For example, if answers in a QA tasks depend greatly on other sections of the text other than the their immediate context, then a QA model will have to learn to reason across co-references in the text. Similarly, if the answers are often entities, then a QA model will have to learn to reason across entities. Through QA we can relieve the complexity one faces setting up and becoming familiar with individual NLP tasks, and instead solve and benchmark on a single QA task.

To say that QA models *reason* across components of the text given a question is of course a vague statement. Although we are not quite sure what true reasoning across text means, we do know what it's not. It definitely isn't the ability to learn to match simple patterns found both in the text and question. This unfortunately it seems, is what current state-of-the-art models on SQuAD tend to do. It is well known that in many NLP tasks such as paraphrase detection, natural language inference, and any other task where two or more segments of texts must be compared, current models tend to model the bias of the annotations to achieve good results instead of, in some way, truly learning to understand the text at hand. Knowing this, it is no surprise that QA datasets and models demonstrably suffer from similar biases.

Over the past months, a series of new QA datasets have been released which try to remedy some of the deficiencies of past datasets. Unfortunately, these came too late to have been included in this thesis. Notably, datasets such as SQuAD 2.0 include simple tricks such as *context-question* pairs for which no answers exist. Whether these new datasets will help nudge existing or future models into more appropriate patterns of reasoning is yet to be seen.

Another way of nudging a model to "reason" better is to architect such that it has a so-called inductive bias for. My hunch is that the various flavours of attention mechanisms used by current QA models is the wrong way to go. They are simply too powerful at extracting relationships between context and question words when there are strong annotation biases, and too weak at handling some of the more complicated datasets which have recently been released.

A few promising avenues of future work are:

- Explore sentence-level embedding for questions in the reading comprehension task. The motivation for this is that this would prevent attention mechanisms from being able to reason across question words but instead a single embeddings of the question would have to be used. Until recently, this method would have been impractical because good sentence-level embeddings were an open problem. With the advent of new models over the past year, we now have sentence-level embedding models which outperform other simpler embeddings such as *tfidf* on *classification* tasks by a wide margin. It would be interesting to see whether this would hold in the reading comprehension setting. Conversely, perhaps encoding the context and then attending across question words could also be explored.

- There has been a recent trend to learn word and sentence embeddings under some form of supervision. For example, sentence level embeddings can be trained by training on natural language inference datasets. It is believed that this is better than fully unsupervised training because it forces the models to pickup linguistic features it other wouldn't have. It would be interesting to see (1) which pre-trained sentence or word embeddings models perform best on SQuAD and other reading comprehension datasets and (2) can reading comprehension itself be used as a supervision signal?
- IBC-C is an interesting dataset but more work needs to be done on it. It would be interesting to see whether a incident-detection system could be trained on it and then applied on a stream of live news data.

Bibliography

[1] *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017*. IEEE Computer Society, 2017.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[3] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab. Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE transactions on medical imaging*, 35(5):1313–1321, 2016.

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

[5] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[6] Y. Bachrach, A. Z. Gregoric, S. Coope, E. Tovell, B. Maksak, J. Rodriguez, and C. McMurtie. An attention mechanism for answer selection using a combined global and local view. *CoRR*, abs/1707.01378, 2017.

[7] Y. Bachrach, A. Z. Gregoric, S. Coope, E. Tovell, B. Maksak, J. Rodriguez, C. McMurtie, and M. Bordbar. An attention mechanism for neural answer selection using a combined global and local view. In *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017* [1], pages 425–432.

[8] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[10] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[11] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 31–40. ACM, 2010.

[12] E. Breck, M. Light, G. S. Mann, E. Riloff, B. Brown, P. Anand, M. Rooth, and M. Thelen. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of the workshop on Open-domain question answering-Volume 12*, pages 1–8. Association for Computational Linguistics, 2001.

[13] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[14] R. D. Burke, K. J. Hammond, V. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI magazine*, 18(2):57, 1997.

[15] W.-t. Y. M.-W. Chang and C. M. A. Pastusiak. Question answering using enhanced lexical semantic models. 2013.

[16] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630. ACM, 2009.

[17] H. L. Chieu and H. T. Ng. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[18] J. P. Chiu and E. Nichols. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*, 2015.

[19] S. Chojnacki, C. Ickler, M. Spies, and J. Wiesel. Event data on armed conflict and security: New perspectives, old challenges, and some solutions. *International Interactions*, 38(4):382–401, 2012.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[21] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[22] C. L. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365. ACM, 2001.

[23] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[24] N. Craswell. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer, 2009.

[25] T. G. Dietterich et al. Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15, 2000.

[26] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou. Applying deep learning to answer selection: A study and an open task. *arXiv preprint arXiv:1508.01585*, 2015.

[27] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou. Applying deep learning to answer selection: A study and an open task. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 813–820. IEEE, 2015.

[28] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[29] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics, 2003.

[30] M. Franco-Salvador, S. Kar, T. Solorio, and P. Rosso. Uh-prhlt at semeval-2016 task 3: Combining lexical and semantic-based features for community question answering. *arXiv preprint arXiv:1807.11584*, 2018.

[31] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156, 1996.

[32] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[33] A. Gaunt, D. Borsa, and Y. Bachrach. Training deep neural nets to aggregate crowdsourced responses. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence. AUAI Press*, page 242251, 2016.

[34] N. P. Gleditsch, P. Wallensteen, M. Eriksson, M. Sollenberg, and H. Strand. Armed conflict 1946-2001: A new dataset. *Journal of peace research*, 39(5):615–637, 2002.

[35] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[36] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe. Deep convolutional ranking for multilabel image annotation. *arXiv preprint arXiv:1312.4894*, 2013.

[37] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[38] B. F. Green Jr, A. K. Wolf, C. Chomsky, and K. Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

[39] A. Z. Gregoric, Y. Bachrach, and S. Coope. Named entity recognition with parallel recurrent neural networks. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 69–74. Association for Computational Linguistics, 2018.

[40] A. Z. Gregoric, Y. Bachrach, P. Minkovsky, S. Coope, and B. Maksak. Neural named entity recognition using a self-attention mechanism. In *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017* [1], pages 652–656.

[41] A. Z. Gregoric, Z. Luo, and B. Veyhe. IBC-C: A dataset for armed conflict analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics, 2016.

[42] GTD. Global terrorism database. `http://www.start.umd.edu/gtd`, 2015. (Accessed on 02/23/2016).

[43] Y. Guan, Z. Yuan, G. Sun, and J. Cong. Fpga-based accelerator for long short-term memory recurrent neural networks. In *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, pages 629–634. IEEE, 2017.

[44] H. He, K. Gimpel, and J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, 2015.

[45] H. He and J. Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, 2016.

[46] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[47] M. H.-R. Hicks, H. Dardagan, G. G. Serdán, P. M. Bagnall, J. A. Sloboda, and M. Spagat. Violent deaths of iraqi civilians, 2003–2008: analysis by perpetrator, weapon, time, and location. *PLoS Med*, 8(2):e1000415, 2011.

[48] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 241–248. ACM, 2016.

[49] F. Hill, A. Bordes, S. Chopra, and J. Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

[50] L. Hirschman and R. Gaizauskas. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300, 2001.

[51] L. Hirschman, M. Light, E. Breck, and J. D. Burger. Deep read: A reading comprehension system. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 325–332. Association for Computational Linguistics, 1999.

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

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

[54] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339, 2018.

[55] H.-Y. Huang, C. Zhu, Y. Shen, and W. Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*, 2017.

[56] Z. Huang, W. Xu, and K. Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[57] IBC. Iraq body count. `https://www.iraqbodycount.org/database/`, 2016. (Accessed on 02/23/2016).

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

[59] V. Jijkoun and M. De Rijke. Answer selection in a multi-stream open domain question answering system. In *ECIR*, volume 2997, pages 99–111. Springer, 2004.

[60] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[61] K. Kandasamy, Y. Bachrach, R. Tomioka, D. Tarlow, and D. Carter. Batch policy gradient methods for improving neural conversation models. *arXiv preprint arXiv:1702.03334*, 2017.

[62] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[63] G. King and W. Lowe. An automated information extraction tool for international conflict data with performance as good as human coders: A rare events evaluation design. *International Organization*, 57(03):617–642, 2003.

[64] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[65] J. Ko, L. Si, and E. Nyberg. A probabilistic framework for answer selection in question answering. In *HLT-NAACL*, pages 524–531, 2007.

[66] O. Kolomiyets and M.-F. Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434, 2011.

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[68] C. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262, 2001.

[69] T. M. Lai, T. Bui, and S. Li. A review on deep learning techniques applied to answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2132–2144, 2018.

[70] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[71] C. T. Lee, E. M. Rodrigues, G. Kazai, N. Milic-Frayling, and A. Ignjatovic. Model for voter scoring and best answer selection in community q&a services. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 116–123. IEEE, 2009.

[72] K. Lee, S. Salant, T. Kwiatkowski, A. Parikh, D. Das, and J. Berant. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*, 2016.

[73] K. Leetaru and P. A. Schrodt. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA Annual Convention*, volume 2. Citeseer, 2013.

[74] W. G. Lehnert. The process of question answering. Technical report, YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE, 1977.

[75] J. L. Leidner, T. Dalmas, B. Webber, J. Bos, and C. Grover. Automatic multi-layer corpus annotation for evaluation question answering methods: Cbc4kids. In *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) at EACL 2003*, 2003.

[76] Y. Lewenberg, Y. Bachrach, S. Shankar, and A. Criminisi. Predicting personal traits from facial images using convolutional neural networks augmented with facial landmark information. In *IJCAI*, pages 1676–1682, 2016.

[77] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015.

[78] X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[79] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics, 2009.

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

[81] G. Luo, X. Huang, C.-Y. Lin, and Z. Nie. Joint named entity recognition and disambiguation. In *Proc. EMNLP*, pages 879–880, 2015.

[82] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[83] X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.

[84] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

[85] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

[86] G. Mesnil, X. He, L. Deng, and Y. Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, pages 3771–3775, 2013.

[87] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753, 2014.

[88] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[89] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *hlt-Naacl*, volume 13, pages 746–751, 2013.

[90] S. Min, M. Seo, and H. Hajishirzi. Question answering through transfer learning from large fine-grained supervision data. *arXiv preprint arXiv:1702.02171*, 2017.

[91] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[92] P. K. Mudrakarta, A. Taly, M. Sundararajan, and K. Dhamdhere. Did the model understand the question? *arXiv preprint arXiv:1805.05492*, 2018.

[93] J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792, 2016.

[94] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[95] H. T. Ng, L. H. Teo, and J. L. P. Kwan. A machine learning approach to answering questions for reading comprehension tests. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 124–132. Association for Computational Linguistics, 2000.

[96] N. Okazaki. Crfsuite: a fast implementation of conditional random fields (crfs), 2007. (Accessed on 02/23/2016).

[97] S. P. Obrien. Crisis early warning and decision support: Contemporary approaches and thoughts on future research. *International Studies Review*, 12(1):87–104, 2010.

[98] A. Passos, V. Kumar, and A. McCallum. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*, 2014.

[99] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[100] C. S. Perone, R. Silveira, and T. S. Paula. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*, 2018.

[101] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

[102] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[103] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.

[104] P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

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

[106] C. Raleigh, A. Linke, H. Hegre, and J. Karlsen. Introducing acled: An armed conflict location and event dataset special data feature. *Journal of peace research*, 47(5):651–660, 2010.

[107] J. Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142, 2003.

[108] J. Rao, H. He, and J. Lin. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM, 2016.

[109] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009.

[110] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 41–47. Association for Computational Linguistics, 2002.

[111] N. Reimers and I. Gurevych. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861*, 2017.

[112] Y. Ren, Y. Du, and D. Wang. Tackling adversarial examples in qa via answer sentence selection. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 31–36, 2018.

[113] M. T. Ribeiro, S. Singh, and C. Guestrin. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 856–865, 2018.

[114] M. Richardson, C. J. Burges, and E. Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, 2013.

[115] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[116] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[117] C. d. Santos, M. Tan, B. Xiang, and B. Zhou. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*, 2016.

[118] M. Sato, H. Shindo, I. Yamada, and Y. Matsumoto. Segment-level neural conditional random fields for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 97–102, 2017.

[119] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[120] P. A. Schrodt. Automated coding of international event data using sparse parsing techniques. In *annual meeting of the International Studies Association, Chicago*, 2001.

[121] P. A. Schrodt. Open event data alliance (oeda), 2016. (Accessed on 02/23/2016).

[122] P. A. Schrodt, O. Yilmaz, D. J. Gerner, and D. Hermreck. The cameo (conflict and mediation event observations) actor coding framework. In *2008 Annual Meeting of the International Studies Association*, 2008.

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

[124] B. Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107. Association for Computational Linguistics, 2004.

[125] A. Severyn and A. Moschitti. Automatic feature engineering for answer selection and extraction. In *EMNLP*, volume 13, pages 458–467, 2013.

[126] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM, 2015.

[127] L. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil. Generating long and diverse responses with neural conversation models. *arXiv preprint arXiv:1701.03185*, 2017.

[128] R. F. Simmons. Answering english questions by computer: a survey. *Communications of the ACM*, 8(1):53–70, 1965.

[129] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank.

In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[130] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[131] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[132] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[133] M. Tan, C. dos Santos, B. Xiang, and B. Zhou. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[134] M. Tan, C. d. Santos, B. Xiang, and B. Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[135] H. Tanev, J. Piskorski, and M. Atkinson. Real-time news event extraction for global crisis monitoring. In *Natural Language and Information Systems*, pages 207–218. Springer, 2008.

[136] Y. Tay, L. A. Tuan, and S. C. Hui. Multi-cast attention networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2299–2308. ACM, 2018.

[137] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

[138] A. Trischler, Z. Ye, X. Yuan, J. He, P. Bachman, and K. Suleman. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884*, 2016.

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

[140] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal. On orthogonality and learning recurrent networks with long term dependencies. *arXiv preprint arXiv:1702.00071*, 2017.

[141] S. Wadhwa, K. R. Chandu, and E. Nyberg. Comparative analysis of neural qa models on squad. *arXiv preprint arXiv:1806.06972*, 2018.

[142] M. Wang and C. D. Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1164–1172. Association for Computational Linguistics, 2010.

[143] M. Wang, N. A. Smith, and T. Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[144] Z. Wang, W. Hamza, and R. Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[145] N. Weller and K. McCubbins. Open event data alliance (oeda)raining on the parade: Some cautions regarding the global database of events, language and tone dataset, 2014. (Accessed on 05/18/2016).

[146] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[147] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[148] C. Xiong, V. Zhong, and R. Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[149] J. Xu, A. Licuanan, J. May, S. Miller, and R. M. Weischedel. Trec 2002 qa at bbn: Answer selection and confidence estimation. In *TREC*, volume 54, page 90, 2002.

[150] Y. Yang, W.-t. Yih, and C. Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, 2015.

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

[152] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[153] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[154] G. Zhou and J. Su. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics, 2002.

[155] D. Zhu, S. Shen, X.-Y. Dai, and J. Chen. Going wider: Recurrent neural network with parallel cells. *arXiv preprint arXiv:1705.01346*, 2017.