On the Integration of Conceptual Hierarchies with Deep Learning for Explainable Open-Domain Question Answering

Harish Tayyar Madabushi

A Thesis Submitted to the University of Birmingham for the Degree of
Doctor of Philosophy

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
May 2019

Abstract

Question Answering, with its potential to make human-computer interactions more intuitive, has had a revival in recent years with the influx of deep learning methods into natural language processing and the simultaneous adoption of personal assistants such as Siri, Google Now, and Alexa. Unfortunately, Question Classification, an essential element of question answering, which classifies questions based on the class of the expected answer had been overlooked. Although the task of question classification was explicitly developed for use in question answering systems, the more advanced task of question classification, which classifies questions into between fifty and a hundred question classes, had developed into independent tasks with no application in question answering.

The work presented in this thesis bridges this gap by making use of fine-grained question classification for answer selection, arguably the most challenging subtask of question answering, and hence the defacto standard of measure of its performance on question answering. The use of question classification in a downstream task required significant improvement to question classification, which was achieved in this work by integrating linguistic information and deep learning through what we call Types, a novel method of representing Concepts.

Our work on a purely rule-based system for fine-grained Question Classification using Types achieved an accuracy of 97.2%, close to a 6 point improvement over the previous state of the art and has remained state of the art in question classification for over two years. The integration of these question classes and a deep learning model for Answer Selection resulted in MRR and MAP scores which outperform the current state of the art by between 3 and 5 points on both versions of a standard test set.

This work is dedicated to my parents, Usha Tayyar Madabushi and
Narasimhan Tayyar Madabushi. None of this would have been possible without your vision,
understanding, sacrifice, support and love.
Thank you.

Acknowledgements

This work would have been impossible without the support and guidance of my supervisors Dr Mark Lee and Professor John Barnden. Dr Lee, through a careful combination of flexibility and discipline, provided support when necessary and pushed me to discover solutions for myself, when he knew I could, even when I myself wasn't sure. I see Mark more as a mentor and friend than solely as a supervisor. Professor Barnden consistently provided detailed feedback on all aspects of my work which was instrumental in refining my thoughts, ideas and writing, and I thank him for supporting my work even after his retirement.

The additional guidance and support by Dr David Parker and Dr Manfred Kerber, my thesis group, was invaluable in the completion of this work. Dr Parker always made himself available for which I am grateful as I am for Dr Kerber's encouragement.

Others at the University of Birmingham have played an important role in helping me complete this work, and I would like to thank them, specifically Dr Phillip Smith, Dr Mohab Elkaref, and Dr Tomáš Jakl for the insightful conversations, and Mark Buhagiar for collaborating with me on my first publication.

My interest in science is the direct result of a truly wonderful teacher at my school, Rishi Valley, the late Mr Jayant Tengshe. Through interesting discussions, chess, physics and volleyball he pushed me, over six years, to explore science in a way I had not previously imagined. I am incredibly grateful for this and regret not telling him so when I still had the chance: thank you.

This general interest in science was refined during my Masters at the Chennai Mathematical Institute specifically through the guidance of Professor Madhavan Mukund and Professor Vinay Vishwanathan. Both Professor Mukund and Professor Vishwanathan have been my mentors for over a decade and have been instrumental in shaping my career and my thinking. Thank you for all your help and support.

I would like to thank Danai Papachristopoulou for her incredible patience, support, affection and her belief in me. Thank you for all that you've done for me, for showing me there is more to life than work and for making me want to push myself.

Publications

1. **Harish Tayyar Madabushi**, Mark Lee, and John Barnden *Integrating Question Classification and Deep Learning for improved Answer Selection.* Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, USA (COLING2018) (Tayyar Madabushi et al., 2018).
2. **Harish Tayyar Madabushi**, and Mark Lee *High Accuracy Rule-based Question Classification using Question Syntax and Semantics.* Proceedings of the 26th International Conference on Computational Linguistics, Osaka, Japan (COLING2016) (Tayyar Madabushi and Lee, 2016).
3. **Harish Tayyar Madabushi**, Mark Buhagiar, and Mark Lee *UoB-UK at SemEval-2016 Task 1: A Flexible and Extendable System for Semantic Text Similarity using Types, Surprise and Phrase Linking.* Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016) (Tayyar Madabushi et al., 2016).

Table of contents

List of figures

List of tables

1 Introduction and Motivation

The indexed surface web is estimated to contain around 4.68 billion pages (de Kunder, 2008, 2015). Admittedly the vast majority of this information might have little relevance to a particular individual, however, given that just the popular AI journals publish over five thousand articles a year (SCImago, 2016), it is safe to assume that individuals have access to vastly more information than they can sift through.

Existing solutions to this information overload range from search engines to (human) personal assistants. With no indications that this explosive growth in information is likely to diminish, the need for an alternate solution is more urgent than ever before (Etzioni, 2011).

1.1 The Current Limitations of Web Search

Web Search, in its current form, is tremendously useful *in specific contexts*. Finding information on the web was initially accomplished by use of directories such as Yahoo! As the number of web pages grew, this soon became impossible, and search engines became the preferred solution. Web search continues to be good at providing a list of web pages that contain specific information. Until recently, searching for "pizza delivery" did not provide one with a list of pizza outlets that are currently open and deliver to the user's location (D'Assisi, 2016). This has radically changed in the last couple of years with local and geotargeted listings becoming the norm. Similarly, search engines now provide more emphasis on recency and context rather than just relevance.

Despite these advances, searching for information on the web continues to be a significantly different experience from asking someone for the same information. Bridging this gap in experience has the potential to save users an incredible amount of time while simultaneously increasing the number of people interacting with content. A more intuitive way of exploring information can also help businesses augment their support teams so saving money while providing 24-hour support.

1.2 Beyond Search

Several competing approaches have the potential to provide a more exciting and intuitive way of interacting with web-data than search does. The most prominent amongst them is Chatbots (Dale, 2016). Chatbots provide a method of interaction that is similar to everyday conversations. While these systems can provide a more intuitive way of interacting with web-information, in the context of accessing information, it is reasonable to assume that the majority of interactions between users and chatbots are in the form of questions from users that are to be answered by the chatbot. However, chatbots additionally require techniques of extending conversations with users, motivating further interaction and other elements of human-human talk. To avoid these additional research requirements of Chatbots, this thesis focuses on Question Answering, which is arguably a prerequisite for effective chatbots. This thesis describes experimentation with and methods of improving Question Answering (QA) systems.

The significant improvement in the accuracy of QA achieved through methods described in this thesis relies on the use of linguistic and cognitive information that had, until now, been thrown by the wayside due to over-reliance on pure machine learning methods. We hope that these improved results show that linguistic analysis continues to play an important and impactful role in NLP in general and QA in particular, and will, even if in a small way, reverse the trend of exclusively using deep learning methods.

1.3 The Shift to Statistical Methods in NLP

In December 2013 Tomas Mikolov presented the now famous paper “Distributed representations of words and phrases and their compositionality” (Mikolov et al., 2013a). The use of context to represent words was not new (Firth, 1957), and neither was the use of vectors to represent the meaning of words (Bengio et al., 2003; Levy et al., 2015). This work, however, additionally described an analogy task and the resultant embeddings, captivatingly called ‘word2vec’ which were publicly released by Google [1]. The technical advantage of word2vec is speed. However, it was the ease of availability and the media attention that drove adoption. Google released pre-trained vectors, trained on a part of the Google News dataset, containing about 100 billion words and very soon word vectors were being used in NLP tasks from Semantic Text Similarity (Section 2.2) to Answer Selection (Chapter 5).

Two years later, at the same conference, Neural Information Processing Systems, researchers from Oxford and DeepMind, by then a subsidiary of Google, presented a paper titled “Teaching Machines to Read and Comprehend” (Hermann et al., 2015), in which they introduced the idea of “attention”, from machine vision, to vastly improve accuracy in reading comprehension. Their analysis included results using traditional models such as frame-semantics, exploitation of word embeddings, and deep learning models with attention.

As can be seen from Table 1.1, a reproduction of Hermann et al. (2015)’s results, deep learning models performed vastly better than the other two models. These works, in addition to the deep learning methods not needing annotation, led research in nearly every subfield of NLP to quickly shift to deep learning models that used pre-trained word embeddings as input and consisted of CNNs, RNNs, LSTMs, several layers, and attention. This not only led to

[1] https://code.google.com/archive/p/word2vec/

	Validation	Test
Maximum frequency	30.5	33.2
Exclusive frequency	36.6	39.3
Frame-semantic model	36.3	40.2
Word distance model	50.5	50.9
Deep LSTM Reader	55.0	57.0
Uniform Reader	39.0	39.4
Attentive Reader	61.6	63.0
Impatient Reader	61.8	63.8

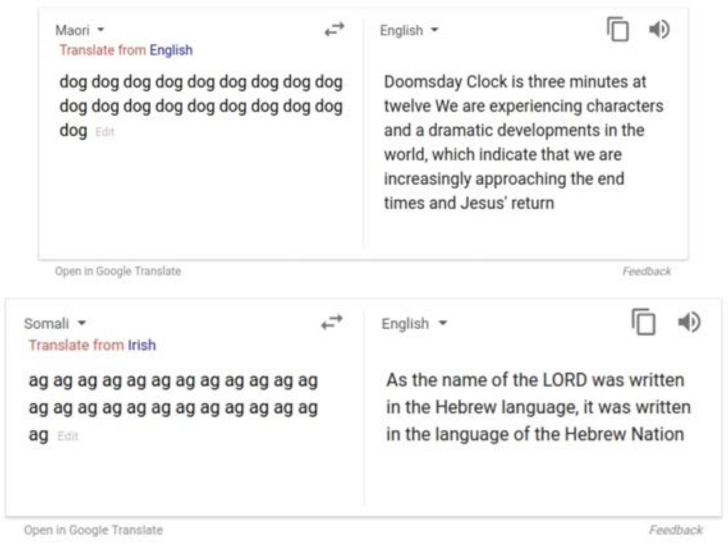
Table 1.1 Accuracy of Reading Comprehension of models and benchmarks on the CNN dataset (Hermann et al., 2015)

the lack of exploitation of existing research into linguistics but also resulted in researchers being faced with shortcomings inherent to Learning models.

1.4 Deep Learning - The Problems

The significant advantages provided by purely deep learning methods, including their effectiveness (Pereira et al., 2009), come not without their drawbacks. The ability of deep neural networks to approximate extremely complicated functions has also meant that error analysis has become a near impossibility. The popularity of deep learning has translated into its ubiquitous usage across domains, including those that require rigid error analysis, such as medicine and autonomous cars and recent projects have started to emphasise the need for these models to be explainable (Gunning, 2018; Wierzynski, 2018). The problem of neural networks generating bizarre results is exemplified by Google Translate as shown in Figure 1.1 [1]. A similar difficulty in performing an error analysis is explored in the context of a different task in Section 2.1.6.

[1]Source: https://motherboard.vice.com/en_us/article/j5npeg/why-is-google-translate-spitting-out-sinister-religious-prophecies



dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog dog

Doomsday Clock is three minutes at twelve We are experiencing characters and a dramatic developments in the world, which indicate that we are increasingly approaching the end times and Jesus' return

ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag ag

As the name of the LORD was written in the Hebrew language, it was written in the language of the Hebrew Nation

Fig. 1.1 Google Translate Acting Bizarre (2018).

Additionally, there is reason to believe that the improvements in accuracy provided by deep learning systems are plateauing. A look at state of the art in Answer Selection [1] shows significant improvements in accuracy between 2013 and 2016, followed by almost two years of stagnation - until work done as part of this project improved on the state of the art in 2018.

1.5 Integrating Classical Natural Language Processing with Deep Learning

Methods that exploit the systematic analysis of language, its semantics and syntactic structure are referred to, especially in this work, as *traditional methods in natural language processing*. These methods do make use of machine learning; however, they do not do so without linguistic information in the way that deep learning methods do. They often rely on carefully selected features, which are then fed into machine learning models.

The superior performance of deep learning models shifted the emphasis away from the use of classical methods; however, this work shows that the use of traditional methods to

[1] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

engineer *some* features, while simultaneously making use of the implicit feature extraction provided by deep learning models, provides a method of achieving a significant increase in accuracy of some NLP tasks.

This break from *pure* machine learning methods implicitly provides the work presented in this thesis with additional advantages, namely: a) better error analysis, and b) access to decades of work into language structure, semantics, language learning and cognitive processes associated with language.

1.6 Research Questions

The preceding sections have discussed the recent resurgence of deep learning techniques and the consequent gains achieved across several NLP tasks. While most current state of the art techniques utilise deep learning, it is not clear if this implies that traditional methods are unable to match the achievements of deep learning methods. This leads to the first research question explored in this work:

- **Research Question 1:** Has the success of deep learning methods made it impossible to improve upon the state of art of various NLP tasks without the use of deep learning?

The answer to this question will have far reaching implications to research in NLP as it will determine if it is worth pursuing research that does not exploit deep learning. If traditional methods cannot achieve the performance of deep learning models, this will imply that computational linguists are better off exploring methods of creating features for statistical models rather than exploring the nature of language.

A related line of exploration is based on the fact that, unlike in other fields, such as vision, the impact of deep learning on NLP has been limited[1]. This is not to say that deep learning models have not had an impact on NLP tasks, but that the improvement in performance has not been as significant as in other domains. In addition the biggest contributor to NLP has

[1]https://www.reddit.com/r/MachineLearning/comments/2fxi6v/ama_michael_i_jordan/

been the introduction of Distributed Word Embeddings rather than the exploitation of the generalisation capabilities of deep learning models (Manning, 2015), which leads us to our second research question:

- **Research Question 2:** Given that the success of deep learning methods lies in their ability to abstract learning, and that learning abstraction is not where the gains in NLP stem from, are there other forms of generalisation that might be more suitable to NLP?

Despite the interest in finding such an alternative, this work does not fail to recognise the contribution and potential of deep learning models. In the four years since Manning (2015) wrote his piece, deep learning models have continued to improve upon the state of the art in NLP. Given this, the final line of enquiry explores the possibility of bringing together these methods and decades of work that has gone into traditional NLP research, which leads to the final research question:

- **Research Question 3:** How can features discovered through the analysis of language and language structure be fed into deep learning models without fundamental changes to those models?

Section 6.2 of this work's Conclusions (Chapter 6) discusses how this work addresses these questions and what that implies in terms of the contributions of this work.

1.7 Novel Methods and Contribution

This work has resulted in the creation of hundreds of rules that outperform all machine learning methods for Question Classification - a task that classifies questions based on the type of answer expected (see Section 2.4). The generation of these rules was made possible by using the idea of concepts, which are generalisations or abstractions that allow the use of previous experience in new situations (see Chapter 3). The resultant Question Classification system achieved an accuracy of 97.2%, close to a 6 point improvement over the previous state of the art of 91.6% (Tayyar Madabushi and Lee, 2016). This system has also been made available publicly through an API.[1].

These question classes were subsequently integrated into a deep learning system for Answer Selection, which outperformed the then state of the art by between 3 and 5 points on both versions of a standard test set. These results were presented at COLING 2018 (Tayyar Madabushi et al., 2018) (see Chapter 5).

1.8 Thesis Structure

As discussed in this chapter, deep learning methods have had a tremendous impact on research into natural language processing during the course of this project (2014 to 2018). However, this work provides novel methods of integrating information obtained through traditional NLP methods into deep learning to improve accuracy and simultaneously make deep learning models more transparent.

Chapter 2 explores research related to this work, including related work in the field of QA, the different subproblems in QA, and the approaches to these problems (Section 2.1). The same chapter then provides an overview of work related to tasks that this work explores such as Semantic Text Similarity (STS) (Section 2.2), Question Classification (Section 2.3) and Answer Selection (Section 2.4).

Chapter 3 provides the theoretical foundation for the work presented in this work. It explores the idea of Concepts, their utility, their implementation through Types before then providing an empirical evaluation using the STS task.

Chapter 4 and Chapter 5 detail the central contributions of this work with regard to the tasks of Question Classification and Answer Selection respectively.

Finally, Chapter 6 provides our conclusions and ideas on how the work presented in this thesis can be extended in future projects.

[1]Available at http://www.harishmadabushi.com/research/questionclassification/

2

Related Work

This Chapter explores recent research into the field of automated Question Answering. There is significant variation in the kind of questions that various systems attempt to answer and the methods they use to do so. This Chapter starts off by providing an overview of the topic before providing a more in-depth analysis of specific methods related to the work presented in this thesis, namely Question Classification and Answer Selection.

2.1 Question Answering - An Overview

Question Answering is a simple problem to define and yet has remained the focus of research for decades (Simmons, 1965). This complexity has led to the subdivision of the problem into several sub-problems, each of which has been addressed using a variety of methods. An exploration of various surveys of automated QA shows that researchers do not agree on either the criteria for classification or the classification within any given criterion. This section provides an overview of some of the (subjectively) more interesting criteria and the classification associated with each criterion.

Before exploring these criteria, we discuss some of the surveys on Question Answering systems starting with the earliest by Simmons (1970). Simmons (1970) explores five Question Answering systems that were state of the art in the late 60s, including one capable of making geometric inferences. These early systems were limited to a subset of English and often relied on basic template matching (Section 2.1.4) and trivial semantic distance based on thesaurus word frequencies. The next survey into Question Answering by Androutsopoulos et al. (1995) focused on how natural language questions can be answered using database systems. The next survey was that by Hirschman and Gaizauskas (2001) which coincided with their introduction of the TREC Question Answering track (Voorhees, 2001b), data from which is still used today.

The introduction of the TREC QA track reignited interest in Question Answering which was captured in surveys by Andrenucci and Sneiders (2005), which continued to focus on research into converting natural language into structured data, and that by Prager (2007). Prager (2007)'s work explored different kinds of Question Answering (i.e. factoid, list, . . .) which we discuss in more detail in Section 2.1.2.

In line with the computer science zeitgeist at the time, research in to Question Answering then focused on methods involving Information Retrieval, such as work by Mollá and Vicedo (2007) and Kolomiyets and Moens (2011). Simultaneously, there was an emphasis on domain specific Question Answering capture in the survey by Athenikos and Han (2010).

Finally, the survey by Yao (2014) explores various features which are useful in Question Answering, a popular direction of research before the popularity of deep learning methods. The latest available survey on Question Answering is that by Mishra and Jain (2016) which includes an overview of deep learning techniques which we discuss in Section 2.5.

2.1.1 Domains

The specific domain that a QA system can answer questions regarding is often the most general criterion for classification. Closed domain QA systems answer questions related to specific domains such as medical questions (Athenikos and Han, 2010). Open domain

QA systems, such as the one presented in this work, on the other hand, impose no such restrictions.

The need for this distinction arose from the fact that domain-specific data often requires domain specialisation that a generic system might not have or require. Popularly used tools, such as pre-trained word vectors, are trained on generic text making them less effective in a domain-specific context.

2.1.2 Types of Questions

Questions can be of different types, for example, the question ‘What is the name of the actress who has won the most Oscars?’, requires the answer to be a fact and hence is referred to as a factoid question. The question ‘What is the meaning of life?’ requires a longer answer and so is called a descriptive question. Table 2.1 provides a list of typical classes of questions and references to the subjectively more striking works associated with each.

2.1.3 Answer Generation

A QA system might either extract an answer from existing text or generate text containing the answer. The first method is more popular as it ignores the additional task of, and consequently the additional possibility of errors in, generating coherent and grammatically correct sentences. The latter, more difficult method is called Generative Question answering and requires specialised natural language generation algorithms.

Methods exploiting Answer Selection (Bian et al., 2017; Yu et al., 2014) are examples of those that extract answers from existing text. Work by Yin et al. (2016) into neural generative Question Answering is an example of a popular method in generative Question Answering where the task is treated as a sequence to sequence problem that is solved using neural networks, in a method similar to that used in machine translation.

Type	Description	Related Work
Factoid	Expected answers are short, usually a single word of phrase.	(Iyyer et al., 2014)
List	Expected answers are a list or a part a list. Sometimes considered to be Factoid QA.	(Schone et al., 2005)
Yes or No	Questions require a simple yes or no as an answer. Sometimes considered to be Factoid QA.	(Srihari and Li, 2000)
Multiple Choice	Multiple choice questions, answers are one or more of several options. Sometimes considered to be Factoid QA.	(Jansen et al., 2016)
Descriptive	Questions require detailed explanations as answers.	(Tan et al., 2015)

Table 2.1 The Different Types of Questions.

2.1.4 Approaches to Question Answering

Each of the above kind of questions and answer generation methods is addressed using several approaches. Some approaches are naturally better suited for certain kinds of questions; however, the classification provided in this section ignores this and presents all possible approaches to solving Question Answering.

Template Matching

Early QA systems (Simmons, 1970) relied on creating templates for questions and extracting answers based on related answer templates. For example, the question "Who is the prime minister of the United Kingdom?" can be generalised using the template *who be <role> of*

<*entity*> (Hovy et al., 2000). Similar patterns presented by Hovy et al. (2000) are listed in Table 2.2.

Question	**Question templates**
Who was Johnny Mathis' high school track coach? Who was Lincoln's Secretary of State?	who be <entity>'s <role>
Who was President of Turkmenistan in 1994? Who is the composer of Eugene Onegin? Who is the CEO of General Electric?	who be <role> of <entity>
Actual answers	**Answer templates**
Lou Vasquez, track coach of ... and Johnny Mathis	<person>, <role> of <entity>
Signed Saparmurad Turkmenbachy [Niyazov], president of Turkmenistan	<person> <role-title*> of <entity>
... Turkmenistan's President Saparmurad Niyazov...	<entity>'s <role> <person>
... in Tchaikovsky's Eugene Onegin...	<person>'s <entity>
Mr. Jack Welch, GE chairman...	<role-title> <person>... <entity> <role>
... Chairman John Welch said ... GE's	<subject>|<psv object> of related role-verb

Table 2.2 Question and Answer Patterns (Hovy et al., 2000, p. 6)

Some templates were manually defined, like those described above, and others were learned using large corpora including the web (Wu et al., 2005). Templates and patterns have been used in a variety of tasks, most notably by Hearst (Hearst, 1992) for the discovery of hyponyms. This work by Hearst was the motivation behind the template based Question Classification system developed by the author and detailed in Chapter 4.

Matching templates has been attempted by use of machine learning techniques by researchers including Li et al. (2010), who use a semi-supervised method of matching while Yih et al. (2013) provide a method dependent on word alignment. Several of these methods make use of Hearst patterns.

Template based systems, often criticised for their inability to adapt to new domains or language, have the advantage of being explainable. Chapter 4 presents a novel method of using templates on the parse trees of questions for Question Classification, which achieved state of the art results.

Question Classification and Answer Selection

Question Classification consists of classifying questions based on the class of the expected answer. For example, the question "Which year did the first world war begin?" can be classified as "Numeric Year" as the answer is a year. Answer selection is the related, yet independent task, of selecting, from a list of sentences some of which contain the answer to a given question and some that do not, that subset which does. As this work follows this model, these methods are explored in greater detail in Sections 2.3 and 2.4.

Web Redundency

Work by Brill et al. (2002) showed that redundant information in web results is a powerful signal in determining the answer to a given question. They proposed a simple method of translating a question into a web query before then using redundant n-grams in web results to extract the answer. They also showed that mapping the type of n-gram to the question class improved the results dramatically, as it cut down on noise inherent to web data. Brill et al. (2002), however, used an extremely simplistic Question Classification system. As an example of this method, given the question "Where is the Louvre Museum located?", they query a web search engine with the queries "the Louvre Museum is located", "the Louvre

Museum is in", "the Louvre Museum is near", and "the Louvre Museum is". The results are mined for n-grams, which are pruned based on the type of the question. The results are then tiled to combine smaller phrases into larger ones (e.g. phrases A, B, C and B, C, D are combined to form A, B, C, D) and sorted by frequency to extract the answer.

Roussinov et al. (2005) subsequently used a similar method of exploiting redundancy in web results for QA. Recently, Tsai et al. (2015) showed how significant improvements in search engine technologies have significantly improved this system. They also use a simplistic version of WordNet hierarchies similar to Types (Chapter 3).

Chapter 6 presents one possible method of combining the work presented in this thesis with a system that makes use of a web search engine and redundancy to create an end to end Question Answering system capable of competing with the very best QA systems in the market today.

Tree and Graph Matching

As parsing technologies improved, it was natural to look to the structure of questions and sentences which possibly contained the answer (called candidate sentences). The primary objective of these methods was to use only the structure and match the structures of the question and the sentence using different metrics such as mutual information (Lin and Pantel, 2001) or alignment (Cui et al., 2005).

At first glance, the idea of using only structure to find answers seems misguided. However, research into pattern grammar (Hunston and Francis, 2000) has shown that words that appear in similar structures tend to have the same meaning. Other work in this regard involved the use of different methods of finding similarities such as kernel-based classifiers (Severyn and Moschitti, 2013).

Some of these methods involve merging the structure of the question and existing structured databases such as those described previously in this Section (Yih et al., 2014). An interesting extension is work by He et al. (2014), who use Markov Logic Networks, as described by Richardson and Domingos (2006), for reasoning.

Matching over Existing Structured Databases

Querying databases for answers to user questions is arguably the easiest method of automatically answering questions. Unfortunately, translating information and questions from natural language to database data and queries has been fraught with difficulty.

The limited amount of information available in database form stifled initial attempts to use database systems for QA. This changed with the introduction of Freebase and DBpedia, both Linked Data structures. Linked Data is a method of interlinking data to include additional semantic information. Freebase was a sizeable structured knowledgebase which accumulated information from several sources including Wikipedia and online collaboration (Bollacker et al., 2008), which was acquired by Google and subsequently shut down (Google+, 2015). DBpedia (Auer et al., 2007) is a project aimed at extracting and storing structured information from Wikipedia, primarily by use of Wikipedia's infoboxes.

The introduction of the Free917 dataset, a set of 917 questions associated with information available on Freebase shifted the focus of database reliant question answering to Freebase. Yao and Van Durme (2014) have shown that methods using Freebase can often outperform some sophisticated approaches while Berant et al. (2013) have shown how question-answer pairs can be used to boost semantic parsing. Fader et al. (2014) have shown how data from Freebase can be combined with automatically extracted tuples to improve the quality of Question Answering systems. Several other researchers have developed systems based on Freebase (Bao et al., 2014; Berant and Liang, 2014; Cai and Yates, 2013; Kwiatkowski et al., 2013)

There have been several attempts at creating Question Answering Systems using DBPedia (Walter et al., 2012; Yao et al., 2012). While most of these methods first create triples

(Sections 2.1.4) before then matching them, Unger et al. (2012) have proposed a method that extracts information from DBpedia by going beyond the representation of questions as triplets. Fader et al. (2014) provide an overview of systems that use similar KBs.

Work by He et al. (2014) has explored the use of First-order Logic in finding answers within Linked Data, a method that has since come to be known as Logical Forms over Linked Data. Other recent work in this area has revolved around an attempt at converting natural language to a logical form (Yang et al., 2014).

Triples

Another method of representing and reasoning over data for QA is to store information in the form of Triples. For example, the question "Who wrote The Neverending Story?" can be represented as:
<[person,organization], wrote, The Neverending Story>
The system would then use similarity metrics to match the relevant sub-graph from a Linked Data repository (Unger et al., 2012). The most recent and influential work in this regard has been by Fader et al. (2011). In their paper, they describe ReVerb[1], a system that provides a method of extracting relevant triplets from natural language text. Fader et al. (2013) subsequently described a method of mapping Open Domain Questions onto the relations extracted by ReVerb. Fader et al. (2014) then combined these two methods to implement a Question Answering system that was, at the time, state of the art.

There have been others who have represented information as tuples, such as Yih et al. (2014), who use Convolutional Neural Network models to find similarities between entities and relations.

[1]ReVerb is available at http://reverb.cs.washington.edu/

2.1.5 Community Question Answering

Community QA (cQA) refers to the large number of forums and websites dedicated to allowing users, or the *community*, to answer questions posed by other users. Matching new questions to existing questions on such websites has recently gained tremendous popularity (Nakov et al., 2017, 2016) with cQA websites investing heavily in research into the field. Additionally, companies have started using similar techniques to provide automated chat support by matching questions by users on a chat system to existing FAQs.

2.1.6 Visual Question Answering

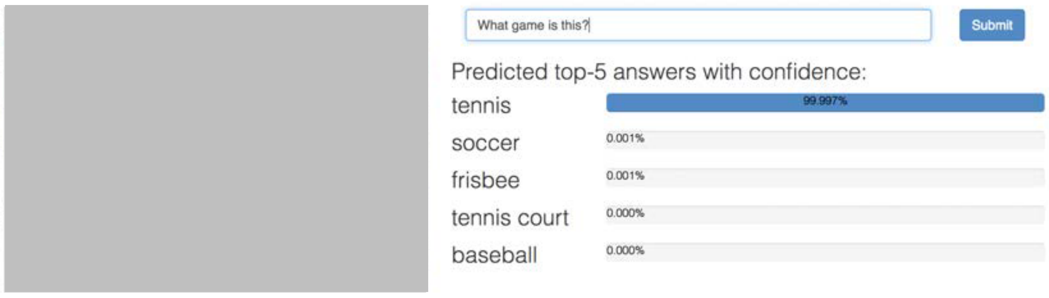
Visual Question Answering (vQA) (Antol et al., 2015) is the task of automatically answering natural language questions about an image and so requires a multidisciplinary approach to its solution. Despite results that can appear surprisingly accurate, the task is representative of the problems inherent to pure deep learning methods, which is the lack of understandability. Deep networks fed with elements of the images and vector representations of words are trained to generate answers to questions about the images, making it near impossible to get to the source when errors present themselves. Figure 2.1[1] provides examples of vQA.[2] One of the primary contributions of the work presented in this thesis is in making Answer Selection (Chapter 5) more transparent by use of Question Classification (Chapter 4). The necessity of explainable AI is further explored in Chapter 3.

2.2 Semantic Text Similarity

The goal of Semantic Text Similarity (STS) is to find the degree of overlap in the meaning of two pieces of text, which ranges from text fragments that are exact semantic equivalents, to

[1]Visual QA Demo: https://vqa.cloudcv.org/

[2]Image Source: https://www.dreamstime.com/editorial-photo-coonoor-tamil-nadu-india-january-th-nilgiri-mountain-railway-runs-mettupalayam-udagamandalam-south-image95527956

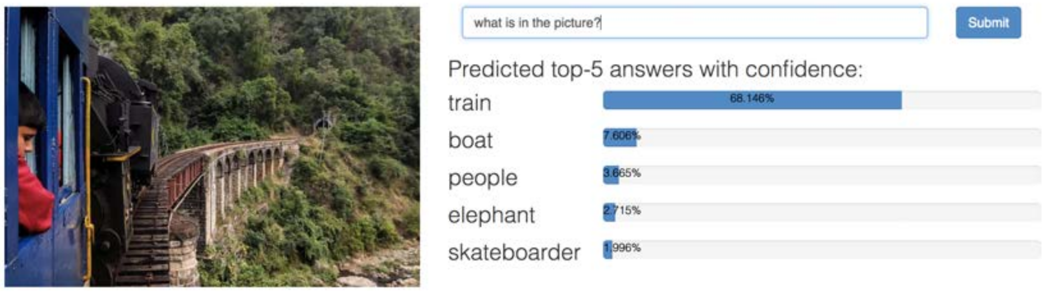


What game is this?

Predicted top-5 answers with confidence:

tennis 99.997%
soccer 0.001%
frisbee 0.001%
tennis court 0.000%
baseball 0.000%

(a) Visual QA is, at first glance, magical (Lu et al., 2016).

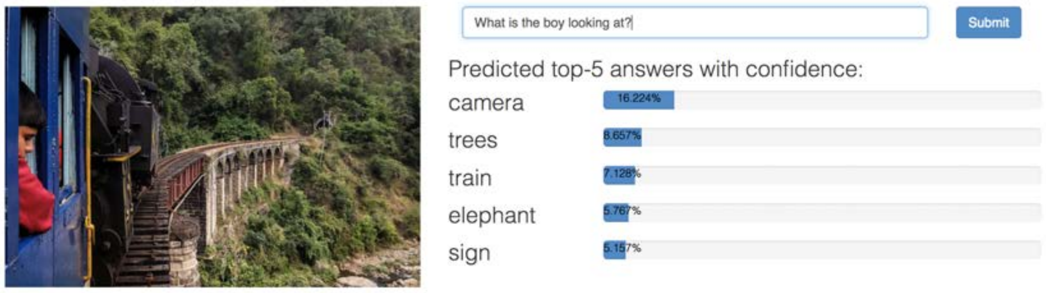


what is in the picture?

Predicted top-5 answers with confidence:

train 68.146%
boat 7.606%
people 3.665%
elephant 2.715%
skateboarder 1.996%

(b) . . . and can be surprisingly accurate.

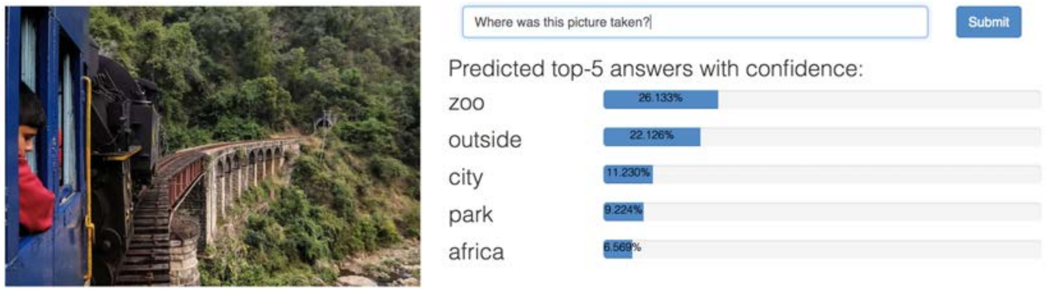


What is the boy looking at?

Predicted top-5 answers with confidence:

camera 16.224%
trees 8.657%
train 7.128%
elephant 5.767%
sign 5.157%

(c) When errors creep in, however . . .



Where was this picture taken?

Predicted top-5 answers with confidence:

zoo 26.133%
outside 22.126%
city 11.230%
park 9.224%
africa 6.569%

(d) . . . analysis and debugging is near impossible.

Fig. 2.1 A Demonstration of Visual QA and its Shortcomings which are Representative of the Shortcomings of Deep Neural Networks in General.

others that have no semantic relation. STS has a wide variety of applications, including text summarisation (Aliguliyev, 2009), machine translation (Kauchak and Barzilay, 2006), and search optimisation (Sriram et al., 2010).

The STS task, which was set by the SemEval conference for a number of years (Agirre et al., 2014, 2015), requires that submitted systems assign a score between 0 (the sentences are on different topics) and 5 (the sentences mean exactly the same thing) that reflects how similar two sentences are (Agirre et al., 2014, 2015, 2013, 2012).

Most systems that tackled SemEval's STS task consist of three main approaches: The first is text alignment, based on the content words' meaning (Sultan et al., 2014b, 2015). The second represents text as vectors, which are used to find the similarity score using a vector similarity metric (such as cosine). Third, machine learning approaches are used to compute multiple lexical, semantic, and syntactic features to classify each sentence pair's similarity.

Work on STS done as part of this project is presented in Section 2.2.

2.3 Question Classification

A crucial element of QA is Question Classification (QC), which is the task of classifying a question based on the expected answer. As an example, the question "Who is the prime minister?" could be assigned the class "person", whereas the question "Where is the prime minister?" could belong to the class "location". Since the task involves identifying the type of answer, it is sometimes referred to as Answer Type Classification. While there do exist QA Systems that do not make use of QC, QC has been shown to significantly improve the performance of QA systems (Hovy et al., 2001).

Work on QC, as in most NLP tasks, can be broadly divided into three categories: **a)** those that make use of machine learning, **b)** those that rely purely on rules, and **c)** those that are a hybrid of the two. With the increased popularity and success of machine learning techniques, most recent work on QC has been limited to methods that make use of purely statistical methods. While there continues to be some exploration into semantic information contained in sentences, such information is often converted into features for statistical models.

While several Question Taxonomies are available for use in training and testing QC systems, the most popular is the one introduced by Li and Roth (2002). This popularity stems from the 5,500 training questions and corresponding classification they provide, in addition to the classification of the 500 TREC 10 (Voorhees, 2001a) questions. Their classification is a two-level system which contains a coarse and a fine level of classification for each question. Table 2.3 lists the classification introduced by them. In this thesis, specific classes are referred to in the following way: Coarse:Fine. For example, the class animal, contained in the coarse class ENTY, is referred to as Enty:Animal.

Coarse	**Fine**
ABBR	abbreviation, expansion
DESC	definition, description, manner, reason
ENTY	animal, body, color, creation, currency, disease, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word
HUM	description, group, individual, title
LOC	city, country, mountain, other, state
NUM	code, count, date, distance, money, order, other, percent, percent, period, speed, temperature, size, weight

Table 2.3 Question Taxonomy introduced by Li and Roth (2002).

The original method proposed by Li and Roth (2002), relies on machine learning and first classifies questions into coarse classes, before then using the coarse class as a feature in fine-grained classification. They also report their results for both the coarse and fine classes. This work, however, is on fine-grained classification.

Metzler and Croft (2005) provide a detailed analysis of statistical methods of QC before 2005 while dismissing rule-based systems as "cumbersome and inflexible", and a more recent survey by Loni (2011) details QC methods using more recent Machine Learning techniques. Work on QC over the last couple of years has involved either reducing the number of features (Pota et al., 2016, 2015), focusing on specific domains (Feng et al., 2015) or using new methods in machine learning such as Convolutional Neural Networks (Kim, 2014) and Skip-Thought Vectors (Kiros et al., 2015).

State of the art in fine-grained classification, before the publication of work done as part of this project, on Li and Roth (2002)'s data was 91.6% and was achieved by Van-Tu and Anh-Cuong (2016), who base their work on using semantic features in a linear SVM. Of specific relevance to work presented in this thesis is the work by Silva et al. (2011), who first extract headwords, before then mapping these headwords into various categories using WordNet (Miller, 1995) (discussed in further detail in Section 3.2) to achieve an accuracy of 90.8%. Previous work by (Huang et al., 2008), which also makes use of both headwords and WordNet, while using slightly different methods, achieves an accuracy of 89.2%.

Work on QC done as part of this project achieved an accuracy of 97.2%, close to a 6 point improvement over the previous state of the art and is detailed in Chapter 4.

2.4 Answer Selection

Most QA systems consist primarily of three components: **a)** a question analysis component, **b)** an Information Extraction (IE) component that extracts a set of candidate sentences, and **c)** an answer extraction component that prunes this set of sentences to extract the answer. QC is performed in the first component. Its results are sometimes used in the IE component but generally used in answer extraction. The other important aspect of the answer extraction component is the analysis of linguistic features. Together, these two elements can be used to prune a set of sentences, some of which might contain the answer to a given question.

This task of selecting, from a list of sentences produced by an IE component, a subset A which contains the answer to a given question is called Answer Selection (AS). For example, given the question "Where is the group Wiggles from?", and two possible sentences (called *candidate sentences*): "the Wiggles are four effervescent performers from the Sydney area:

Anthony Field, Murray Cook, Jeff Fatt and Greg Page", and "six of the Wiggles' videos have reached multi-platinum status in Australia", the task would require one to return the first (*positive*) candidate and not the second (*negative*) candidate. AS leaves the task of extracting the Answer from a positive candidate to a downstream task.

Methods of AS rely on establishing some form of relation between the question and each of the answer candidates, such as bag-of-words, tree edit models (Heilman and Smith, 2010), semantic distances based on word embeddings (Wang and Ittycheriah, 2015), or deep learning methods such as Convolutional Neural Networks (Rao et al., 2016). To the best of our knowledge, however, this task has not been attempted with the extensive use of fine-grained QC.

A lot of the work in using QC for QA took place before the resurgence of Machine Learning. For example, Kwok et al. (2001) introduce a QA system "MULDER", that makes use of wh-phrases, which they define as the interrogative word followed by the words associated with it. Hermjakob (2001) used an extensive QC system consisting of 115 elementary question classes in their work on QA.

2.4.1 Question Taxonomy and Classification

The specific system of classes used by a QC system is known as a taxonomy, and while several taxonomies are available, this work makes use of that proposed by Li and Roth (2002), for two reasons: **a)** This is one of the most widely used taxonomies, possibly because of the large training set that Li and Roth (2002) provide, **b)** while it has been pointed out that this taxonomy might not have the widest coverage (Mishra and Jain, 2016), it is shown in Chapter 5 that it is most suited for domain-independent QA.

This taxonomy originally consisted of fifty fine classes divided amongst six coarse classes. Table 2.3 provides a complete list of these classes along with the changes made in this work (described in Section 5.2.1).

While this work on QC is an extension of previous work done as part of this project on AS (Tayyar Madabushi and Lee, 2016) that achieved an accuracy of 97.2%, other work on the same taxonomy has involved the use of Linear SVMs by Van-Tu and Anh-Cuong (2016) and Pota et al. (2016) which achieved accuracies of 91.6% and 89.6% respectively. Work using Convolutional Neural Networks (Kim, 2014) and Skip-Thought Vectors (Kiros et al., 2015) has not focused on fine-grained classification.

Work on answer selection, done as part of this project, achieved state of the art results and is detailed in Chapter 5.

2.5 Deep Learning for Answer Selection

Except for the methods presented in this work, all recent improvements to the task of Answer Selection have been achieved by use of deep learning models. This section describes deep learning components used by Rao et al. (2016) in their method for Answer Selection - We use their method as a baseline, and significantly improve upon it by integrating linguistic information as detailed in Chapter 5. While deep learning methods have become a popular choice across NLP tasks, it is our opinion that a return to linguistic analysis will significantly help in several tasks, even if thay are integrated into deep learning methods as we have done.

2.5.1 Deep Learning

Deep learning refers to a family of deep architectures that learn high-level feature representations. Although deep learning can be achieved using methods such as Markov random fields (Jordan et al., 1999; Kindermann and Snell, 1980), the most popular of deep learning methods is the use of Artificial Neural Networks with hidden layers. These hidden layers provide a way of abstracting the input and learning representations of the input features, thus doing away with the need to construct features manually.

2.5.2 Convolutional Neural Networks

Convolutional neural networks (CNN) (LeCun et al., 2004, 1990), first used in image processing, provide dense representations of sections of input. This method of dividing up the input into a grid and finding compact representations for each section of the grid for use in classification has since been used in several domains such as recommender systems (van den Oord et al., 2013) and NLP (Collobert and Weston, 2008; Young et al., 2017).

CNNs consist of four different operations: a) Convolution, b) adding non-linearity, c) Pooling or sub-sampling and d) classification (a fully connected layer). Convolution is the process of applying a convolutional filter to the input to extract features. The use of different “filters”, which consist of a matrix that is multiplied by sections of the input matrix, result in detecting different kinds of features. Figure 2.2[1] illustrates the use of a convolutional filter. [2]. While there exist convolutional filters for standard operations such as sharpening and filtering images, the filter is learned during training. Convolution filters are also called Kernels.

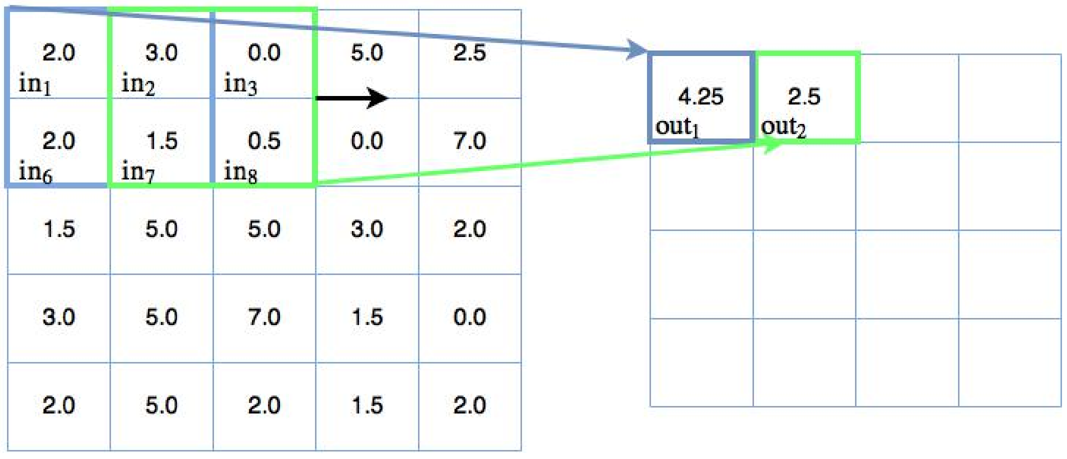


Fig. 2.2 A Convolutional filter.

CNNs next introduce non-linearity into the model which allows the modelling of non-linear functions. The prefered non-linear function in CNNs is $f(x) = max(0,x)$ and is introduced by a layer consisting of Rectified Linear Units (ReLU) due to its speed (Krizhevsky

[1]Source: http://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/
[2]A more illustrative visual is the animation at https://ujwlkarn.files.wordpress.com/2016/07/convolution_schematic.gif

et al., 2012) over other non-linear functions such as *logh* and *sigmoid* due the its relative simplicity.

Before the final classification through a fully connected layer, the result of the previous step is downsized using some down-sampling method, the most common amongst which is max-pooling. Max-pooling consists of replacing a grid with the maximum value contained within it.

2.5.3 Convolutional Neural Networks for Natural Language Processing

Words are typically input to neural networks as (dense) vectors called word embeddings. Creating a representation for sentences from such word embeddings is done using Recurrent Neural Networks (RNN) (Mikolov et al., 2013b), Long Short Term Memory Networks (LSTM) (Hochreiter and Schmidhuber, 1997) or CNNs. The relative speed of training CNNs has led to their adoption in natural language processing. Figure 2.3 (Zhang and Wallace, 2015) illustrates typical CNNs used in Natural Language Processing.

In NLP, convolutions are performed on the matrix of word embeddings as opposed to the image matrix in image processing. These convolutions are learnt during training. Max-pooling remains the most common pooling method in NLP after the introduction of non-linearity. The number and size of the kernels in a CNN are hyper-parameters that are handcrafted.

2.5.4 Multi-Perspective Convolutional Neural Networks

The method for answer selection used by Rao et al. (2016) makes use of Multi-Perspective CNNs introduced by He et al. (2015). Also, Rao et al. (2016) make use of a “Siamese” structure in which two sentences are processed in parallel by two subnetworks that share all their weights (Bromley et al., 1993). Figure 2.4 provides an overview of their network.

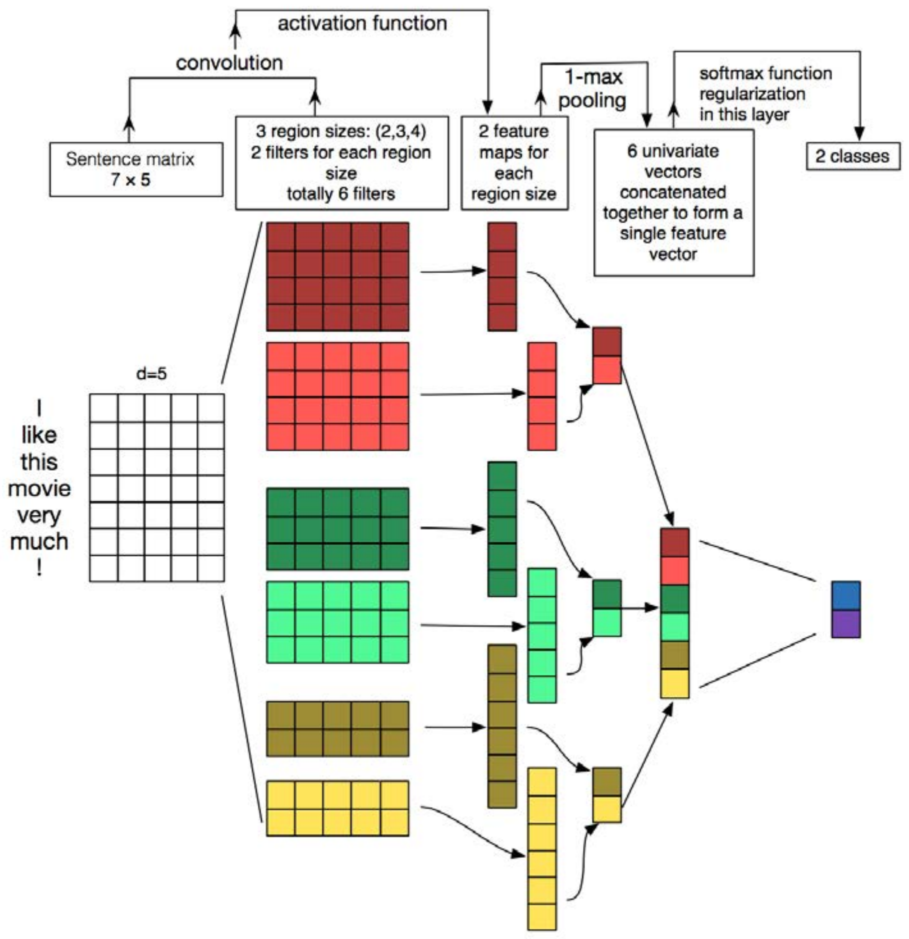

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I like this movie very much !

Fig. 2.3 Depicts three filter region sizes: 2, 3 and 4, each of which has two filters. Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps. Then 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all six maps, and these six features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here binary classification is assumed and hence two possible output states are depicted. (Zhang and Wallace, 2015).

Instead of applying convolutional filters on multiple words across the sentence, (He et al., 2015) additionally apply convolutional filters to individual dimensions of word embeddings. The intuition they provide is that, while humans may not be able to understand what individual dimensions of a word embedding represents, there might be information within a single

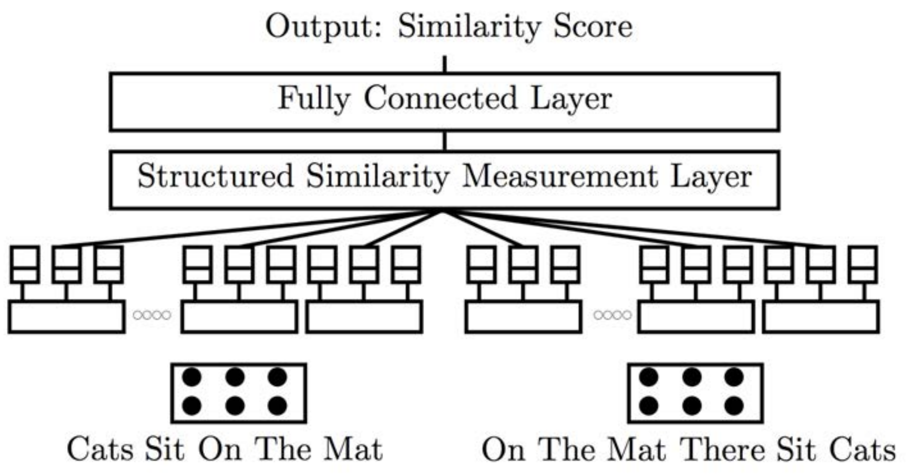

Output: Similarity Score
Fully Connected Layer
Structured Similarity Measurement Layer
Cats Sit On The Mat
On The Mat There Sit Cats

Fig. 2.4 Model overview: Two input sentences (on the bottom) are processed in parallel by identical neural networks, outputting sentence representations. The sentence representations are compared by the structured similarity measurement layer. The similarity features are then passed to a fully-connected layer for computing the similarity score (top). (He et al., 2015).

dimension of an embedding that the model can exploit. The application of convolutional filters across a single dimension of word embeddings is interpreted as over a new perspective, hence the name. They also note in their work that this multi-perspective convolution provides the most gain. Figure 2.5 illustrates the different perspectives used over word embeddings.

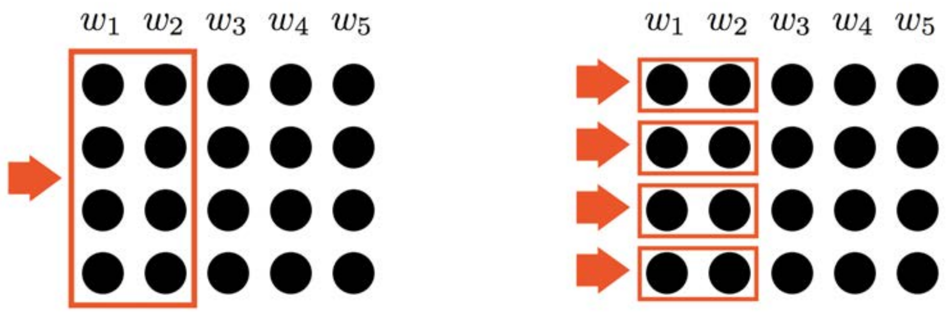


Fig. 2.5 Left: a holistic filter matches entire word vectors (here, window size is 2). Right: per-dimension filters match against each dimension of the word embeddings independently. (He et al., 2015).

He et al. (2015) also include a layer for finding the similarity between the representations of the two sentences along each of the "perspectives". The result of this is passed through a fully connected layer to find the similarity between two sentences.

2.5.5 Pairwise Ranking and the Triplet Ranking Loss Function

Rao et al. (2016) introduce a method of pairwise ranking for answer selection where prior methods relied on pointwise classification. They do this by learning a joint representation of the triplet input (question, positive answer, negative answer) before stacking a triplet ranking loss function on top, whose objective is to minimise the total number of inversions in the rankings. This method also uses the "Siamese" structure described in Section 2.5.4 with the difference that it takes a question-answer pair as input. Rao et al. (2016) make use of the Multi-Perspective CNN used by He et al. (2015) to train pairwise rankings. An illustration of this model is provided in Figure 2.6.

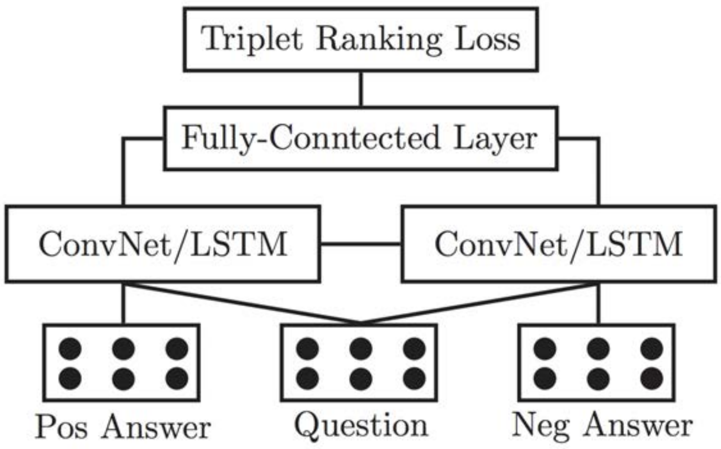


Fig. 2.6 Architecture of the pairwise ranking model, which is trained on triplets comprised of (question, positive answer, negative answer) (Rao et al., 2016).

In essence the encoding of a positive or negative answer candidate and the question are passed to a CNN or an LSTM to create a combined representation of the two. This resultant representation consists of the high-level features extracted from the data, which are passed through a fully connected layer to extract non-linear combinations. These features are used by the triple ranking loss function as described below. It should be noted that the *same* model is used for creating representations for the positive and the negative candidates. We note that Rao et al. (2016) also use a word-level component in their experiments, which is not described here as we do not use word-level representations.

The triplet ranking loss function introduced by them is a function $f(.)$ that captures the similarity scores, such that, given some question q, positive pairs (q, p^+) and negative pairs (q, p^-), the objective is to ensure that $f(.)$ assigns a larger similarity score to positive pairs than negative pairs:

$$f(q,p^+) > f(q,p^-), \forall q, p^+, p^- \tag{2.1}$$

The triplet loss function, which ensures equation 2.1 holds true by minimising the distance between the question and the positive answer while maximising the distance between the question and the negative answer, is given by equation 2.2

$$\underset{W}{min} \sum_{(q,p^+)} \sum_{p^- \in N} max(0, 1 - (f(q,p^+) - f(q,p^-))) + \lambda \|W\|^2 \tag{2.2}$$

where λ is a regularisation parameter, and W is the parameters of the neural network model $f(.)$.

2.6 Linguistic Theories

The work presented in this thesis makes use of two preexisting resources, WordNet and Dependency Parsing. This section provides the theoretical linguistic background for these resources.

2.6.1 WordNet

The first resource we make use of is WordNet, a lexical database we use for rule generalisation. While Section 3.2 details the use of WordNet for this purpose, this section focuses on the linguistic and psycholinguistic underpinnings of the WordNet project.

WordNet organises nouns, verbs, and adjectives using a different method. The way in which nouns are represented is of particular interest as this work exploits this structure for generalising learning. The independent handling of nouns as a lexical subset is backed by the observation that patients suffering from anomic aphasia, a condition often resulting from stroke, are left unable to name things or otherwise use nouns in speech, while otherwise unaffected (Caramazza and Berndt, 1978).

Nouns in WordNet are organised in a hierarchical structure, one that is very relevant to the work presented in this thesis. This organisation is based on psychological experiments that showed that people store semantic information in a hierarchical structure so as to optimise the amount of information that is required to be stored. As a consequence people took less time to verify that canaries could sing than to verify that they could fly and even longer to verify that they had skin (M. Collins and Ross Quillian, 1972). This increased time is associated with the increased distance across a semantic hierarchy. M. Collins and Ross Quillian (1972) also argued that this difference in reaction time implies that information associated with different nouns is stored in an inheritance system, much like the system we use to define Types (Section 3.7.1).

However, this notion of inheritance has been questioned by other researchers such as Conrad (1972) who point showed that while “can move” and “has ears” are both properties that people associate with animal, “An animal can move” is confirmed more rapidly than is “An animal has ears”.

Despite this, the creators of WordNet assume that information is both hierarchical and that the inheritance assumption is indeed correct, but that reaction times probably indicate a difference in word use rather than word meaning (Miller and Charles, 1991).

2.6.2 Dependency Parsing

Dependency Parsing is the process of extracting dependency relations, which are based on the linguistic notion of grammatical relationships, from a sentence. Modern statistical parsing algorithms are based on the Eisner algorithm (Eisner, 1996) for finding the dependency parse of a sentence and employ deep learning methods to estimate the parse tree significantly faster than the original algorithm. Eisner (1996) provided three statistical models which are used as the basis for generating the dependency parse of a sentence. Unlike previous work, they establish these probabilistic models not on what is observed in training data, but base them on linguistic analysis. Of the three models they describe and test (A, B and C), the first two assume that a speaker (or writer) focus on the hearer's needs (comprehension) whereas the last emphasises the speaker's needs (i.e. ease of generation). Their experiments show that Model C, with its emphasis on the speaker's needs, produced the best results implying that "speakers should not hesitate to add extra prepositional phrases to a noun, even if this lengthens some links that are ordinarily short, or leads to tagging or attachment ambiguities".

The list and scope of relationships between various elements of a sentence developed by linguists, while extensive, are often contradictory. Recent work, such as the Universal Dependencies project (Nivre et al., 2016), has focused on providing a list of relations that are linguistically motivated and computationally useful. In this work, we analyse sentence structure using the Stanford Parser (Chen and Manning, 2014). The Stanford parser generates dependencies that adhere to the Stanford typed dependencies representation (de Marneffe and Manning, 2008). This representation was developed with the aim of ensuring that the description of relationships must be accessible to any user who could benefit from text understanding. de Marneffe and Manning (2008) provide a detailed analysis of the design choices and trade-offs along with a comparison against other frameworks such as PARC (King et al., 2003).

2.7 Summary

This chapter provided details on current Question Answering systems and their shortcomings before introducing two tasks, Question Classification and answer selection. An overview of the related work on these tasks was provided with specific emphasis on a couple of deep learning methods for answer selection that this work relies on (Chapter 5). The final sections of the Chapter explored certain linguistic theories that this work is based on.

The next chapter introduces the theoretical basis for our work into a hierarchical structure of concepts before then detailing its implementation through what we call Types. An empirical analysis of the effectiveness of Types for the task of Semantic Text Similarity is provided as a proof of concept. Subsequent chapters explore the use of these Types in the tasks of Question Classification and Answer Selection.

3

Conceptual Hierarchies and their Representation through Types

This chapter provides the theoretical foundation for subsequent work into Question Classification and Answer Selection. Cognitive psychology and philosophy provide several models of cognition and reasoning within the human mind. Without relying on any one of these hotly debated and sometimes contradictory theories, this work merely assumes that concepts exist and they can be hierarchical.

3.1 Concepts

Concepts are generalisations or abstractions that allow the use of previous experience in new situations. For example, a person might have a concept of a dog based on some general characteristics of dogs, such as them having four legs, barking or being living beings. Such characteristics are the default set of characteristics associated with the concept. A dead dog, for example, has different characteristics as does a blind dog. Figure 3.1[1] provides a visual representation of the concept of a TREE. Prototype theory postulates that such abstractions (Concepts) are represented by their "best" example (Hampton, 2006), which while useful when thinking about Concepts as presented in this chapter is not central to this work.

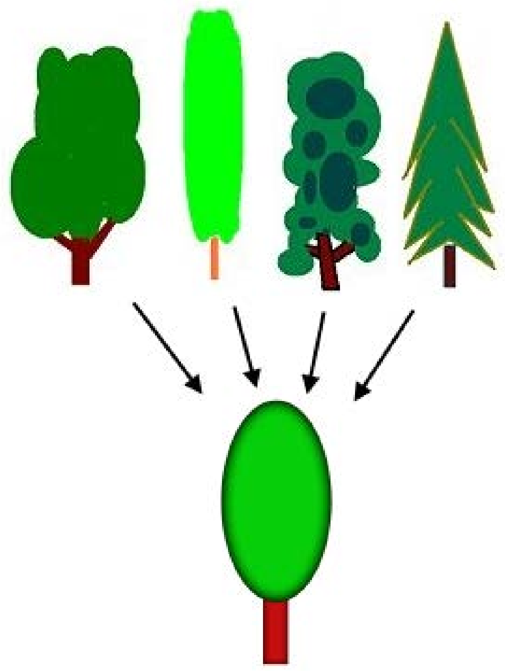


Fig. 3.1 The Concept TREE is created in the mind by collecting similarities from different examples and so creating a generalisation.

Concepts are related to other Concepts providing us with ways of making use of the experience we accumulate. Such relationships between Concepts can be represented using semantic networks which are expressed as semantic triplets as in "Bob knows John" (Lehmann, 1992). Conceptual graphs are graphical representations of semantic networks which allow logical operations (Sowa, 1976). Semantic networks also lack formal semantics which was

[1]By Tomwsulcer [CC0], from Wikimedia Commons

rectified by the introduction of ontologies which have their logical formalism provided through description logic (Baader et al., 2003). There are several publicly available semantic networks including ConceptNet[1] and WordNet. ConceptNet includes common sense knowledge (obtained from Open Mind Common Sense) and links to other knowledgebases such as DBPedia and Wiktionary[2] whereas WordNet provides lexical information which contains less noise, an important reason we use WordNet in this work.

3.2 WordNet

WordNet, inspired by the psycholinguistic theories of lexical memory from the 1990s, provides different organisations for each part of speech (Miller et al., 1990). WordNet organises nouns as topical hierarchies, verbs as entailment relationships and adjectives and adverbs as N-dimensional hyperspaces.

As described by Miller et al. (1990) the word "word" is used to describe both the physical utterance or inscription and the Concept behind it. They distinguish the two by calling the first a "word form" and the latter the "lexicalised concept". As a consequence of polysemy and synonymy, a lexicalised concept may be expressed by multiple word forms (synonyms) each of which might be polysemous. WordNet does not attempt to explain what each lexical concept is but instead aims to provide a way to distinguish them from each other. They exemplify this by use of the word "board" which is distinguished using the synonym sets (synsets) {board, committee} and {board, plank}. If no such synonyms exist, WordNet provides a gloss, as in {board, (a person's meals, provided regularly for money)}. Thus, WordNet uses synonym sets or synsets to capture lexicalised concepts or different senses of words. We call the most commonly occurring sense of a word the first sense and the others the *other-senses*

[1]http://conceptnet.io/
[2]https://en.wiktionary.org/wiki/Wiktionary:Main_Page

The hierarchical structure WordNet used for nouns organises (noun) synsets using hyponyms and hypernyms. If $\forall e \in W_1$, e is an instance of W_2, W_1 is a hyponym of W_2 and W_2 is a hypernym of W_1. The hypernym or hyponym closure of a synset is the transitive closure starting at that synset with the direction defined by hypernymy or hyponymy. In this work, we only consider the first sense of words in the closure, unless stated otherwise. Also, through the rest of this work the word "word" is often used to mean the "synset representing the correct sense of that word in the given context" (which might not be its first sense).

3.3 WordNet, Question Classification and Types

The task of classifying questions based on the class of the expected answer is called Question Classification and has previously been discussed in Section 2.3. This work breaks down the task of Question Classification into three subtasks (further detailed in Chapter 4): a) analysis of the structure of a question to find the most relevant phrase that can be used to classify the question, b) the extraction of the "head" or another essential word, and c) the linking of that part to the relevant question class.

Consider the examples presented in Table 3.1. Notice how questions with the word "What" complicate the classification of questions making the task of classifying questions non-trivial - nearly every question that uses any other wh-word (including "how" and "describe") can be re-written using "what". For example, the question "How many hostages were killed in the Entebbe raid?" can be re-written as "What is the number of hostages killed in the Entebbe raid?"

There are specific words in the question that give away its question class as is clear from Table 3.1. This specific word, which is used to classify the question, is determined using the parse tree of the question and hence its syntactic structure (detailed in Section 4.2). The intuition behind the method for classifying questions presented in this work is that *a set of words can replace such a word without changing the class that the question belongs to. We*

Question	Word used for Classification	Question Class
What athlete makes the most money from sports merchandise sales?	athlete	Human:Individual
What president lived at 219 North Delaware Street, Independence, Missouri?	president	Human:Individual
What city boasts Penn 's Landing, on the banks of the Delaware River?	city	Location:City
How many hostages were killed in the Entebbe raid?	many (hostages)	Number:Count
In what year was De Gaulle elected president of France?	year	Number:Year
What Indian tribe is F Troop perpetually doing battle with?	tribe	Human:Group

Table 3.1 Questions with associated headwords and question classes.

postulate that such a set of words represents a Concept, and call such a set a ***Type***. When a synset and its hyponym closure belong to a Type, we replace them by just the synset so as to make the definition of Types more compact. This definition using synsets is translated into words by considering those words that are most representative of each synset.

The remainder of this section details the relationship between the word used for Question Classification in a question and its WordNet synsets and the next section (Section 3.4) provides a more general description of Types including how we create them specifically for Question Classification using a training set. As a first approximation: all words in the hyponym closure of the word can replace it in the question without changing the question's class.

For example, the syntactic structure of the question "What athlete makes the most money from sports merchandise sales?" is analysed and the word "athlete" is determined to provide information regarding the class of this questions, which is Human:Individual. Once we have established this, we can deduce that if any hyponym of the word "athlete" appears in the same location, then such a question has the same class. For example, “What ball hawk makes the most money from sports merchandise sales?”, “What fielder makes the most money from sports merchandise sales?”, and “What wingback makes the most money from sports merchandise sales?” all belong to the same question class, namely Human:Individual. It should be noted that this also allows for the classification of questions of the same *form* (syntactic structure) such as “What wingback recently retired?” and “What kicker lived at 219 North Delaware Street, Independence, Missouri?”

However, not all questions of the class Hum:Individual having this syntactic structure are captured by this information. Consider, for example, the question “What president lived at 219 North Delaware Street, Independence, Missouri?”. The word “president”, which is used to classify this question, is not a hyponym of the word “athlete”, but both words are hyponyms of the word “person”. So, it is important to add the “highest” (assuming moving through hypernyms is considered going “up”) possible synset the hyponym closure of which consists of words that can replace the current word without changing the class of the question - we call such a word the **primary classification word**. Finding the primary classification word ensures that a single rule “person or any word in its hyponym closure located at this location in a question implies that the question belongs to the question class Human:Individual” (see Section 4.4.2 for a more detailed description of rules) captures several hundred cases.

This information, however, is still not enough information to classify all questions belonging to Human:Individual that are of the same syntactic structure as is exemplified by the question “What movie star acted in the movie Titanic?”. Here the word “star” (disambiguated using the word movie - Section 4.4.1) is used to classify the question. Similar words whose first sense is not a hyponym of “person” must be included in the set of words (which we call a Type) that can replace the word in the question structure without changing its class. Also, those words whose non-first sense is one that we are interested in (e.g.

musician.n.02, which we refer to as *other-sense* words) along with their hyponym closure must also be added to the Type.

So far we have established that for every syntactic structure and question class available, a particular word (the “primary classification word”), its hyponym closure, relevant *other-sense* hyponyms and their hyponym closure all belong to the Type used to classify the question. However, two further modifications need to be made to the Type to capture all relevant words that can appear at this location of similarly structured questions belonging to the same question class. The first is when there are multiple “primary classification words” for the same class as in the case of “food”, “fruit”, and “produce” for the question class Entity:Food, and the second is when the hyponym closure must exclude a certain sub-tree as in the case of the sub-tree starting at “person” when classifying questions of the class Entity:Animal. For example the question “What animal can run fast?” can be classified using the word animal. However, when creating the associated Type we must exclude the sub-tree starting at “person” in the hyponym closure of “animal” as the question “What person can run fast?” does not belong to the question class Entity:Animal but instead to the class Human:Individual.

Finally, WordNet provides a hierarchical structure only for nouns. However, words that are not nouns also provide information on the class a question belongs to as in the case of the word “cause” in the question “What caused the tsunami?” In such cases, there are no hyponyms to be considered when creating the relevant Type.

3.4 Types beyond Question Classification

The previous section described how sets of synsets are used to determine question classes. While WordNet uses synsets to define lexicographic concepts, this work makes use of a set of synsets (and their hyponym closure) associated with a syntactic location in the question to represent Concepts which help in classifying questions. We call such sets Types.

Types consist of:

1. One or more primary classification synsets, for each of which the following must also be added to the Type
   - Its hyponym closure
   - *Excluding* those synsets which are an exception (along with their hyponym closure)
   - *Including* those non-first sense synsets which act in a way similar to the other hyponyms (and their hyponym closure)
2. Any other words (excluding their hyponym closure) which behave similarly but their hyponyms do not.

Types represent a Concept which provides crucial information required to classify a question. The specific combination of synsets that make up a Type is determined through manual intuition with the aid of careful analysis using the method described in Section 3.4.1 below.

It should be noted that Types are specific to the task at hand, and so far we have seen Types in the context of Question Classification. However, Types are not the same as the question classes. For one, multiple Types (and their associated syntactic locations) are mapped to the same question class, and secondly, the same set of synsets (a Type) can map to different question classes based on their location within the syntactic structure of the question. Section 4.4.2 which details the rules relating Types to question classes further clarifies this difference.

This work makes use of Types in two independent tasks, the first of which is Question Classification. The second is the task of Semantic Text Similarity (STS), and in each case the number of Types and the sets of synsets that make up each Type are different. While Types determine the class of questions in Question Classification they provide a way of comparing "similar" elements of sentences in Semantic Text Similarity. The intuition is that it helps to increase the weight of aligned words of the same Type when measuring similarity. For example, consider the two pairs of sentences: "The man watched the boys play", "The man watched the birds fly away", and "The man watched the boys play", "The man watched the children play." In this case, increasing the similarity of sentences where the aligned words are of the same Type (a Type representing individuals in the second pair) as opposed to those having different Types (Types representing individuals and animals in the first pair) provides a more accurate representation of the similarity of two sentences. A more detailed description of this is provided in Section 2.2.

While both of these Type definitions are created manually, the simpler one, used for Semantic Text Similarity, is defined top-down, by first observing all possible synsets starting with the top level one in WordNet, namely "Entity". The next section provides details on the more involved process of creating Types for Question Classification.

3.4.1 Learning Types for Question Classification

As mentioned in the previous section, Types are specific to a task. This section describes the method used to define Types as used in Question Classification.

The process of defining Types for a task as complicated as Question Classification is impossible without looking at examples of questions due to a large number of possible syntactic structures and classes of questions. It is also essential to have a systematic process to analyse and sift through such a large amount of data. To this end, a method of simultaneously "learning" Types and rules for Question Classification was developed and consists of:

1. Parse the question and automatically create its semantic map (Section 4.2, specifically Table 4.1) which provides the necessary syntactic information to identify the word useful in classifying the question. For example, the question "Which actress . . . " is analysed to identify the word 'actress' located at the head of the main noun phrase.

2. Explore the hypernyms of the word identified and extract the highest level at which the class of the question does not change (thus identifying the primary classification word) and add it to the Type representing words at this location in similarly structured questions. In the above example, these hypernyms are *performer*, *entertainer* and *person*.
3. Explore the hyponyms of the primary classification word, so any exceptions are added to the Type as exceptions.
4. Explore the hyponyms of the primary classification word and add any *other-sense* synsets (and their hyponyms) which result in the question retaining the same class when replacing the primary classification word.
5. Once this process is complete, the required rule is *"If any word contained in this Type occurs at the previously identified location in a question, then such a question is of type Human:Individual"*.

Importantly, this learning process, which we call Abstraction-related Concept Hierarchies Learning (ArCH Learning) is currently a manual process as there is no obvious way to extract synsets that belong to a particular Type, especially since small errors in this process are vastly magnified through hyponyms. So, in the above example, it is important to stop at "person" and not to go further "up" to its hypernym "organism", as the question of the form "What is the name of the organism ... " is not of the class Human:Individual.

As an example of words that are not nouns but belong to some Type, the word "meaning" would enable the creation of a Type and associated rule to classify questions such as "What is the meaning of the word ... ?", also, "What does the word ... mean?" to the question class Desc:Definition.

Not all of the defined Types have a direct association with a question class. For example, the Type *people_from*, consisting of 'inhabitant.n.01' and its hyponym closure enables the identification of the class Entity:TermEq (i.e. equivalent term). This classification is done by checking to see if the question asks us what people from a particular place call something, by use of the syntactic structure "What *auxiliary_verb people_from* call *word*?". As an example, the question "What do Italians call noodles?" matches this rule and belongs to the question class Entity:TermEq.

Groups of verbs are also defined as belonging to certain Types, such as the Type of verbs that can only be performed by a person (e.g. invent) and the Type of words that require certain additional processing, such as a possessive or a prepositional *roll* (Section 4.3.1). These rules do not benefit from the hierarchical structure of Concepts provided by WordNet. We leave the automation of the process of creating Types for future exploration and present our ideas on how this might be possible in Section 6.3.1.

3.5 The Relationship Between Types and Word Embeddings

Typically, word embeddings are used to find semantic similarity between words using the cosine of the angle between the vectors representing two words. Since word embeddings are generated from a co-occurrence matrix, such a semantic similarity is established based on the assumption that words in a similar context (having similar words surrounding them) tend to have similar meanings.

However, it is not just synonyms that occur in the same context. Consider, for example, the sentence "There were a number of dogs waiting to be fed". The word 'dogs' can be replaced by 'cats', 'cows', 'animals', and 'alsatians'. The relationship between these words is that they are hypernyms, hyponyms or sister terms of the word 'dog'. So the semantic similarity captured by embeddings is a hybrid between synonyms, hypernyms, hyponyms and sister terms (Perek, 2016).

Types derived from a cognitive model are very similar to word embeddings which are derived using statistical methods, a requirement for each to capture the information they do.

However, the advantage of Types stems from the flexibility and especially the transparency they provide.

One direction of research that can be explored in the future is the use of word embeddings to reduce the amount of manual labour required for defining types (Chapter 6).

3.5.1 Types as an Accessible Alternative to Deep Learning

In addition to the shortcoming of not being transparent and thus making debugging near impossible (Section 1.4), deep learning methods have an additional shortcoming that is inherent to statistical methods - they require a tremendous amount of annotated data.

In recent times, access to such information has become restricted, and large corporations have a monopoly on such data. For a small organisation, academic or corporate, to compete with established players is like starting a lemonade stand to compete with a large soft drink manufacturer.

Popular deep learning methods are of little use without the vast amount of data and powerful servers required to train them. Types, on the other hand, drastically reduce the amount of data required to create models (Section 3.4) thus making state of the art natural language processing methods available to those with fewer resources.

3.6 Empirical Evaluation of Types: Semantic Text Similarity

Having explored the theoretical basis for the use of hierarchical Concepts and their implementation through Types, we explore an empirical evaluation of Types as applied to the task of Semantic Text Similarity (STS) in this section.

Work described in this section was published in the paper *UoB-UK at SemEval-2016 Task 1: A Flexible and Extendable System for Semantic Text Similarity using Types, Surprise* *and Phrase Linking* in the Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016) (Tayyar Madabushi et al., 2016).

This work was done in collaboration with Mark Buhagiar, who worked on STS between definitions described in Section 3.7.4. The other co-author of this paper is Dr Mark Lee one of the supervisors of this thesis.

3.6.1 An Introduction to the Task of Semantic Text Similarity

This section presents a system for measuring Semantic Text Similarity in English. Three novel techniques are introduced: the use of *Types* (Section 3.7.1), methods of linking phrases (Section 3.7.1), and the use of a *Surprise Factor* (Section 3.7.1) to generate 8,370 similarity measures. We choose a subset of these measures using methods detailed in Section 3.7.3, combine them with a limited set of features and use Support Vector Regression and Kernel Ridge Regression to generate a Similarity Score (Section 3.7.3).

The system presented herein outperformed the previous state of the art of SemEval 2015, and our best performing run achieved a score of 0.71 on the 2016 test set as a whole and over 0.8 on the majority of the datasets.

Our approach also handles definitions separately from arbitrary sentences, as we observed that their structure is significantly different. Since the method of measuring the similarity of definitions does not use Types, this section focuses on the generic approach, with definition similarity discussed briefly in Section 3.7.4.

3.6.2 Preprocessing

Due to the varied nature of the input presented we perform various data cleaning operations. We start by expansion of common contractions (e.g. “isn’t”) and informal contractions (e.g. “howz”, “couldve”). We then perform a spell check and hyphen removal, which are conditional, in the sense that a word is not modified unless the modified form appears in the other sentence. All remaining hyphens are replaced by spaces, a method different from those that previously handled hyphens (Han et al., 2013).

We also perform case correction, as has been done previously (Hänig et al., 2015), since we observe several instances wherein sentence capitalisation is not suitable for parsing (e.g. headlines and forums).

3.7 Similarity Measures

We use two measures, which are boosted based on the different parameters described in Section 3.7.1.

Alignments

The first measure makes use of the aligner developed by Sultan et al. (2014a), which was used to achieve State of the Art results in 2014 and 2015 (Sultan et al., 2014b, 2015).

Our use of the aligner disregards sequences thus making use of the aligner more as a synonym finder, with the additional power of the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013).

Word Embeddings

Word embeddings provide a method of mapping words or phrases to vectors, whose cosine distance represents semantic similarity. They have proved to be dominant in many NLP tasks, and have been used by top-ranking systems at SemEval STS (Hänig et al., 2015; Sultan et al., 2015). We use word2vec[1], with the model trained by Google on the Google News dataset, through its Python interface Gensim[2].

[1]https://code.google.com/p/word2vec/

[2]https://radimrehurek.com/gensim/models/word2vec.html

We make use of word2vec in two distinct ways. The first is by extracting the mean of the vector representation of each word of a Type and finding its cosine similarity between the two sentences. The second is by adding the word2vec similarity scores of words not aligned by contained in the same Type. We also provide the option of disregarding word pairs that have a score of less than 0.3, a method similar to that by Hänig et al. (2015).

3.7.1 Boosting Similarity

In this section, we detail the variations used to generate different similarity measures. These variations are not used simultaneously, but are instead combined as described in Algorithm 1 (Section 3.7.2), which iterates through all possible variations to generate a different similarity score associated with each combination.

Type Specific Comparison

Given a sentence pair, we calculate their similarity based only on how similar corresponding Parts-of-Speech (POS) are, a method previous systems have made use of, either implicitly (Kashyap et al., 2014; Sultan et al., 2015) or explicitly (Hänig et al., 2015).

We extend this idea by defining what we call *Types* for Semantic Text Similarity, which further subdivides each POS, previously discussed in Section 3.4. A Type represents an abstract Concept that several words can share. Consider the sentence pair “A man is sitting on a stool”, “A boy is sitting on a chair”. Although the words “man”, “boy”, “stool” and “chair” are all nouns retrieved, an effective strategy for comparing these sentences would be to compare the first two and the last two words independently, before then adding up their similarity. To achieve this, we categorise words into different Types, which are then compared across sentences. In this case, such a categorisation might place the first two into the Type “Person” and the others into the category “Artifact”. This problem could very easily extend to the problem of Word Sense Disambiguation, which we avoid by use of a heuristic.

We calculate the Type of a noun for Semantic Text Similarity by the use of WordNet hypernyms. We recursively find hypernyms until we reach a manually selected set of Concepts (such as food.n.02). We manually combine sets of such Concepts to define a Type. As a concrete example, we combine the WordNet Concepts "communication.n.02", "food.n.02" and other similar Concepts into the Type "thing_r1". As a single word can be part of several Types, based on the particular sense of the word, we pick the most frequently occurring Type for each word. Appendix A provides details on the definitions of Types used for STS.

Phrase Linking

Consider sentences with the phrases "Prime Minister" and "Prime Number". Although the word "Prime" is present in both sentences, the context in which it is being used makes this irrelevant. In this particular case, the semantic similarity of the sentences is dependent on the head of the phrase that the word "Prime" is contained in (i.e. "Minister" and "Number"). This is also the case with phrases that contain adjectives and adverbs.

We address this by finding phrases that consist of adjectives, adverbs and nouns, and varying the importance of the semantic similarity between words that are not the head of that phrase by multiplying the similarity score by various weighting factors described below. The similarity of each word that is part of such a phrase, but not the head of the phrase, is additionally weighted in three different ways. The first assigns a zero or one weight based on whether or not the head of the phrase is aligned, the second provides a weight based on the number of words, following this word, that is aligned in the phrase, and the third ignores the phrase structure.

Noun Importance

Consider the following sentence pairs with relations assigned by human annotators: “A *boy* is playing a guitar.”, “A *man* is playing a guitar.”, rel: 3.2; and “A man is cutting up a *potato*.”, “A man is cutting up *carrots*.”, rel: 2.4. Although both pairs of sentences differ by exactly one noun, the first pair was considered to be more closely associated than the second. We associate this to what we call the “Surprise” and assign a value to this, which we call the “Surprise Factor”. Surprise is based on the work by Dunning (1993), who observed that the assumption of normality of data is invalid as “simple word counts made on a moderate-sized corpus show that words that have a frequency of less than one in 50,000 words make up about 20-30% of typical English language news-wire reports. This ‘rare’ quarter of English includes many of the content-bearing words ....”

We define the Surprise Factor of a noun or phrase to be proportional to the number of Web Search Hits for that phrase or term, while inversely proportional to the Search Hits in the case of proper nouns. Intuitively this makes sense, as words that are more common generate less Surprise, carry less information, and will also be more widely used on the Internet.

We incorporate this idea of Surprise by adding the option of additionally weighting nouns by the total number of Web Search Hits or Results[1]. We define, H_i to be the number of Web Search Hits for the noun i, HT the total number of hits for all nouns defined by:

$$HT = \sum_{i=0}^{N} H_i \tag{3.1}$$

N_i the fraction of the Search Hits that noun i captures, defined by:

$$N_i = \frac{H_i}{HT} \tag{3.2}$$

and NT the normalised total of all nouns (C) in a given sentence defined by

[1]We use the Bing Web Search API: http://www.bing.com/toolbox/bingsearchapi

$$NT = \sum_{i=0}^{C} N_i \tag{3.3}$$

We define the Surprise of word *i* in terms of the above in Equation 3.4.

$$S_i = \frac{N_i}{NT} \tag{3.4}$$

3.7.2 System Overview

Algorithm 1 provides an overview of the system we use to generate the various Similarity Scores. We call each combination that generates a score a “Method”. We use thirty weights for Types A while providing the option of dividing the scores by the number of WordNet Synsets (UseSSToWeight), which captures any dilution due to a word’s different senses. We also scale word2vec scores by different values. This gives us a total of 8,370 “Methods”.

In calculating the similarity score, we capture the fraction of each Type that is aligned and scale it by the weight of that Type. This is captured in Equation 3.5 where $score_t$ represents the Similarity Score assigned to Type t by either of the measures detailed in Section 3.7, $count_t$ represents the number of words of Type t in *both* sentences, w_t the weight of Type t in the current iteration, and T is the total number of Types.

$$5 \times \left(\frac{\sum_{t=0}^{T} score_t \times w_t \times 2}{\sum_{t=0}^{T} count_t \times w_t}\right) \tag{3.5}$$

3.7.3 Combining Similarity Scores

As described above, we use variations to generate thousands of Similarity Scores, each of which we call a “Method”. Each Method’s performance varies depending on the input. In this section, we detail the process for combining these Methods, which is performed using either Support Vector Regression (SVR) or Kernel Ridge Regression (KRR).

```
Data: Sentence Pairs
Result: List of Similarity Scores
Initialise list of similarity scores “SimScores” to empty list;
for w in Type-Weights do
    for n in NounHandleMethod do
        for av in Adjective-AdverbHandleMethod do
            for UseSearchHits in [True,False] do
                if UseSearchHits == True then
                    Calculate Similarity Score (SS) using Alignments;
                    Append SS to SimScores;
                    Continue;
                for UseSSToWeight in [True, False] do
                    Calculate Similarity Score (SS) using Alignments;
                    Append SS to SimScores;
            for UseWeightCutOff in [True,False] do
                for VectorCombineMethod in [UseAlignments, UseMeanVector] do
                    for ScaleVectorSimBy in [ 6, 5, 4, 3, 2, 1, “log” ] do
                        Calculate Similarity Score (SS) using Word Embeddings;
                        Append SS to SimScores;
Return SimScores (the list of similarity scores);
```

Algorithm 1: Calculating Semantic Similarity Scores

Picking a Subset of Methods

Instead of using all generated scores as features for the SVR or KRR model, we first prune the scores generated by various methods. To perform this pruning, each of our Methods is ranked using three metrics with respect to the training set. The first is by use of the Pearson Correlation between the similarity scores generated and the similarity scores of the training set (a criterion we call “Method”), the second is based on the sum of the absolute error between the similarity scores (a criterion we call “Error”). The third metric aggregates the rankings from the two criterion described above, and is called this criterion “Combine”. We select the similarity scores generated by the top 50 methods using one of the three selection criteria for each run.

Generating Similarity Scores

In addition to using scores from the chosen Methods, we add the following features to some of our submitted runs: a) a binary value to represent whether each of the sentences were case corrected, b) the length of each of the sentences, c) the number of continuous aligned or unaligned sequences, d) the maximum and minimum lengths of continuous aligned or unaligned sequences, and e) a binary value to represent alignments that are non-sequential.

It should be noted that the specific Methods we choose for use in the SVR or KRR will depend on the training data picked. We found, by testing our system using several different combinations of training data, that the best results were achieved when our system was trained on the headlines data from the years 2015, 2014 and 2013. The method selection criterion, the regression model and parameters used for each of the runs submitted are detailed in Table 3.2. Although some of the settings are very similar (e.g. run2), we noticed that these minor changes translated to significant differences in performance.

3.7.4 Finding Similarities between Definitions

To find similarities between definitions, we first identify the word that is being defined by the definition. We achieve this by use of OneLook's reverse dictionary search[1], which returns many candidate words for a given definition. For each definition, the similarity of the top 10 candidates is then computed using word2vec and five similarity metrics provided by WordNet: Path distance (Ferlež and Gams, 2004), Leacock-Chodorow (Leacock and Chodorow, 1998), Wu and Palmer (Wu and Palmer, 1994), Jiang-Conrath (Jiang and Conrath, 1997) and Lin (Lin, 1998). The final score is scaled between 0 and 5 and averaged across the ten candidates returned by OneLook.

We found this method of calculating similarities between definitions to be very good at telling if two definitions refer to the same word, but not ideally suited for measuring *how*

[1]http://www.onelook.com/reverse-dictionary.shtml

Run	Headlines		Other Datasets	
Run1	Model:	KRR	Model:	SVR
	Features:	False	Features:	False
	Train:	Headlines	Train:	Headlines
	Picked:	Combine	Picked:	Combine
	Kernel:	Poly	C:	100
	Alpha:	50	Epsilon:	0.05
			Gamma:	9e-05
Run2	Model:	SVR	Model:	SVR
	Features	True	Features:	True
	Train:	Headlines	Train:	Headlines
	Picked:	Method	Picked:	Method
	C:	100	C:	100
	Epsilon:	0.01	Epsilon:	0.05
	Gamma:	9e-05	Gamma:	9e-06
Run3	Model:	SVR	Model:	SVR
	Features	True	Features:	True
	Train:	Headlines	Train:	Headlines
	Picked:	Method	Picked:	Combine
	C:	100	C:	100
	Epsilon:	0.01	Epsilon:	0.01
	Gamma:	9e-05	Gamma:	9e-06

Table 3.2 Parameters and models used for each run. The row Features represents if features were used, Train represents the training data used, and Picked represents the selection criterion (Method, Error or Combine).

similar they are. As a consequence, we found that the results were clustered around 0 and 5.

The system produced a Pearson correlation of 0.69 on the SemEval 2014 definitions data set.

3.7.5 Results and Analysis

Dataset	Best	Run1	Run2	Run3
Mean	.77807	.70940	.70168	.70911
postedit-ing	.86690	.81272	.80835	.81333
ques-ques	.74705	.56040	.47904	.56451
headlines	.82749	.81894	.82352	.81894
plagiarism	.84138	.82066	.82406	.81958
ans-ans	.69235	.52460	.55217	.52028

Table 3.3 Performance on the 2016 STS Test Set

We list the performance of our system in Table 3.3. Our system's poor performance on the ans-ans and ques-ques datasets can be attributed to our choice of training data, which, although well suited for previous years, was not well suited for these datasets.

However, our system produces State of the Art results on the 2015 Test Sets. A breakdown of each of the run's performance against the 2015 STS data set is provided in Table 3.4. We note that the results we have reported for the previous State of Art for individual data sources are not the results from just the winning system but the State of Art across all Systems for that data source. Our system also achieves comparable results (0.7793) to that presented by Sultan et al. (2015) (0.779) on the 2014 STS dataset. The weighted mean reported herein does not include definitions as the method for finding similarities between definitions does not make use of Types.

Source	St. of Art	Run1	Run2	Run3
Mean	0.8015	0.8086	**0.8147**	0.8130
ans-std	0.7879	0.7919	**0.7965**	0.7953
ans-for	**0.739**	0.7184	0.7137	0.7090
belief	0.7717	0.7703	**0.7811**	0.7752
head-lines	0.8417	0.8508	**0.8532**	**0.8532**
images	**0.8713**	0.8448	0.8617	0.8615

Table 3.4 Performance on the 2015 STS Test Set.

Source	St. of Art	Run1	Run2	Run3
Mean	0.779	0.7714	**0.7793**	0.7790
de-forum	0.504	0.5435	0.5630	**0.5636**
de-news	**0.785**	0.7718	0.7774	0.7756
head-lines	0.765	**0.8082**	0.8055	0.8055
images	0.834	0.8340	0.8492	**0.8496**
OnWN	0.875	–	–	–
tweet-n	**0.792**	0.7551	0.7569	0.7573

Table 3.5 Performance on the 2014 STS Test Set.

Table 3.5 provides a comparison of our system against the previous State of the Art for the STS 2014 data set. The overall State of Art across all data sets was reported by Sultan et al. (2015) based on their 2015 System.

3.8 Summary

This chapter provided the theoretical background for Types, a novel method of representing Concepts that can be used in different tasks. The chapter then provided details on what constitutes a Type before using the task of Semantic Text Similarity to validate the effectiveness of Types.

The next chapter details the task of classifying questions based on their expected class of answer and details how Types are used to achieve state of the art results in the task on a standard dataset.

4 Question Classification

This chapter details a purely rule-based system for Question Classification (Section 2.3) which is built on conceptual hierarchies and Types. This system is divided into two parts: The first is the extraction of relevant words from a question by use of its structure, and the second is the classification of questions based on rules that associate these words to concepts. This work achieved an accuracy of *97.2%*, close to a 6 point improvement over the previous State of the Art of 91.6% and has remained state of the art over the last two years.

The work described in this chapter was published in the paper *High Accuracy Rule-based Question Classification using Question Syntax and Semantics* in the Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (Coling 2016). 2016 (Tayyar Madabushi and Lee, 2016). The co-author of this paper is one of the supervisors of the author.

4.1 System Overview

The system presented in this work consists of three parts: **a)** extracting a Question's Syntactic Map (defined in Section 4.2.1), **b)** identifying the headword of the noun phrase in the question while handling Entity Identification and phrase detection, and **c)** using rules to map words at different positions in the Syntactic Map to identify the QC. These are further broken down into the following steps (programmatically, methods):

Syntactic Map Extraction	*Question Rewrite*	Rewrites questions that are in non-standard form.
	Parse Tree Analysis	Extract structure information from the question using Constituency-based parse trees
Word, Phrase and Entity Extraction	*Headword Extraction*	Extract headwords from noun phrases in the question using **a)** Possessive Unrolling **b)** Preposition Rolling **c)** Entity Identification
	Verb, Wh-word and Adjective Extraction	Extract the Auxiliary and Major Verbs, the Wh-word and all adjectives from the question.
Rule-based Classification	*Match Rules based on the Question Syntax and Word Type*	Using a hierarchy of syntactic positions in a question, iteratively check to see if there exists a rule for mapping the word at that position to a QC.

For example, given the question “Name of actress from England in the movie ‘The Titanic’ is what?”, our system identifies its QC as follows: We first identify that this question is not in a form that we can analyse to extract the Syntactic Map and rewrite it as “What is the name of the actress from England in the movie ‘The Titanic’?” (Section 4.2.2). The question’s parse tree is then analysed to generate the Question’s Syntactic Map (Section 4.2.1). We then identify the headword to be the noun *actress* using prepositional rolling (Section 4.3.1). At this stage, we have established that the question’s wh-word is “What”, auxiliary verb is “is”, and headword is “actress”. We check for the existence of a rule that classifies this question by iterating through these elements in a predefined order (Section 4.4.2). This results in the word “actress” matching the rule : *‘occupation.n.01’ and its hyponyms in SQ-NNP when the wh-word is ‘what’ indicate that the question class is **hum:ind***, so enabling us to classify the question as hum:ind.

4.1.1 Methodology

To avoid bias, we use the 5,500 questions and their respective question classes provided as training data by Li and Roth (2002) for exploration and rule discovery, and ensure that the 500 TREC questions, which comprise the test set, are not observed during the creation of rules (although the system is, at regular intervals, tested on this set to ensure progress). Once we complete the analysis of a question's parse tree, not all words in the question are of further relevance to the task of QC. However, so as to maximise the number of words that we have rules for, we try to create rules for all words that appear in the training set.

4.2 Syntactic Maps

Previous work that has made use of parse trees includes that by Silva et al. (2011), who used Collin's Rules (Collins, 1999) to extract headwords and work by Shen and Lapata (2007) who made use of FrameNet (Baker et al., 1998). Unlike these works, we first extract what we call a Question's *Syntactic Map*, before creating rules that depend on the position of words in this Map.

A Syntactic Map (SM), unlike a parse tree, is a *fixed* structure that we fill in with information from a question's parse tree and can contain empty or "None" elements. It is a generic template for all the different kinds of questions that we can classify, and any question that we cannot convert to a Syntactic Map, cannot be classified using our system. Crucially, the SM contains the following five elements of a question: **a)** the question's wh-word **b)** the noun phrase (if any) contained in the WHNP sub-tree and its internal phrase structure, and from the SQ sub-tree of the parse tree: **c)** the Auxiliary Verb (AVP) **d)** the noun phrase (if any) and its internal phrase structure, and **e)** the Main Verb (MVP) (if any). Noun phrases including possessives, and prepositional phrases are extracted into similar fixed structures. Programmatically, a SM is a class (object-oriented programming), as are the constituent noun phrases, prepositional phrases, and verbs. The generic structure of a SM, along with the structure of its constituents is shown in Table 4.1.

In the question "How much does the President get paid ?", it is the adverb "much" that allows us to infer that the expected answer is a number and additionally, the word "paid"

Syntactic Map		
WH Word		What/Name/Who/...
WHNP		
	NNP	Noun Phrase in WHNP
SQ		
	AVP	Axillary Verb of SQ
	NNP	Noun Phrase in SQ
	MVP	First Main Verb of SQ

Constituent Noun Phrase		Constituent Prepositional Phrase		Constituent Verb	
JJ	Adjective	PP	Prepositional word		
NN	Noun	NN	Attached Noun Phrase		
PRP	Preposition	VP	Attached Verb Phrase	VB	Verb
POS	Possessive	CPP	Attached Prepositional Phrase		
TJJ	Trailing Adjective				

Table 4.1 The fixed structure of a Syntactic Map (left), and the constituent phrase structures (right).

allows us to infer that the number, in fact, represents money hence resulting in the question class num:money.

In the questions "What is a golf ball made of ?" and "What does gringo mean ?" the verbs after the noun (the first Main Verb or MVP) provide us with important clues on which question class these questions belong to (in this case enty:substance and desc:def). It is for this reason that we move beyond conventional headword extraction and focus on populating Syntactic Maps, which capture more information about the question. Although Silva et al. (2011) consider words other than nouns, they do so only when the questions contain certain exact phrases.

4.2.1 Syntactic Map Extraction

The first step in SM extraction is the extraction of the "WHNP" and "SQ" sections of a question from its constituent parse tree, which we generate using the Stanford CoreNLP toolkit (Manning et al., 2014). The WHNP sub-tree represents the Wh-noun Phrase and the SQ sub-tree the main clause of a wh-question. In cases where there is neither (e.g. Name the highest mountain.), we use the first noun phrase as the SQ sub-tree. From the WHNP and the SQ sections of the parse tree, we extract the various elements of the SM as shown in Table 4.1. This requires the parsing of noun, prepositional, possessive and verb phrases. Due to space constraints, we only provide an overview of each of these below. Additionally, extracting each of these elements is done recursively as sentences often contain possessive phrases or prepositional phrases within one another. Table 4.2 illustrates one such scenario in which a question has two recursive possessive phrases.

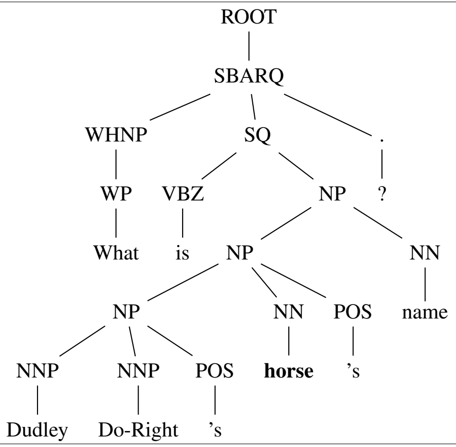
Parse Tree	**Extracted Structure**		
	WH Word		What
	WHNP		
		NNP	None
	SQ		
		AVP	['is']
		NNP	(Possessive)Dudley Do-Right (Possessive)horse name
		MVP	

Table 4.2 The Parse Tree and Extracted SM of a Question Consisting of a Nested Structure.

We make the conscious decision of stopping the SM extraction process after reaching the first main verb. This is because we observed that there were very few questions that require structural information beyond this point.

Our method of analysing noun phrases handles the extraction of adjectives, possessive phrases, prepositions and trailing adjectives but ignores all determiners. Prior to analysing parse trees of noun phrases, we first modify certain parse tree patterns that noun phrases occur in. The resultant Constituency-based parse trees are not always valid but greatly simplify the analysis of noun phrases. Two examples of the modifications we perform to noun phrase sub-trees are illustrated in Table 4.3

This simplification process leaves us with the task of extracting information from noun phrases that belong to a much smaller set of sub-tree patterns. Some of the more common noun phrase patterns are illustrated in Table 4.4. Possessive phrases are treated as nouns that must have, attached to them, yet another noun. When we identify a preposition phrase

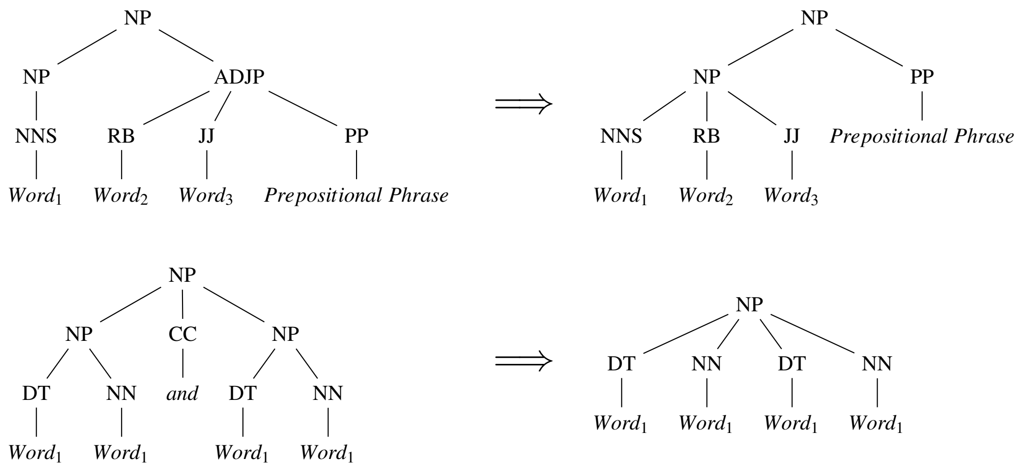


Table 4.3 Some of the Parse Tree Modifications that are Performed on Noun Phrases.

or a verb phrase, that sub-tree is passed to either the preposition or verb analysis method respectively.

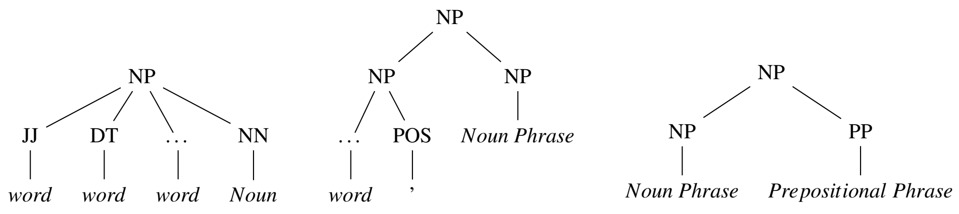


Table 4.4 Some Common Sub-tree Patterns that Noun Phrases occur in.

Similarly, we extract information from prepositional sub-trees based on their structure, which nearly always belongs to one of the following three patterns: A preposition phrase with one child that is the preposition and the other that is one of either a noun phrase, verb phrase or another prepositional phrase (e.g. "name of the prime minister of UK"). These patterns are illustrated in Table 4.5. Just as in the case of noun phrases, we pass on any sub-trees of phrases that are of a different kind to the appropriate analysis module, which enables us to generate a recursive SM.

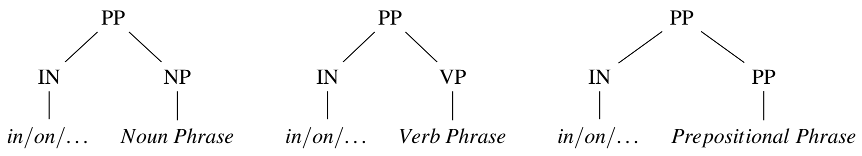


Table 4.5 Some Common Sub-Tree Patterns that Prepositional Phrases occur in.

4.2.2 Question Rewrites

There are some questions that do not belong to the standard structure of questions such as "A corgi is a kind of what?" and "In 1309 the papal court was forced to move from Rome to where?". We identify several of these structures and create rewrite rules (e.g *x is/was y in/of what z?*) to rewrite these questions to a form that we can parse. We use regular expressions instead of parse tree analysis as these structures are very easy to identify and so the overhead of parsing is not justified. Using these rules the above two questions will be rewritten as "What is a corgi a kind of?" and "To where was the papal court forced to move from Rome in 1309?".

4.3 Concept Identification

In this section, we provide details on methods we use for identifying relevant Concepts using Types (described in Chapter 3), which we extract by analysing the SM.

4.3.1 Preposition Rolling and Possessive Unrolling

Rolling and Unrolling refer to the selective moving forward through a preposition, or backwards through a possessive noun. Consider the question "What is the quantity of American soldiers still unaccounted for from the Vietnam war?" from which we extract *quantity(PP) of PP-NN:(JJ)American soldiers*, and the question "What are the different types of plastic?" from which we extract *(JJ)different types(PP) of PP-NN: plastic*. In the second instance, we must *roll* through the preposition to reach the relevant word “plastic”, whereas, in the first instance, we must not, so identifying “quantity’.

Similarly, consider the question “What game’s board shows the territories of Irkutsk, Yakutsk and Kamchatka?” from which we extract the noun phrase *(Possessive)game board*, and the question “Name Alvin’s brothers.” from which we extract *(Possessive)Alvin brothers*. In the first instance we need to *unroll* through the possessive to reach the relevant word “game”, whereas in the second case we must not. We call this selective process of moving forward through a preposition “Rolling”, and the process of selectively moving backwards through a possessive “Unrolling”. Rolling and Unrolling are achieved through a list of rules that depend on the Type of the target and source of the Roll or Unroll.

4.3.2 Headword and Phrase Extraction

Consider the question “What mystery writer penned ‘...the glory that was Greece, and the grandeur that was Rome’?”. The relevant noun phrase that we extract from the SM is “mystery writer” and the head of this noun phrase is “writer”, the last noun in the noun phrase. This is often the case, and some previous works have used only this to identify the head of a noun phrase (Metzler and Croft, 2005). Unfortunately, this is not always the case, and does not always provide the word that is most useful for QC. For example, the noun phrase extracted from “What crop failure caused the Irish Famine?” is “crop failure” and the relevant noun is “crop”. Although it can be argued that the head noun in this phrase is “failure”, qualified by “crop”, this would not aid us in classification, as “crops” are a form of food and the expected Question Class is enty:food, while “failure” is a very different Concept.

We automatically identify the head noun by identifying *Verb Nouns* and *Descriptive Nouns* starting at the right of the noun phrase and ignoring such nouns. We define Verb Nouns as nouns that have a more common verb form (e.g. fail) or verbs that are “acts”, which we identify by parsing the definition of the verb. Similarly, we define Descriptive Nouns as nouns that belong to a Type we define as descriptive which includes, for example, hyponyms of the synyset 'digit.n.01'.

4.3.3 Entity Identification

Let us now consider the question “What is bipolar disorder?”. The correct Question Class for this question is desc:definition, however, it is easy to miss-classify this question as belonging to the class enty:dismed (entity, disease or medicine), because the word “bipolar” is tagged as an adjective. To get around this we require a method of identifying that “bipolar disorder” must be considered as a single entity.

Even in instances wherein it is relatively easy to identify an entity, as in the case of phrases that consist of consecutive nouns, it is important to be able to convert these phrases to a form that appears in WordNet. For example, the phrase “equity securities” can be identified as a single entity, however, it is listed in WordNet under the entry “shares”.

We identify these phrases using a method called Wikification (Mihalcea and Csomai, 2007), which is the process of linking words and phrases in a piece of text to titles of Wikipedia entries. The intuition behind this is that a phrase that appears as a Wikipedia Article title must be important enough to be considered as a single Entity. We base our method of Wikification on the original, while replacing the process of keyword identification with SM and that of Word Sense Disambiguation with the method detailed in Section 4.4.1. For example, there is an article on Wikipedia titled “Bipolar Disorder” on Wikipedia and the Wikified term for “equity securities” is “Shares”.

4.4 Question Classification using Syntactic Maps

Once we have the SM of a question, we use rules to identify the relevant QC. However, before we can match appropriate words, we require a way of identifying the correct sense of a word.

4.4.1 Word Sense Disambiguation

SMs often provide us with a single word that represents the object that the question expects as an answer. The question “What album put The Beatles on the cover of Time in 1967 ?”, for example, requires that the answer consists of an “album”. However, it is unclear whether album refers to “one or more recordings issued together” or “a book of blank pages with pockets or envelopes”. Huang et al. (2008) address this problem by use of the Lesk Algorithm (Lesk, 1986).

Our use of SM allows for implicit Word Sense Disambiguation as it is rare for the same word to appear at the same syntactic location but in different senses. When this does happen however, we identify the sense of a word based on the Types of the surrounding elements of the SM. For example, “How much does it cost to fly to Japan?” and “How much does a plane weigh?” both have the word “much” at the same position and so require us to identify the Types of associated words (i.e. “cost” and “weigh”) to be able to disambiguate the relevant Concept.

4.4.2 Mapping Question Classes

The intuition behind the mapping process is that words or phrases at certain positions in the SM trigger certain Concepts, which reveals the question class. To this end, we use Types defined for each different position in the SM to map questions to question classes. For example, the word “do” appearing as the auxiliary verb is handled differently from when it appears as the main verb in the SM. The order in which different sections of the SM are considered determines which word is finally used during classification.

There are some special words, such as “much”, “do”, “name” and “call”, that require more complex classification rules. The adjective “much” for example could indicate the class num:money or num:weight depending on whether the other sections of the SM contain the Type “money” or the Type “weight”. As in the case of WSD, we define disambiguation rules for each such word.

Algorithm 2, while not exhaustive in listing the mapping rules (due to space constraints), provides a simplified overview of the mapping of Semantic Maps to Question Classes. It takes as input the SM, the Type definitions and associated Question Classes and returns a tuple consisting of the Major and Minor question classes. Just over 230 Type definitions and 10 special Word Sense Disambiguation definitions cover the entire test set, and at the time of writing, these have been expanded to around 600 Type definitions and 70 WSD definitions.

Algorithm 2 can broadly be divided into three parts: The first identifies the noun phrase most likely to be useful in finding the class of the question before then extracting a list of words, ordered by priority, that are used for this classification (up to line 24). Once this is done, the second part (loop detailed between lines 25 and 27) then iterates through all of these words so as to identify the relevant class. The last part of the algorithm checks to see if the question contains non-nouns (such as in questions starting “How many ...”) which overwrite the question class determined so far. If all else fails, the algorithm returns Entity:Other as the default class (line 44).

4.5 Results

We achieve an accuracy of 97.2% on the TREC 10 dataset which translates to an incorrect tagging of 14 of the 500 questions in the dataset. This is close to a 6 point improvement over the previous state of the art of 91.6% (Van-Tu and Anh-Cuong, 2016). We list our accuracy against that of various other works that have reported results on the TREC 10 dataset in Table 4.6.

Study	Classifier	Accuracy	
		Coarse	Fine
This Work	**None**	**-**	**97.2%**
Van-Tu and Anh-Cuong (2016)	Linear SVM	95.2%	91.6%
Pota et al. (2016, 2015)	Linear SVM	89.6%	82.0%
Kim (2014)	Convolutional Neural Networks	93.6%	-
Kiros et al. (2015)	Skip-Thought Vectors	91.8%	-
Silva et al. (2011)	Linear SVM	95.0%	90.8%
Loni et al. (2011)	Linear SVM	93.6%	89.0%
Merkel and Klakow (2007)	Language Modelling	-	80.8%
Li and Roth (2006)	SNoW	-	89.3%
Li and Roth (2002)	SNoW	91.0%	84.2%

Table 4.6 Results Achieved by this Work alongside some other Works that use the same Dataset.

4.5.1 Error Analysis

Table 4.7 provides a list of some of the questions that we misclassify along with the reason for this. One of the advantages of a purely rule-based system is the ability to pinpoint the exact reason for an incorrect classification.

4.6 Summary

This chapter described a method of classifying questions based on the class of the expected answer using Types introduced in the previous chapter (Chapter 3). Our experiments showed that this new method outperforms all previous methods, highlighting the effectiveness of Types.

The next chapter makes use of a modified version of this system to outperform the current state of the art in Answer Selection, a task introduced in Section 2.4.

```
Data: Syntactic Map, Type Definitions, Classes associated with Type Definitions.
Result: Question Class
1  if Preposition Rolling Possible then
2      Perform Preposition Roll
3  if Possessive Unrolling Possible then
4      Perform Possessive Unroll
5  Initialise head_noun_class to None ; /* head_noun_class is a Tuple Consisting of the
     Major and Minor Question Type                                              */
6  head_noun ← Extract Head Noun from Syntactic Map ;
7  head_noun_adjectives ← Extract Head Noun adjectives from Syntactic Map ;
8  for reversed( head_noun_adjectives ) do
9      if adjective has Type Defined then
10         head_noun_class ← Class associated with Type;
11 if head_noun_class is None then
12     if head_noun has Type Defined then
13         head_noun_class ← Class associated with Type;
14 if head_noun_class[0] == "ABBR" then
15     if head_noun is an Abbreviation then
16         return ( 'ABBR', 'exp' )
17     return head_noun_class
18 if All of the following elements in the Syntactic Map are Empty: WHNP-NNP, SQ-MVP,
     head_noun_adjectives then
19     if There has been no Rolling or Unrolling then
20         if AVP is one of "is", "are", "was", "were" then
21             if WH_Word is "What" then
22                 return ('DESC', 'def')
23             if WH_Word is "Who" then
24                 return ('HUM', 'desc')
25 for reversed( head_noun_adjectives ) do
26     if adjective has WSD Type Defined then
27         return Class associated with WSD Type;
28 wh_word ← Extract What Word from Syntactic Map ;
29 if wh_word == "define" then
30     if head_noun_class[0] == "DESC" then
31         return head_noun_class
32     return ( "DESC", "def" )
33 if wh_word == "how" then
34     if head_noun_class[0] == "DESC" then
35         return head_noun_class
36     return ( "DESC", "manner" )
   /* Similar restrictions are imposed on other possible wh_words (i.e.
      ''where'', ''whose'', ''describe'', ''when'', ''why'', ''name'', and ''what'')  */
37 main_verb ← Extract Main Verb from Syntactic Map ;
38 auxiliary_verb ← Extract Auxiliary Verb from Syntactic Map ;
39 for verb in [ main_verb, auxiliary_verb ] do
40     if verb has Type Defined then
41         return Class associated with Type;
42     if verb has WSD Type Defined then
43         return Class associated with WSD Type;
44 if head_noun_class is None then
45     return ("ENTY", "other")
46 return head_noun_class
```

Algorithm 2: A Simplified Algorithm showing the Mapping of the Syntactic Map to Question Classes

Question	Correct Class	Classified As	Reason
What are the twin cities?	LOC city	DESC def	We classify both these as definitions because we (correctly) identify “twin cities” and “speed of light” as entities. The presence of the word “the” however requires information about the entity instead of a definition for the entity - a rule that requires to be added.
What is the speed of light?	NUM speed	DESC def	
What is compounded interest?	DESC def	DESC desc	Our Wikification system fails to identify “compounded interest” to be the same as the entity “compound interest”.
What is the spirometer test?	DESC def	ENTY instru	The word “test”, has a natural verb form so forcing the system to identify “spirometer” as the head noun. Some modifications to the function identifying Verb Nouns are required to rectify this.

Table 4.7 An analysis of some of the questions that we fail to classify correctly.

5 Answer Selection

This chapter presents work on a system for Answer Selection (Section 2.4) that integrates fine-grained Question Classification (Section 2.3) with a deep learning model designed for explainable Answer Selection. We detail the necessary changes to the Question Classification taxonomy and system, the creation of a new Entity Identification system and methods of *highlighting* entities to achieve this objective. Our experiments show that Question Classes are a strong signal to deep learning models for Answer Selection, and enable us to outperform the current State of Art in all variations of our experiments except one. In the best configuration, our MRR and MAP scores outperform the current State of Art by between 3 and 5 points on both versions of the TREC Answer Selection test set, a standard dataset for this task.

Work on this system was published in the paper "Integrating Question Classification and Deep Learning for Improved Answer Selection" that was published in the Proceedings of COLING 2018 (Tayyar Madabushi et al., 2018). The co-authors of this paper are the author's supervisors.

5.1 System Overview and Contribution

In working towards a method of integrating QC with AS, we first redefine the taxonomy provided by Li and Roth (2002) to better suit entity identification, before then modifying the Question Classification system developed by Tayyar Madabushi and Lee (2016) to match this modified taxonomy. We then create an entity identification method to extract entities belonging to those classes in our taxonomy. Finally, we use different methods of "highlighting" entities, so this information can be passed on to *any* model that uses word embeddings. We use the model developed by Rao et al. (2016), which performs AS, to test our method.

In addition to showing the significant impact that Question Classification has on Answer Selection, we make several datasets available so others might exploit QC in Question Answering tasks including a Question Classification API that reflects the modified taxonomy.

5.2 Question Classification

Our experiments with using the taxonomy proposed by Li and Roth (2002) showed the need for changes to allow the classification system to lend itself more easily to Entity Identification and AS. For one, we found that some categorisations would make entity identification harder. For example, the question "What's the world's longest suspension bridge?" is categorised under "Location" while we believe that it is more appropriate to consider a bridge an entity. We base this on the hierarchical classification provided by WordNet (detailed in Section 3.2).

Similarly, we disagree with the prioritisation of the classes provided. Prioritisation is important as this particular taxonomy does not allow a question to be a part of two classes. As an example, the question "What country did the ancient Romans refer to as Hibernia?" can be classified as either belonging to the class "Location:Country" or "Entity:termeq" (Equivalent Term). While Li and Roth (2002) categorise this question under the first, we categorise it under the latter, because the question is not about where something is or happens. We also believe that this choice makes it easier for a downstream QA system.

We also found the need for a new class that constitutes either "Human:Individuals" *or* "Human:Groups" (such as companies, teams and universities). This specific requirement is a direct result of the restriction that a question must be classified without prior knowledge of the answer. For example, the question “Who won the Nobel Peace Price in 2012?” is impossible to classify without knowing if the answer was an organisation (as it was in 2012) or an individual (as in 2016), even if we were to ignore the possibility of multiple individuals (as in 2014). To get around this we introduce the class “Human:IndividualOrGroup”. We retain the classes “Human:Individuals” and “Human:Groups” for instances where the distinction is clear.

Finally, we find that certain types of entities within certain classes are much more frequent than others in that class. While this could be because of the specific method we use for Entity Identification (Section 5.3), we create separate classes for these types of entities so as to avoid noise in our AS feature generation. We also expand the class “Location:State” to include the provinces of Canada and the counties of the U.K. We list the taxonomy thus modified in Table 5.1.

Coarse	Fine
ABBR	abbreviation*, expansion*
DESC	definition*, description*, manner*, reason*
ENTY	animal, body, colour, creation, currency, disease, event, food, instrument, language, letter*, other*, plant, product, religion, sport, substance*, symbol*, technique, term*, vehicle*, word*, **movie***, **book***, **extraterrestrial**
HUM	description*, group, individual, title, **individualOrGroup**
LOC	city, country, mountain, other, **state**
NUM	code, count, date, distance, money, order, other*, percent, percent, period, speed, temperature, size, weight, **year**, **volume (Size)**, **volume (Liquid)**, **time**, **numeric range***

Table 5.1 Question Taxonomy introduced by Li and Roth (2002), with our modifications in bold (Section 5.2.1) and those classes not used in AS starred* (Section 5.7)

5.2.1 Modifications to Question Classification

The QC system used by us is an extension of the one presented in Chapter 4. It primarily involves: **a)** extracting a Question's Syntactic Map (a structure defined for holding certain types of syntactic information), **b)** identifying the headword of the noun phrase in the question, while handling Entity Identification and phrase detection, and **c)** using rules to map words at different positions in the Syntactic Map to question classes using a hierarchical structure.

The QA system classifies questions as follows: Consider the question "What is the name of the actress from England in the movie 'Titanic'?". The system identifies its question class as follows: The question's parse tree is analysed to generate the Question's Syntactic Map, which enables the identification of the headword actress using, what they call, "prepositional rolling". This process provides us with the question's wh-word ("What"), the auxiliary verb ("is"), and headword ("actress"). This information is used by the system to check for the existence of a rule that classifies this question. Such a rule is found by matching the noun "actress" to the rule: 'occupation.n.01' and its hyponyms in this section of the question when the wh-word is 'what' indicate that the question class is hum:ind.

These rules are manually defined using sets of WordNet synsets they call Types. Types are defined by manually picking specific synsets within WordNet and associating them and all their hyponyms to a particular question class based on where in a question they appear. In the previous example, the relevant Type is the word occupation and all hyponyms of the synset 'occupation.n.01'. Similarly, the synsets 'people.n.01', 'organization.n.01', 'university.n.01', 'company.n.04', 'socialgroup.n.01', and all of their hyponyms are assigned to the question class "Human Group".

We describe below some elements of the system described in Chapter 4 which were subsequently modified, a necessity given our changes to the taxonomy.

5.2.2 Word Sense Disambiguation and Rule Extensions

A primary difficulty in identifying the specific rule to use once the correct head of the question has been identified arises due to the polysemous nature of some words. For example, the question “What rank did you achieve in the test?” and the question “What rank did she achieve in the military?” both have the same headword “rank” but differ in the meaning of that word (position in ordering versus military status such as captain). The question class assigned to each of these question must also change based on these meanings (Number:order versus Human:title).

As described in Chapter 4, words useful in identifying the question class are often nouns, as in the case of the question “What is the name of the *actress* in the Titanic?”. However, such words, useful in defining the question class, need not always be a noun. In the case of the question “How much does the President get paid?”, for example, it is the adverb “much” which allows us to infer that the expected answer is a number and additionally, the word “paid” allows us to infer that the number represents money hence resulting in the question class “number:money” as opposed to the question class “number:weight” as in the case of “How much does the Big Ben weigh?”

Rules defined by the QC system map sub-trees in WordNet to specific question classes. We make changes to the rules to align the classification of questions with the modifications we make to the taxonomy (Table 5.1) and add further rules where possible to cover a larger section of WordNet. Additionally, there are instances wherein the system makes use of certain heuristics to find the appropriate rule to use, as in the case of questions starting with “How much . . . ” which sometimes leads to classification errors. To mitigate this problem, we modify the system to return a possible second class when there is ambiguity.

5.3 Named Entity Recognition

Given that our objective is to "highlight" all entities in candidate answers that belong to the class assigned to a particular question, we require a method of Named Entity Recognition (NER) at the same granularity as our taxonomy. Unsurprisingly, there is no off the shelf NER system that identifies entities with the exact granularity and classes that we classify questions into. To get around this problem we start by relating entities in text to Wikipedia titles and subsequently mapping those titles to our classes. This process of mapping entities in text to Wikipedia titles is called Wikification. The hierarchical tree-structure provided by Wikipedia helps in mapping a large number of titles to a given class by allowing us to map sub-trees to classes.

5.3.1 Wikification

Wikification was introduced by Mihalcea and Csomai (2007), as a means of automatic keyword extraction and Word Sense Disambiguation. It has since been used for a variety of tasks especially the semantic enrichment of text. A significant advantage of using Wikification is that entities, once identified, are in a normalised format, namely the title of the linked Wikipedia article, thus making entity matching (Section 5.4) easier.

While simple entity identification involves the direct matching of phrases to Wikipedia titles, more advanced versions of Wikification additionally involve mapping phrases to *related* titles based on the contents of the Wikipedia article. For example, one might choose to map the phrase "the first Briton in space" to the Wikipedia article on "Helen Sharman". We however, limit ourselves to the simpler version as we are only interested in finding entities and not concepts.

Typically, Wikification involves the identification of potential entities and the subsequent matching of those entities with Wikipedia titles. For example, given a sentence, one could potentially use a Parts of Speech tagger to tag the sentences before then extracting sequences of PoS tags that match a predefined set (such as NNP+ or DT*NNP+ and so on). Entities thus extracted could then be matched with Wikipedia titles.

After experimenting with several off the shelf Wikification tools, we found them lacking in the ability to work with Wikipedia Disambiguation pages and topic specific pages. For example, when looking for entities of type "movie" in the sentence "He went to watch the movie 'New York"', we want to be able to match this to the Wikipedia article "New York (film)" and *not* "New York (state)" or "New York City".

5.3.2 Wikification without PoS Tagging

The obvious way to tag entities in text with Wikipedia titles would be to match every possible phrase in a sentence with every title on Wikipedia. This, however, is impractical as there are over 13.04 million titles in the English Wikipedia. To get around this we run through the titles, and for each title, we split it into its constituent words and save the rest of the title in a file whose name is the first word. Thus, all titles that begin with a particular word are clubbed into a single file and for those titles that are of length one, we add an empty line into the corresponding file. This results in just over 2.1 million files each of which are relatively short and easy to process.

As we sweep through each word in a sentence, we process the file containing Wikipedia titles starting with the same word, and check to see if it contains entities that match the current sentence. This greatly speeds up the process of matching titles to the words in a sentence and provides us with a list of titles that are contained in a given sentence.

We note that this method of Wikification can be used in languages where capitalisation is dissimilar to English or even those languages wherein there is no capitalisation.

5.3.3 Wikification to Question Classes

Once candidate answers are Wikified, we are then left with the task of mapping these titles to the question classes. We do this by first linking each Wikipedia title to the corresponding DBPedia entry. DBPedia is an attempt to extract structured information from Wikipedia and provides a list of labels and classes associated with each entry. We use these labels and classes to map Wikipedia (and so DBPedia) titles to question classes associated with our taxonomy.

5.3.4 NER without Wikification

We use the Stanford Named Entity Recogniser (Finkel et al.) to identify entities belonging to the classes "Human Individual", "Human Group" (such as institutions, universities, etc.), and "Location Other", the three classes with compatible granularity. All numeric entities, such as Number:Money, Number:count, and Number:date are identified using an extensive list of regular expressions.

5.4 Entity Matching

When grading papers, a good maxim to identify plagiarism is "While there is only one way to get it right, there are several ways to get it wrong". We observe that this maxim works because the probability of two students answering a question incorrectly in *the same way* is extremely small, unless of course it's a trick question. Similarly, the chance of candidate answers having the same incorrect entity that also match the class of the question is exceedingly small and machine learning models can make use of this information. To this end, we count the number of occurrences of each entity across all answer candidates of a given question. This requires us to be able to match entities that have been written differently, but are in fact the same.

Entities that have been extracted through Wikification are often normalised “for free.” However, there is no simple way to get around this problem in the case of entities extracted through regular expressions, as in the case of numbers and dates where it is common for sentences to contain approximations. For example, consider the question “How many lives were lost in the air-crash?”, the answer is contained in all of the following sentences: “253 lives were lost in the air-crash”, “241 passengers and 12 crew died in the air-crash”, and “around 250 lives were lost in the air-crash”. This problem is further expanded when the numbers we are dealing with become larger as it is more common for non-technical literature to approximate large numbers. To get around this we round down all numbers to the nearest billion, million, hundred thousand, thousand, hundred or ten.

5.5 A Shallow Model for Answer Selection

This section explores a shallow model for AS which follows treditional methods in NLP. It makes by making use of a modified QC system and a case insensitive entity extraction system which is used to classify entities into the relevant question classes. Using only this and a very basic overlap percentage, a SVM is used to achieve Mean reciprocal rank (MRR) and Mean average precision (MAP) scores of 0.82 and 0.72 on the TREC dataset.

5.5.1 Training Data

One of the problems we face is in the lack of training data in a format that is usable. Unfortunately, the clean training data (the training data cleaned as described below), which has been shown to produce better results does not contain enough data split reasonably across the question classes that we use. To get around this we extract 100 training instances from the raw automatically tagged data while ensuring that the questions we pick have a reasonable distribution across our question classes. Each of these instances are then manually cleaned to ensure that all incorrectly tagged sentences, which exist in this dataset due to the automated creation method used to generate it, are removed.

5.5.2 Model Details

So far our work has focused on classifying questions and then extracting, from candidate answer sentences, all entities that belong to that class. For example, if we are given the sentence "When did the Beatles release their first album?", we would first identify that this question belongs to the class "Number:date" and subsequently use methods described in Section 5.3 to extract all dates from candidate sentences. In this section, we describe the model used to actually select specific sentences as containing the answer.

Baseline model

Before moving forward with using question classes for AS, we create a baseline word overlap model. The model simply assigns the percentage overlap of words in the question and a given candidate answer as the likelihood of that candidate answer being the one containing the answer. There is however, one caveat: Instead of using every word contained in the question, we only use those words that are not stop words and have a Wikipedia title associated with them. We note that this isn't much of a restriction as nearly every word has at least a disambiguation page on Wikipedia. We evaluate this model using the same *trec_eval* program and it achieves an MRR score of 0.66. This low score is expected as this particular dataset is designed to require systems more complex that those using simple bag-of-words. Again as expected, this model performs more poorly than the baseline models presented by Wang et al. (2007) when introducing this task.

Extracting Features from Entities

Given a particular question, and a corresponding set of answer candidates we have so far generated a list of entities that match the question type in each answer candidate along with the percentage of overlap between the question and each of the answer candidates.

Using this we create the following features for each answer candidate: 1. The number of answer candidates available for the question under consideration. 2. the number of answer candidates that contain entities of the relevant type. 3. the maximum overlap percentage of answer candidates that contained entities. 4. whether or not an entity that is part of the answer candidate with the highest overlap is contained in this answer candidate. 5. the overlap percentage 6. the maximum overlap percentage (including answer candidates without any entities) 7. whether or not this answer candidate contains the most frequently occurring entity. 8. whether or not this answer candidate contains the entity with the highest average overlap across all answer candidates.

Additionally we find the entity contained in the answer candidate under consideration with the maximum occurrences across all answer candidates and add the following features: 9. The number of occurrences of this entity 10. the percentage of all answer candidates that this entity occurs in 11. the percentage of answer candidates with entities that this entity occurs in 12. the total overlap percentage of all answer candidates that this entity occurs in 13. the total overlap percentage of all answer candidates that this entity occurs in divided by the total number of answer candidates. 14. the total overlap percentage of all answer candidates that this entity occurs in divided by the number of answer candidates with entities. 15. the total overlap percentage of all answer candidates that this entity occurs in divided by the number of answer candidates with this entity. 16. the maximum overlap percentage of all answer candidates with entities

SVM Parameters

We train a simple SVM with just these sixteen features and a linear kernel. We use the SVM model provided by scikit-learn (Pedregosa et al., 2011) with C, the penalty of the error term set to 70, and the tolerance for stopping set to 0.0001. We additionally enable probability estimates, which are calculated using Platt scaling. scikit-learn documents this as the logistic regression on the SVM's scores, fit by an additional cross-validation on the training data.

We note that not all question classes have entities that can easily be identified. For example, it is extremely difficult to extract relevant entities for questions of the class "Entity:Other" as *every* entity is a potential candidate, and impossible for the class "Description" (as there is no entity but a description that is being sought). For such classes, we simply pass on the baseline (percentage of word overlap) as the score for a given candidate sentence.

5.5.3 Results of the Shallow Model

We list the results achieved by our method alongside prior results on the same dataset in Table 5.2. We note that unlike Wang and Ittycheriah (2015), our sentence similarity is based on an extremely weak baseline, so as to measure the true impact of QC. The baseline achieves scores of 0.6113 and 0.6686 for MAP and MRR respectively and the system achieves scores of 0.7186 and 0.8164 for MAP and MRR. Except for the work by Wang and Ittycheriah (2015), all the other systems use a models based on word embeddings.

When no Entities Exist

As noted earlier, not all questions classes have entities that can easily be identified (Section 5.5.2). So as to measure the accuracy of this system on those questions we can identify entities for and those that we cannot, we report the MAP and MRR scores for only those questions that we do identify entities for.

Reference	MAP	MRR
Baseline	0.6113	0.6686
All Data	0.7186	0.8164
Only Entities	0.7151	0.8113
Pruned Data	0.6978	0.7975
All Data Fixed	0.7284	0.8276
Wang and Ittycheriah (2015)	**0.746**	**0.820**
Tan et al. (2015)	0.728	0.832
dos Santos et al. (2016)	0.753	0.851
Wang et al. (2016)	0.771	0.845
He et al. (2015)	0.777	0.836
Rao et al. (2016)	0.801	0.877
Wang et al. (2017)	0.802	0.875

Table 5.2 Results from this work, including "All Data Fixed", which represent results for the corrected test data (Section 5.5.3), alongside prior results on the same dataset, with comparable methods in bold.

Surprisingly, we achieve lower scores of 0.7151 and 0.8113 for MAP and MRR, when considering only those question for which we identify entities. This implies that the weak baseline is actually a good method for Answer Selection for questions that belong to certain classes. We believe this to be a possibly useful guide in focusing future research into Answer Selection and so Question Answering.

Eliminating Answer Bias

When picking sentences to include as answer candidates, Wang et al. (2007) search for all sentences that either contain any non-stop word in the question or any of the possible answers. Since we use the number of entities as one of our features, it is important to ensure that the method of extracting candidate sentences does not skew our results.

To ensure we avoid this bias, we first extract all entities from the questions, and then prune the candidate sentences for each question based on whether or not they contain the

the entities identified. We make exceptions for those entities that are implied as a result of a perspective bias in the data. For example, the location “United States” is often missing but assumed.

With these modifications, we achieve MAP and MRR scores of 0.6978 and 0.7975. The fact that the scores drop is not surprising, however, it must be pointed out that other systems enjoy the same advantage.

Data inconsistencies

There are some questions and answer candidates that are incorrectly tagged in the Answer Selection test set. For example, the question “What years did Sacajawea accompany Lewis and Clark on their expedition?” has, amongst its candidate sentences, the following two sentences: “the coin honors the young woman and teen-age mother who accompanied explorers Meriwether Lewis and William Clark to the pacific ocean in 1805”, and “in 1804, Toussaint was hired by Lewis and Clark, not for his own skills but for those of Sacagawea.” While the first candidate answer is marked as one containing the answer the second is not marked as such.

While this is the only error of this kind, we found several other sentence candidates similarly marked as containing the answer when they do not contain important elements of the question. For example, the question “How long are Syrian presidential terms?” has the positively marked candidate sentence “parliament formally announced that bashar assad ’s swearing-in ceremony will be on july 17 , after which he will embark on a seven-year term that is full of risks.” While this does contain the answer, it requires a step in reasoning which we believe takes away from what is being tested here. We release a full list of such sentences from all test sets.

Restrictions Impossed on the Shallow Model

We note that we intentionally do not include additional methods to attempt to improve the performance of this system. As an example, we could replace the simple percentage overlap used in this work with the sentence alignment used by Wang and Ittycheriah (2015), or a similarity measure based on word vectors as do Yu et al. (2014). Similarly, we could have heavily exploited phrases contained within candidate sentences to further boost our performance. We work with these limitations so as to measure the true impact of QC.

5.5.4 Error Analysis - the Shallow Model

One source of errors stems from our choice of Entity Identification. Since we limit ourselves to Wikipedia entities, no entities are identified for the question “Whom did Eileen Marie Collins marry?” as the positive answer candidate contains the name “pat youngs”, a name we do not find on Wikipedia. We note that this would not have been a problem, had the answer candidates been correctly capitalised. Additionally, this error could have been avoided if the answer candidate contained his full name “patrick youngs”, as that would have enabled us to identify Patrick as a common given name (Section 5.3.2).

Other errors stem from an expected source: The use of a weak similarity measure. Yu et al. (2014) point this out when detailing examples that their system can handle, but pure bag-of-words systems cannot: Consider the question “When did James Dean die?” and the two corresponding answer candidates “In [date], actor James Dean was killed in a two-car collision near Cholame, Calif.”, and “In [date], the studio asked him to become a technical adviser on Elia Kazan’s ‘East of Eden’ starring James Dean”. Our system is unable to distinguish between the two as both contain entities of the required answer type, forcing us to rely on the weak similarity measure.

Interestingly though, we are successful when working with an example that they say they are unable to work with due to the requirement of deep understanding: Consider the question

"What is the name of Durst's group?" and the corresponding answer candidate, "Limp Bizkit lead singer Fred Durst did a lot before he hit the big time". Our system identifies the question class of this question, based on the head noun *group*, as "Human:Group", and subsequently extracts the entity "Limp Bizkit" as an entity of the corresponding type.

5.6 The Deep Learning Model

In creating the deep learning model, we use the Answer Selection model developed by Rao et al. (2016) who rank candidate sentences using a Multi-Perspective Convolutional Neural Network (He et al., 2015) and a triplet ranking loss function which uses triplets of the question paired with a positive and a negative candidate answer. While other methods model this problem as a pointwise classification problem (He and Lin, 2016; Severyn and Moschitti, 2015), this method models the problem of Answer Selection as a pairwise ranking problem. This involves developing representations for positive and negative answer candidates paired with the question, the primary reason for us choosing this model.

Yet another advantage of this method is that it can make use of existing pointwise models to generate representations which can then be fed into the triplet ranking function. The authors make use of two such pointwise models, one that uses a sentence-level model (He et al., 2015) and the other that uses a word-level model (He and Lin, 2016). We refrain from elaborating on these methods and refer to the reader to the original works.

We make use of the sentence-level model[1] and introduce new representations for entities (Section 5.6.1) which requires us to modify them with answer candidates. The model is additionally initialised with the GloVe word embeddings (Pennington et al., 2014) which are also updated during training.

[1]https://github.com/castorini/Castor

5.6.1 Highlighting Entities

Having extracted and normalised entities that are contained in each of the answer candidates, we are faced with the task of highlighting these entities within the answer candidates. Before we do this however, we perform some prepossessing steps. We discard any entities that also appear in the question as such entities are unlikely to be the answer. For example, the question "Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher"' has, as an answer candidate, the sentence "The Iron Lady; a biography of Margaret Thatcher by Hugo Young" in which both "Margaret Thatcher" and "Hugo Young" are entities that match the question class, namely "Human Individual". The entity "Margaret Thatcher", however, is discarded as it is also contained in the question.

For each question we count the number of occurrences of each entity across all candidate answers and if the most frequently occurring entity occurs more than twice (which was empirically determined) the number of times the second most frequently occurring one, we pick the first as the maximal entity. For those questions where this is not the case, we pick no maximal entity.

We also create four new "words", *max_entity_left*, *max_entity_right*, *entity_left*, and *entity_right*, which are strings that are not contained in the vocabulary, along with associated word vectors which are randomly initialised with entries between -0.05 and 0.05 and are of the same length as the GloVe word embeddings (300). We then add these embeddings to our embedding dictionary and the words to the vocabulary.

Entities are highlighted in the answer candidates by inserting the words *max_entity_left* and *max_entity_right* on either side of maximal entities, and *entity_left* and *entity_right* around other entities. We also include *entity_left* and *entity_right* at the end of the question and the "words" *max_entity_left* and *max_entity_right* at the end of questions that contain maximal entities. We call this method of highlighting *bracketing*.

A second method of highlighting entities in candidate answers is to replace the entity with a word, a method we call *replacing*. To avoid creating two new words for this method, we reuse two of the four words used above: *max_entity_left* and *entity_left*. Table 5.3 details the modifications made to an example question and candidate answer using each of the above methods.

Method	Question	Answer Candidate
Original	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher'	in 'The Iron Lady,' Young traces the winding staircase of fortune that transformed the younger daughter of a provincial English grocer into the greatest woman political leader since Catherine the Great.
Bracketing	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher' *max_entity_left max_entity_right entity_left entity_right*	in 'The Iron Lady,' *max_entity_left* Young *max_entity_right* traces the winding staircase of fortune that transformed the younger daughter of a provincial English grocer into the greatest woman political leader since *entity_left* Catherine the Great *entity_right*.
Replacing	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher' *max_entity_left entity_left*	in 'The Iron Lady,' *max_entity_left* traces the winding staircase of fortune that transformed the younger daughter of a provincial English grocer into the greatest woman political leader since *entity_left*.

Table 5.3 Entities *highlighted* in answer candidates using two different methods. The example assumes that the entity "Young" is a maximal entity.

5.7 Empirical Evaluation

Having described our method of Question Classification, Entity Identification and Entity Highlighting, we next evaluate our method on the task of AS using different Highlighting methods and training data.

The training set commonly used for this task consists of two sets: The first consists of one hundred manually examined questions and corresponding answers candidates and the second, an automatically generated set consisting of just over 1200 questions. We found the manually inspected, and hence higher quality test set to be too small for use in this task. However, we also found that the automatically generated training set contained several inconsistencies.

To prevent noise, we discarded any questions with answer candidates that contained more than one false positive. We similarly discarded questions from the development set. Thus cleaned, we were left with 1164 questions in the training set and 76 and 60 questions in the Raw and Clean versions of the development set respectively. The development data is what the learning model is optimised on before the best performing model is used to evaluate the test data.

To ensure that our results are comparable to those published by others, we make no changes to the test data.

Some question classes cannot have entities highlighted, as in the case of "Description" and "Definition". Some other question classes do not have Entity Identification implemented as we found it impossible to identify all possible elements of the class, as in the case of "Vehicles". We call such questions unhighlighted questions and the rest highlighted questions.

For each of the Clean and Raw versions of the data, we run the model on **a)** unhighlighted data, **b)** data highlighted using *bracketing*, **c)** data highlighted using *replacement*, **d)** unhighlighted data and highlighted data using *bracketing* combined, and **e)** unhighlighted data and highlighted data using *replacement* combined. In cases where we split the data into highlighted and unhighlighted sections, the results are combined to find the MRR and MAP scores of the complete data. The model on data without entities highlighted (as presented in Rao et al. (2016)) is the baseline. We also calculate the MRR and MAP scores for the baseline for *each* of these variations as we change the training data in each case. We use the same hyper-parameters as those provided in the implementation of the work by Rao et al. (2016). We include the highlighted versions of the training, development and test data for each of the variations above along with details of the hyper-parameters used, the trained models and the output as part of the supplemental material. We present our results in Table 5.4.

5.7.1 Result Analysis

The use of question classes embedded in candidate answers *outperforms the current state of art in literature in every case except one*. This result (Number 10 in Table 5.4) is an anomaly that we attribute to over-fitting as we perform no hyper-parameter tuning. The strong performance of the baseline on the unhighlighted data (Sr. No. 1) is expected as answer candidates that are descriptive in nature (as is the case for questions belonging to the classes “Description”, “Definition”, etc., which also do not have entities identified) must necessarily have a larger overlap with the question. Once again, we ascribe the low performance of the corresponding unhighlighted baseline on the Clean Version (Sr. No. 8) to over-fitting.

The highlighting method of *replacement* performs better than *bracketing* except in the case of the anomaly. We believe this is because Named Entities, with their limited frequency, carry little information. Additionally, the replacement of entities that are phrases with a single frequently occurring word could improve sentence representation.

We expected the combination of the model independently trained on unhighlighted and highlighted data to perform better than that trained on the combination. While this is the case for the Raw version of the data, it is not the case for the Clean version. As in the case of the anomaly (Sr. No. 10), it is impossible to say if this is a result of over-fitting without performing hyper-parameter tuning on all ten of the models we present.

We also experimented with training different models for each of the course classes. We did this by extracting subsets of the training, development and test sets belonging to each of the course classes (“HUM”, “LOC”, . . .), training the model on the training subset optimised on the development subset, and testing it on the test subset. We found these results to be surprisingly low, and believe this to be a result of deep learning models gaining more from increased data rather than homogenous data. Homogeneity in data might, in fact, lead to overfitting and hence be detrimental. These results are consistent with results in

work by Kyashif (2018) who used the QC system described in this work in task-based and commonsense QA, where a similar subdivision led to poorer performance.
As part of this work we release the following datasets[1]

1. We release 3,500 of the total 5,500 training questions along with the 500 test questions originally released by Li and Roth (2002) manually updated with our classification.
2. We release the manually verified question classes for all 1500 questions in the AS task.
3. From the questions contained in the AS task, we release the list of entities identified for each answer candidate for the complete test set and for a highlighted training set of 649 questions.
4. We also make available an Application Programming Interface (API) to the modified Question Classification system we describe in this work[2].

5.7.2 Additional advantages

Unlike other systems, a significant advantage of the system presented in this work is the fact that, not only does the system succeed in Answer Selection but in most cases *can also extract the specific factoid answer*, when one is present. When highlighting is possible and a maximal entity exists (as is the case in 60 of the 68 questions in the clean test set), such a maximal entity is nearly always the answer.

5.8 Error Analysis

One of the biggest source of errors tends to be incorrect entity tagging. This is especially so in the case of dates where we tag the date and the year independently thus allowing for the tagging of entities in candidate answer that only mention the year. Unfortunately, this often leads to incorrectly identified maximal entities when both the date and the year are present.

[1]Download from: www.harishmadabushi.com/research/answer-selection/
[2]API available at: www.harishmadabushi.com/research/questionclassification/

In hindsight, we believe a better approach to dates would have been to combine sequential entities before entity matching, so as to capture exact dates, such as “13 October 1997” as a single entity rather than two different entities, one consisting of the day and month and the other of the year.

A similar source of errors occurs when processing the names of individuals, where common ways of shortening names cause errors in both the entity identification and the entity matching steps. For example, the ex-CEO of GE, Jack Welch, is referred to as GE-Welch and John Welch in some answer candidates.

When a candidate answer has too many entities of the required type, it is often misclassified. For example, the candidate answer “on its billions of kilometres flight toward Saturn, Cassini is scheduled to loop its path around the Venus, the Earth and Jupiter to get the gravity boost needed for closing in onto its destination” for the question “What is Cassini’s destination?”, contains a large number of planetary objects diluting the signal generated by the entity. We believe the solution to this lies in increasing the granularity of the QC taxonomy.

There are some completely unexpected sources of errors as in the case of the question “Who established the Nobel Prize awards?”, where all entities that contain the answer (“Alfred Nobel”) are discarded as they are contained in the question. Once again, a more sophisticated method of establishing which entities are discarded could reduce errors of this kind.

Finally, there are some questions and answer candidates that are incorrectly tagged in the Answer Selection test set. For example, the question “What years did Sacajawea accompany Lewis and Clark on their expedition?” has, amongst its candidate sentences, the following two sentences: “the coin honors the young woman and teen-age mother who accompanied explorers Meriwether Lewis and William Clark to the pacific ocean in 1805”, and “in 1804, Toussaint was hired by Lewis and Clark, not for his own skills but for those of Sacagawea.” While the first candidate answer is marked as one containing the answer the second is not marked as such. While this is the only error of this kind, we found several other sentence candidates similarly marked as containing the answer when they, in fact, do not. Despite this, *we make no modifications to the test set*.

5.9 Summary

This chapter presented a method of Answer Selection that makes use of a slightly modified version of the Question Classification method presented in the previous chapter (Chapter 4). We additionally show how Types can be used for entity identification and classification.

This method significantly outperforms the existing state of the art and shows that methods that make use of both linguistic analysis and deep learning can be potent. The next chapter provides an overview of the contributions of this work and the possible ways in which they can be extended in the future. We present one possible method for using this research for the creation of an end-to-end Question Answering system in Section 6.3.5.

Data version		Sr. No.	Question Class	Highlight Method	Train #	Test #	Baseline MRR	Baseline MAP	This Work MRR	This Work MAP
RAW	This Work	1	unhighlighted	N.A.	515	13	(0.8065)	(0.8462)	N.A.	N.A.
		2	highlighted	bracketing	649	82	(0.7583)	(0.8179)	(0.8124)	(0.8304)
		3	highlighted	replacement	649	82	(0.7583)	(0.8179)	(0.8174)	(0.8293)
		4	Combining results from 1 & 2						0.8116	0.8326
		5	Combining results from 1 & 3						0.8262	0.8422
		6	combined	bracketing	1164	95	0.7783	0.8386	0.806	0.8316
		7	combined	replacement	1164	95	0.7783	0.8386	**0.8362**	**0.8625**
	Reported Baseline Performance (Rao et al., 2016)					95	0.780	0.834		
	Implementation Best					95	0.7904	0.8223		
	Prior State of Art (Rao et al., 2016)					95	*0.78*	*0.834*		
CLEAN	This Work	8	unhighlighted	N.A.	515	6	(0.6621)	(0.7222)	N.A.	N.A.
		9	highlighted	bracketing	649	62	(0.7449)	(0.7926)	(0.8354)	(0.8679)
		10	highlighted	replacement	649	62	(0.7449)	(0.7926)	(0.6991)	(0.8211)
		11	Combining results from 8 & 9						0.8201	0.855
		12	Combining results from 8 & 10						0.6958	0.8123
		13	combined	bracketing	1164	68	0.7713	0.8368	0.8324	0.862
		14	combined	replacement	1164	68	0.7713	0.8368	**0.8647**	**0.9039**
	Reported Baseline Performance (Rao et al., 2016)					68	0.801	0.877		
	Prior State of Art (Shen et al., 2017)					68	0.822	0.899		

Table 5.4 Results for each of the different highlighting methods on the Raw and Clean data versions. "Reported Perfomance" indicates the performance reported by Rao et al. (2016) when using the sentence-level model (Section 5.6) which is our baseline and the model that we adapt. "Implementation Best" represents the performance of the same model on the implementation that we use, and "Prior State of Art" represents the prior State of Art reported in literature for this dataset. Results in parentheses represent results on a subset of the test data.

6 Conclusions and Future Work

The incredible success of deep learning methods has led to their increased use across tasks including in natural language processing. Deep learning methods, however, suffer from a lack of transparency. So far, making deep learning systems transparent has been seen as a requirement for understandability but not accuracy.

Work described in this thesis integrates linguistic information into deep learning methods making deep learning more transparent while also achieving higher accuracy. This work shows that the decades of work that exists in linguistics, which was ignored as a consequence of the rise of deep learning, can be integrated into deep learning systems to improve results on tasks while also making methods easier to "understand" and debug.

6.1 Contributions

This section highlights the contributions made by this thesis towards improving Question Answering systems.

6.1.1 A New Method for Learning

This work provided a novel method for creating rules and generalising them through a method we call ArCH learning that makes use of what we call Types. We note that this is a more precise, if labour intensive, version of semantic similarities provided by word embeddings. However, the added precision provided by such flexibility in defining rules results in a system for Question Classification which is not only transparent, but also superior to all existing systems. There are other good reasons for using Types in complex tasks as described in Section 3.5.

An exciting avenue of future exploration is a method of automating the process of defining Types using, amongst other things, word embeddings and deep learning to bring together the advantages of Types with those of deep learning. We describe this in more detail in section 6.3.1.

6.1.2 Semantic Text Similarity

Types provide a way of expressing a Concept making them useful in differentiating between subtle ideas that are instrumental in finding the degree of similarity between two sentences. Section 2.2 provided details on our contribution to the task of Semantic Text Similarity wherein we were able to achieve significant improvements through the use of Types, Surprise and Phrase Linking. Section 6.3.2 details ways in which this research can be built upon.

6.1.3 Question Classification

Chapter 4 presented a purely rule-based system for QC which exploits decades of research into the structure of language and Concepts. A purely semantic system, without a learning component, leads to a system wherein, although the definition of various rules is cumbersome, each additional rule will always increase the number of *kinds* of questions that can be classified while maintaining a perfect recall. Systems that depend on machine learning must sacrifice recall to ensure that they avoid over-fitting. The use of Types also allows us to define classes for a vastly larger number of words than those that are manually defined (e.g. 246 manually selected words in various Types get translated into the definition of 39,055 rules).

We also note that these are a common and vital kind of questions, which are similar to those handled by most modern smartphone interactive systems such as Google Now (Ristovski, 2016).

As part of this work, a simple to use Application Programming Interface (API) [1] is made available so other QA systems may benefit from this work. Several researchers from around the world have already made use of this API to classify over 70,000 questions. We detail possible future directions of research related to Question Classification in Section 6.3.3.

6.1.4 Answer Selection

Chapter 5 presented a method of Answer Selection that first required us to redefine the QC taxonomy provided by Li and Roth (2002), modify the Question Classification system previously developed (Chapter 4, Tayyar Madabushi and Lee (2016)) to ensure that it matches this modified taxonomy, create an entity identification method to extract entities belonging to those classes in our taxonomy, and finally, to use different methods of highlighting entities, so this information can be passed on to the Answer Selection model developed by Rao et al. (2016).

Our experiments show that this method of Answer Selection outperforms the previous state of the art in all variations except one. In the best configuration, our MRR and MAP scores outperform the current state of the art by between 3 and 5 points on both the Raw and Clean versions of the TrecQA Answer Selection test set. The relatively small size of this dataset combined with the rather high accuracy achieved by the current state of the art systems highlights the need for research in the field of AS through QC to move to other datasets. One challenge in doing so could be the lack of QA datasets with QC annotations. We hope that our release of the QC API as part of this work will help in this regard. We describe possible ways of extending this work in Section 6.3.4.

[1] API available at http://www.harishmadabushi.com/research/questionclassification/

6.2 Research Questions: Answers and Discussion

This section explores the research questions introduced in Section 1.6 and how the work presented herein answers these questions. The first research question related to the possibility of methods that do not use deep learning outperforming deep learning methods:

- **Research Question 1:** Has the success of deep learning methods made it impossible to improve upon the state of art of various NLP tasks without the use of deep learning?

This work has shown conclusively that this is not the case. Work presented in Chapter 4 significantly improved upon the accuracy of question classification, well beyond anything machine learning models were able to achieve. This is significant as it shows that deep learning is not the only way forward and that an exploration of linguistic structure remains beneficial.

The second research question related to an alternate method of generalisation as this aspect of deep learning was not being exploited to the extent it was in other fields:

- **Research Question 2:** Given that the success of deep learning methods lies in their ability to abstract learning, and that learning abstraction is not where the gains in NLP stem from, are there other forms of generalisation that might be more suitable to NLP?

Chapter 3 presented Types, a novel method of generalisation that is able to generalise natural language information. We have additionally shown in Section 3.3 that there are parallels between Types and Distributed Word Embeddings. While there is more work required to show that Types are a better form of generalisation than deep learning in the context of NLP, this work has shown that Types do have the potential to be more effective in some tasks, such as Question Classification (Chapter 4).

The final research question pertained to the integration of information extracted through linguistic analysis into deep learning models:

- **Research Question 3:** How can features discovered through the analysis of language and language structure be fed into deep learning models without fundamental changes to those models?

This work has shown that “Highlighting”, detailed in Section 5.6.1, is an effective method of passing additional information to deep learning models. In addition to being able to pass on additional information, our work has shown how such additional information can be incredibly effective in improving the accuracy of the original deep learning model.

6.3 Future Work

This section details possible future directions for the research presented in this thesis. We start by exploring how Types can be extended and generalised, and learning Types can be automated, before describing an end-to-end Question Answering system that can be developed using components described in this work.

6.3.1 Extending Types

As described in section 3.4, Types are specific to the task at hand, and this flexibility has the advantage of allowing individual tasks to handle information at different granularity. However, we hope to create Types modified in such a way as to allow dynamic selection of granularity. This modification will remove the need to redefine Types for individual tasks and allow for the creation of *Universal Types*.

Simultaneously, we would like to automate ArCH learning (Section 3.4.1), the process used to create Types. This will enable the use of Types in niche domains that use specialised vocabulary in addition to aiding in the creation of Universal Types. This automation could be achieved through the use of word embeddings and deep learning methods.

6.3.2 Semantic Text Similarity

Our experiments with the use of Types in Semantic Text Similarity were limited to an exploratory study that used coarse Types. Incorporating Types with a finer granularity into Semantic Text Similarity and integrating it with deep learning methods is the next step in exploring the use of Types in Semantic Text Similarity and we hope to pursue this in future.

6.3.3 Question Classification

Although the method of Question Classification presented in this work has focused on a particular kind of question, we believe that a similar method can be applied to classifying questions belonging to different kinds of questions such as yes/no question, and we intend to extend our work to include those kinds of questions. Multiple-choice questions, however, provide a different set of challenges.

Additionally, the use of question structures, which have the potential to be extended to sentence structures, in conjunction with Concept creation, provides a novel way of creating a Word Sense Disambiguation system. This work might have the potential to lead to a system that can, at least in the most common cases of English language usage, disambiguate word senses.

6.3.4 Answer Selection

One possible avenue of future research in extending our work into Answer Selection is in implementing and testing this method on different datasets including WikiQA, while also increasing the number of classes in the question taxonomy and the number and kinds of entities identified by the Entity Identification component of the system. As in the case of Question Classification an obvious extension of our work is in creating an end-to-end Question Answer system, as described in the next section.

6.3.5 End to End Question Answering

Given that this work has focused on both Question Classification and Answer Selection, a natural and impactful direction for future research lies in the use of these methods in the development of an end to end Question Answering system. The most obvious way of achieving this will be in modifying a Question Answering system that relies on web redundancy (Section 2.1.4) to exploit both the Question Classification and Answer Selection methods detailed in this work.

A question could be trivially transformed into a web search query by stripping away stop words before then feeding such a search query to a search engine. Results from the search engine could be analysed for redundancy to extract candidate answers. The question would also be run through the Question Classification system, and the resultant question class and the previously extracted candidate answers could be fed to the Answer Selection system which would extract the answer. Figure 6.1 provides an overview of this process.

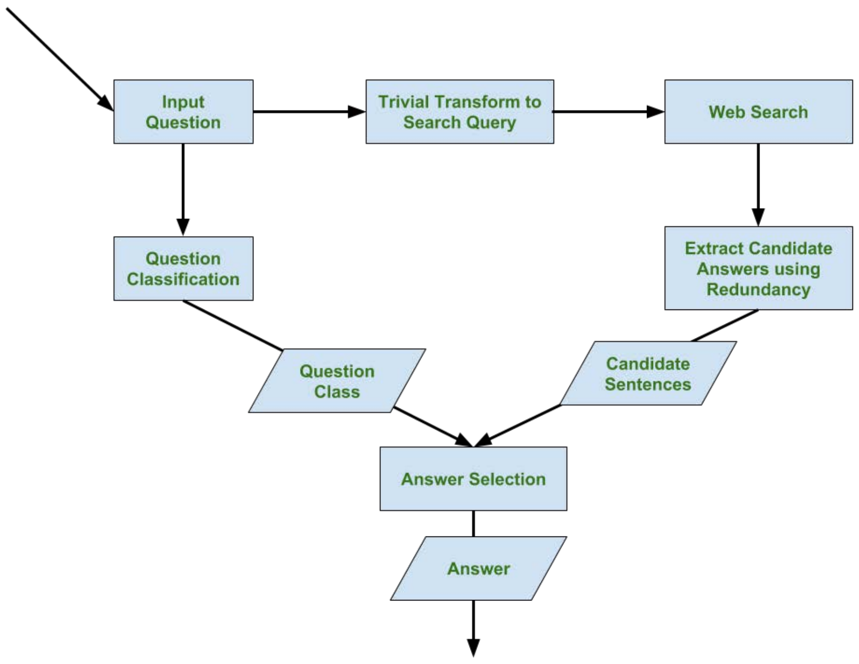


Fig. 6.1 A representation of an end to end Question Answering System based on this work

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 81–91.

Eneko Agirre, Carmen Banea, et al. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), June*.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. SEM 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *In SEM 2013: The Second Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics*. Citeseer.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 385–393.

Ramiz M Aliguliyev. 2009. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Expert Systems with Applications* 36(4):7764–7772.

Andrea Andrenucci and Eriks Sneiders. 2005. Automated question answering: review of the main approaches. *Third International Conference on Information Technology and Applications (ICITA'05)* 1:514–519 vol.1.

Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases - an introduction. *CoRR* cmp-lg/9503016.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. *CoRR* abs/1505.00468.

Sofia J. Athenikos and Hyoil Han. 2010. Biomedical question answering: A survey. *Computer Methods and Programs in Biomedicine* 99(1):1 – 24.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*. Springer-Verlag, Berlin, Heidelberg, ISWC'07/ASWC'07, pages 722–735.

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, ACL '98, pages 86–90.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 967–976.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3:1137–1155.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1415–1425.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, New York, NY, CIKM '17, pages 1987–1990.

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

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the askmsr question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 257–264.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*. Morgan Kaufmann Publishers, San Francisco, CA, NIPS'93, pages 737–744.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 423–433.

Alfonso Caramazza and Rita Sloan Berndt. 1978. Semantic and syntactic processes in aphasia: a review of the literature. *Psychological Bulletin* 85(4):898.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 740–750.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing.*. Ph.D. thesis, University. of Pennsylvania.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, New York, NY, ICML '08, pages 160–167.

Carol Conrad. 1972. Cognitive economy in semantic memory. 92:149–154.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, SIGIR '05, pages 400–407.

Robert Dale. 2016. The return of the chatbots. *Natural Language Engineering* 22(5):811–817.

Robert D'Assisi. 2016. Why search is broken, and how we intend to fix it. (Blog Post). [Retrieved Aug 2018].

Maurice de Kunder. 2008. Geschatte grootte van het geïndexeerde world wide web. *Tilburg University* Master's Thesis.

Maurice de Kunder. 2015. The size of the World Wide Web. http://www.worldwidewebsize.com/. [Retrieved: 1st April 2015].

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*. Association for Computational Linguistics, Stroudsburg, PA, USA, CrossParser '08, pages 1–8.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR* abs/1602.03609.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.* 19(1):61–74.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Oren Etzioni. 2011. Search needs a shake-up. *Nature* 476(7358):25–26.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, EMNLP '11, pages 1535–1545.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1608–1618.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open Question Answering over Curated and Extracted Knowledge Bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, KDD '14, pages 1156–1165.

Guangyu Feng, Kun Xiong, Yang Tang, Anqi Cui, Jing Bai, Hang Li, Qiang Yang, and Ming Li. 2015. Question Classification by Approximating Semantics. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, New York, NY, WWW '15 Companion, pages 407–417.

Jure Ferlež and Matjaz Gams. 2004. Shortest-Path Semantic Distance Measure in WordNet v2.0. *Informatica* 28:385–390.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. ???? Incorporating non-local information into information extraction systems by Gibbs sampling, booktitle = Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, series = ACL '05, year = 2005, location = Ann Arbor, Michigan, pages = 363–370, numpages = 8, url = https://doi.org/10.3115/1219840.1219885, doi = 10.3115/1219840.1219885, acmid = 1219885, publisher = Association for Computational Linguistics, address = Stroudsburg, PA,.

John Rupert Firth. 1957. *A synopsis of linguistic theory 1930-55.*, volume 1952-59. The Philological Society, Oxford.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *HLT-NAACL*. pages 758–764.

Freebase Google+. 2015. Freebase. https://plus.google.com/109936836907132434202/posts/bu3z2wVqcQc. [Retrieved: 17th August 2015].

David Gunning. 2018. Explainable artificial intelligence (XAI). DARPA Blog Post.

James Hampton. 2006. *The Psychology of Learning and Motivation: Advances in Research and Theory*, volume 46.

Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC_EBIQUITY-CORE: Semantic Textual Similarity systems. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. Association for Computational Linguistics, Atlanta, Georgia, pages 44–52.

Christian Hänig, Robert Remus, and Xose de la Puente. 2015. Exb themis: Extensive feature extraction from word alignments for semantic textual similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 264–268.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1576–1586.

Hua He and Jimmy Lin. 2016. 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*. Association for Computational Linguistics, San Diego, CA, pages 937–948.

Shizhu He, Kang Liu, Yuanzhe Zhang, Liheng Xu, and Jun Zhao. 2014. Question answering over linked data using first-order logic. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, pages 1092–1103.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, COLING '92, pages 539–545.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, HLT '10, pages 1011–1019.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR* abs/1506.03340.

Ulf Hermjakob. 2001. Parsing and question classification for question answering. In *Proceedings of the Workshop on Open-domain Question Answering - Volume 12*. Association for Computational Linguistics, Stroudsburg, PA, ODQA '01, pages 1–6.

Lynette Hirschman and Rob Gaizauskas. 2001. Natural language question answering: The view from here. *Nat. Lang. Eng.* 7(4):275–300.

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

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2000. Question answering in webclopedia. In *The Ninth Text REtrieval Conference (TREC 9)*. volume 52, pages 53–56.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*. Association for Computational Linguistics, Stroudsburg, PA, HLT '01, pages 1–7.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, EMNLP '08, pages 927–936.

Susan Hunston and Gill Francis. 2000. *Pattern Grammar: A Corpus-driven Approach to the Lexical Grammar of English*. Pattern Grammar: A Corpus-driven Approach to the Lexical Grammar of English. John Benjamins Publishing Company.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing*.

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an explanation? characterizing knowledge and inference requirements for elementary science exams. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 2956–2965.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR* cmp-lg/9709008.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning* 37(2):183–233.

Abhay Kashyap, Lushan Han, Roberto Yus, Jennifer Sleeman, Taneeya Satyapanich, Sunil Gandhi, and Tim Finin. 2014. Meerkat Mafia: Multilingual and cross-level semantic textual similarity systems. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, pages 416–423.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, pages 455–462.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR* abs/1408.5882.

Ross Kindermann and J. Laurie Snell. 1980. *Markov Random Fields and Their Applications*. AMS books online. American Mathematical Society.

Tracy Holloway King, Richard S. Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The parc 700 dependency bank. In *LINC@EACL*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *CoRR* abs/1506.06726.

Oleksandr Kolomiyets and Marie-Francine Moens. 2011. A survey on question answering technology from an information retrieval perspective. *Information Sciences* 181(24):5412 – 5434.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Curran Associates Inc., USA, NIPS'12, pages 1097–1105.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1545–1556.

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling question answering to the web. *ACM Trans. Inf. Syst.* 19(3):242–262.

Denis Kyashif. 2018. Machine comprehension using commonsense knowledge - a dynamic memory network approach. https://github.com/deniskyashif/sweet-reason.

C. Leacock and M. Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. In Christiane Fellfaum, editor, *MIT Press*. Cambridge, Massachusetts, pages 265–283.

Y. LeCun, Fu Jie Huang, and L. Bottou. 2004. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*. volume 2, pages II–97–104 Vol.2.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1990. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, Morgan-Kaufmann, pages 396–404.

Fritz Lehmann. 1992. Semantic networks. *Computers & Mathematics with Applications* 23(2):1 – 50.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*. ACM, New York, NY, SIGDOC '86, pages 24–26.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL* 3:211–225.

Shasha Li, Chin-Yew Lin, Young-In Song, and Zhoujun Li. 2010. Comparable entity mining from comparative questions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, ACL '10, pages 650–658.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, COLING '02, pages 1–7.

Xin Li and Dan Roth. 2006. Learning question classifiers: The role of semantic information. *Nat. Lang. Eng.* 12(3):229–249.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, ICML '98, pages 296–304.

Dekang Lin and Patrick Pantel. 2001. Dirt @sbt@discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, KDD '01, pages 323–328.

Babak Loni. 2011. A survey of state-of-the-art methods on question classification. journal article uuid:8e57caa8-04fc-4fe2-b668-20767ab3db92, Delft University of Technology.

Babak Loni, Gijs van Tulder, Pascal Wiggers, David M. J. Tax, and Marco Loog. 2011. *Question Classification by Weighted Combination of Lexical, Syntactic and Semantic Features*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 243–250.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. *CoRR* abs/1606.00061.

Allan M. Collins and M Ross Quillian. 1972. Experiments on semantic memory and language comprehension pages vii, 263 – vii, 263.

Christopher D. Manning. 2015. Last Words: Computational Linguistics and Deep Learning. *Computational Linguistics* 41(4):701–707.

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

Andreas Merkel and Dietrich Klakow. 2007. Improved methods for language model based question classification. In *INTERSPEECH*. pages 322–325.

Donald Metzler and W. Bruce Croft. 2005. Analysis of statistical question classification for fact-based questions. *Information Retrieval* 8(3):481–504.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*. ACM, New York, NY, CIKM '07, pages 233–242.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. *CoRR* abs/1310.4546.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM* 38(11):39–41.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: An on-line lexical database*. *International Journal of Lexicography* 3(4):235–244.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes* 6(1):1–28.

Amit Mishra and Sanjay Kumar Jain. 2016. A survey on question answering systems with classification. *J. King Saud Univ. Comput. Inf. Sci.* 28(3):345–361.

Diego Mollá and José Luis Vicedo. 2007. Question answering in restricted domains: An overview. *Comput. Linguist.* 33(1):41–61.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, pages 27–48.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, pages 525–545.

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

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* 12:2825–2830.

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

Fernando Pereira, Peter Norvig, and Alon Halevy. 2009. The unreasonable effectiveness of data. *IEEE Intelligent Systems* 24:8–12.

Florent Perek. 2016. Recent change in the productivity and schematicity of the way-construction: a distributional semantic analysis. *Corpus Linguistics and Linguistic Theory* 14:65–97.

Marco Pota, Massimo Esposito, and Giuseppe De Pietro. 2016. *A Forward-Selection Algorithm for SVM-Based Question Classification in Cognitive Systems*, Springer International, Cham, pages 587–598.

Marco Pota, Angela Fuggi, Massimo Esposito, and Giuseppe De Pietro. 2015. Extracting compact sets of features for question classification in cognitive systems: A comparative study. In *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. pages 551–556.

John Prager. 2007. *Open-Domain Question Answering: Foundations and Trends(R) in Information Retrieval*. Now Publishers, Hanover, MA.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, New York, NY, CIKM '16, pages 1913–1916.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.

Kristijan Ristovski. 2016. A complete list of Google now commands. *http://ok-google.io/*. Retrieved: June 2016.

Dmitri Roussinov, Elena Filatova, Michael Chau, and Jose Robles-Flores. 2005. Building on redundancy: Factoid question answering, robust retrieval and the "other". In *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings*.

Patrick Schone, Gary M. Ciany, R. Cutts, Paul Mcnamee, James Mayfield, and Thomas Smith. 2005. QACTIS-based question answering at TREC 2005. In *The 14th Text REtrieval Conference (TREC 2005) Proceedings*.

SCImago. 2016. SJR - SCImago Journal & Country Rank. http://www.scimagojr.com. [Retrieved: 29th June 2016].

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 458–467.

Aliaksei Severyn and Alessandro Moschitti. 2015. 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*. ACM, New York, NY, SIGIR '15, pages 373–382.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. page 12–21.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1179–1189.

João Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review* 35(2):137–154.

R. F. Simmons. 1965. Answering English questions by computer: A survey. *Commun. ACM* 8(1):53–70.

Robert F. Simmons. 1970. Natural language question-answering systems: 1969. *Commun. ACM* 13(1):15–30.

J. F. Sowa. 1976. Conceptual graphs for a data base interface. *IBM Journal of Research and Development* 20(4):336–357.

Rohini Srihari and Wei Li. 2000. A question answering system supported by information extraction. In *ANLC '00 Proceedings of the Sixth Conference on Applied Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, pages 166–172.

Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 2010. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 841–842.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014a. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics* 2:219–230.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014b. DLS@CU: Sentence similarity from word alignment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, pages 241–246.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, CO, pages 148–153.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR* abs/1511.04108.

Harish Tayyar Madabushi, Mark Buhagiar, and Mark Lee. 2016. UoB-UK at SemEval-2016 Task 1: A Flexible and Extendable System for Semantic Text Similarity using Types, Surprise and Phrase Linking. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, CA, pages 680–685.

Harish Tayyar Madabushi and Mark Lee. 2016. High accuracy rule-based question classification using question syntax and semantics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1220–1230.

Harish Tayyar Madabushi, Mark Lee, and John Barnden. 2018. Integrating question classification and deep learning for improved answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 3283–3294.

Chen-Tse Tsai, Wen-tau Yih, Chris J.C. Burges, and Scott Wen-tau Yih. 2015. Web-based question answering: Revisiting askmsr. Technical report, Microsoft Research, Redmond, Washington.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In *Proceedings of the 21st International Conference on World Wide Web*. ACM, New York, WWW '12, pages 639–648.

Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, pages 2643–2651.

Nguyen Van-Tu and Le Anh-Cuong. 2016. Improving question classification by feature extraction and selection. *Indian Journal of Science and Technology* 9(17):1–8.

Ellen M. Voorhees. 2001a. Question answering in TREC. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*. ACM, New York, CIKM '01, pages 535–537.

Ellen M. Voorhees. 2001b. The TREC question answering track. *Natural Language Engineering* 7(4):361–378.

Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. 2012. Evaluation of a layered approach to question answering over linked data. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, JosianeXavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2012*, Springer Berlin Heidelberg, volume 7650 of *Lecture Notes in Computer Science*, pages 362–374.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *EMNLP-CoNLL*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *CoRR* abs/1702.03814.

Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. *CoRR* abs/1507.02628.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *CoRR* abs/1602.07019.

Casimir Wierzynski. 2018. The challenges and opportunities of explainable AI. Intel Blog Post.

Min Wu, Michelle Duan, Samira Shaikh, Sharon G. Small, and Tomek Strzalkowski. 2005. ILQUA–An IE-Driven Question Answering System. In *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings*.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, ACL '94, pages 133–138.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, pages 645–650.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, ACL '12, pages 712–720.

Xuchen Yao. 2014. *Feature-Driven Question Answering With Natural Language Alignment*. Ph.D. thesis, Johns Hopkins University.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. ACL – Association for Computational Linguistics.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of ACL*. Association for Computational Linguistics.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Workshop on Human-Computer Question Answering*. Association for Computational Linguistics, San Diego, California, pages 36–42.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2017. Recent trends in deep learning based natural language processing. *CoRR* abs/1708.02709.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR* abs/1412.1632.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR* abs/1510.03820.

A STS Type Definitions

A.1 Non-Noun Type Definitions

The Table below details Type definitions for all POS other than Common nouns.

Type	POS and Tokens
Pronouns	PRP, PRP$
Proper Nouns	NNP, NNPS
Foreign words	FW
All Noun	NN , NNS
Numbers	CD
Modal	MD
Predeterminer	PDT
Wha Words	WDT, WP, WP$, WRB
verb_pos	VB , VBD, VBN, VBP
verb_vbg	VBG
verb_vbz	VBZ
Adjectives	JJ , JJR, JJS
Adverbs	RB , RBR, RBS
Sentence Separators	. , !
Question Mark	?
Ignored	CC, DT, EX, IN, LS, PDT, POS, RP, SYM, TO, UH

A.2 Noun Type Definitions

The following table shows all Termination Concepts used, and associated (Common) Noun Types.

Termination Concept	Type
person.n.01	person_r1
group.n.01	person_r2
social_group.n.01	person_r2
body_part.n.01	person_r3
animal.n.01	animal_r1
plant.n.02	plant_r1
body_of_water.n.01	thing_r2
natural_object.n.01	thing_r3
living_thing.n.01	animal_r2
organism.n.01	animal_r2
food.n.01	thing_r1
food.n.02	thing_r1
cognition.n.01	idea_r1
motivation.n.01	idea_r1
agent.n.03	animal_r3
causal_agent.n.01	animal_r3
communication.n.02	thing_r1
writ-ten_communication.n.01	thing_r1
act.n.02	thing_r1
social_event.n.01	thing_r1
location.n.01	place_r1
land.n.04	place_r2
geological_formation.n.01	place_r3
natural_process.n.01	thing_r4
organic_process.n.01	thing_r4
natural_phenomenon.n.01	thing_r4
part.n.03	thing_r3

Termination Concept	Type
relation.n.01	ab-stract_r1
flow.n.01	move_r1
attribute.n.02	thing_r3
surface.n.02	thing_r3
happening.n.01	thing_r5
causal_agent.n.01	thing_r2
measure.n.02	thing_r5
change.n.06	thing_r5
phenomenon.n.01	thing_r5
ability.n.02	thing_r1
quality.n.01	thing_r1
degree.n.01	ab-stract_r1
artifact.n.01	thing_r1
substance.n.04	thing_r3
part.n.02	thing_r1
catch.n.04	thing_r1
processing.n.01	thing_r1
substance.n.07	thing_r3
event.n.01	thing_r2
matter.n.03	thing_r6
entity.n.01	thing_r6
process.n.06	thing_r5
thing.n.08	thing_r6
physical_entity.n.01	thing_r6
object.n.01	thing_r6
thing.n.12	thing_r6
abstraction.n.06	ab-stract_r2
solid.n.01	thing_r6

A.3 Weights

We list below the various combinations of weights we assign to each Type to generate Similarity Scores for each of our “Methods”. Although we try several other combinations, we list here only those that we included in our submission.

Pronouns	Proper Nouns	All Nouns	Numbers	Foreign Words	Modal	Predeterminer	Wha words	All Verbs	Adjectives	Adverbs	Sentence Separators	verb_vbg	verb_vbz	Question mark	person_r1	person_r2	person_r3	animal_r1	animal_r2	animal_r3	idea_r1	move_r1	abstract_r1	abstract_r2	place_r1	place_r2	place_r3	plant_r1	thing_r1	thing_r2	thing_r3	thing_r4	thing_r5	thing_r6
0	1	1	1	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	3	2	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	2	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	5	2	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	5	3	1	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	500	500	0	0	0	0	0	50	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	20	20	0	0	0	0	0	20	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	100	20	0	0	0	0	0	20	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	1000	20	0	0	0	0	0	20	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	20	30	0	0	0	0	0	20	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	10	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	50	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	10	5	5	0	0	0	0	3	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	0	1	0	0	0	0	1	1	1	0	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	1	0	1	0	0	0	0	1	1	1	0	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	1	0	1	0	0	0	0	1	1	1	0	1	1	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Index