Simple Convolutional Neural Networks with Linguistically-Annotated Input for Answer Selection in Question Answering

by

Royal Sequiera

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2018

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

With the advent of deep learning methods, researchers have been increasingly preferring deep learning methods over decades-old feature-engineering-inspired work in Natural Language Processing (NLP). The research community has been moving away from otherwise dominant feature engineering approaches; rather, is gravitating towards more complicated neural architectures. Highly competitive tools like part-of-speech taggers that exhibit human-like accuracy are traded off for complex networks, with the hope that the neural network will learn the features needed. In fact, there have been efforts to do NLP "from scratch" with neural networks that altogether eschew featuring engineering based tools (Collobert et al, 2011). In our research, we modify the input that is fed to neural networks by annotating the input with linguistic information: POS tags, Named Entity Recognition output, linguistic relations, etc. With just the addition of these linguistic features on a simple Siamese convolutional neural network, we are able to achieve state-of-the-art results. We argue that this strikes a better balance between feature vs. network engineering.

Acknowledgements

I would like to thank Prof. Jimmy Lin, my supervisor and my mentor for guiding me throughout my Master's degree. Jimmy has not only helped me grow academically, but also has had deep impact on me personally.

I am grateful Prof. Yaoliang Yu and Prof. Ming Li for agreeing to be on my thesis committee despite their busy schedule.

I thank my friends and colleagues Abhinav, Besat, Chathura, Haotian Zhang, Hemant Saxena, Luchen Tan, Jae Kim, Kshitij, Matt Crane, Melica Mirsafian, Michael Azmy, Michael Tu, Mustafa, Nimesh, Priya, Salman, Sharat, Siddhartha, Victor Yang, and Vijay Menon for always being there for me and for making my time at University of Waterloo an enjoyable experience. I shall definitely miss our spirited discussions, shared luncheons, and water-fetching sessions.

,

Dedication

This is dedicated to Jayashree Miss, my high-school teacher, without whom I wouldn't have been the same person that I am today and to my parents for their steadfast love and support.

Table of Contents

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Contributions . . . 3

1.2 Thesis Organization . . . 4

2 Background and Related Work 5

2.1 Related Work . . . 7

2.1.1 Non-neural methods . . . 7

2.1.2 Neural methods . . . 9

2.2 Datasets . . . 15

2.2.1 TREC QA . . . 15

2.2.2 WikiQA . . . 17

2.3 Metrics . . . 18

2.3.1 Mean Average Precision (MAP) . . . 18

2.3.2 Mean Reciprocal Ranking . . . 20

3 Baselines 21

3.1 Non-nerual Baselines . . . 21

3.1.1 Weighted Word Overlap . . . 21
3.1.2 Word Mover's Distance . . . 22
3.1.3 TextFlow . . . 23
3.2 Neural Baseline: SM Model . . . 24

4 Proposed Model **28**
4.1 Part-of-speech Tagging . . . 29
4.2 Named Entity Recognition . . . 31
4.3 Dependency Parsing . . . 32
4.4 Experimental Setup . . . 35
4.4.1 Word Embedding . . . 36
4.4.2 Part-of-speech . . . 36
4.4.3 NER . . . 36
4.4.4 Dependency + Headword . . . 36

5 Experimental Results **38**

6 Conclusion **43**

References **45**

List of Tables

2.1 Source document collections for TrecQA. 16
2.2 Statistics for various splits of TrecQA. 16
2.3 Statistics for various splits of WikiQA. 18

3.1 Baseline resultson TREC QA dataset 26
3.2 Baseline resultson WikiQA dataset 26

5.1 Results on the TREC QA dataset 39
5.2 Results on the WikiQA dataset 40

List of Figures

1.1 A sample question-answers pair for answer selection chosen from the TrecQA dataset [53]. The relevant and non-relevant answers are indicated by (+) and (−) respectively. . . . 1

2.1 A typical question answering pipeline architecture, adapted from Tellex et al. [46] . . . 6

2.2 An Answer Selection module would rerank the candidate answers such that the most relevant answers will appear at the top of the ranked list. The relevant and non-relevant answers are indicated by (+) and (−) respectively. 6

2.3 A tree edit sequence transforming a premise to an entailed hypothesis as adapted from Heilman and Smith [16] . . . 8

2.4 The architecture of the one dimensional convolutional neural network as adopted from Yu et al. [64] . . . 10

2.5 The architecture of biLSTM CNN as adopted from Tan et al. [42] . . . 11

2.6 The architecture of the pair-wise interaction model as adopted from He et al. [15] . . . 12

2.7 Architecture of the Holographic Dual LSTM model (HDLSTM) as adopted from Tay et al. [43] . . . 14

2.8 Architecture of the bilateral multi-perspective matching model (BiMPM) as adopted from Wang et al. [55] . . . 15

2.9 An example of an answer that does not share non-stop question words . . . 18

3.1 Illustration of IDF weighted word overlap baseline. . . . 22

3.2 Working of Word Mover's Distance . . . 23

3.3 Working of TextFlow as adopted from Mrabet et al. [27] 24
3.4 The architecture of SM model as adopted from Sequiera et al. [35] 25

4.1 The input word embedding to SM CNN. Note that this figure is the rotated form of the input word embedding matrix in Figure 3.4 29
4.2 PTB tagset as adopted from Jurafsky et al. [17] 30
4.3 Universal POS tagset as proposed by Petrov et al. [29] 31
4.4 A list of named entities as adopted from Jurafsky et al. [17] 32
4.5 Examples of dependency relaltionships as adopted from Jurafsky et al. [17] 33
4.6 Dependency parsing of a simple sentence 33
4.7 Longer dependencies while dependency parsing 34
4.8 The constituents of Lai-CNN. Note that this figure is the rotated form of the input word embedding matrix in Figure 3.4 35

Chapter 1

Introduction

Factoid question answering refers to question answering where the answers are facts. Questions such as, *Who is the president of United States?*, *Who is the author of the Harry Potter series?* etc., are examples of factoid question answering. Factoid question answering usually has shorter and one or fewer answers. One of the major components of question answering is answer selection. Answer selection implies choosing the right set of answers given a natural language question. That is, given a question q and a candidate set of sentences $\{c_1, c_2, \ldots c_n\}$, the goal of answer selection is to recognize the sentences that have the correct answers in them. Typically, answer selection forms an important component in an overall question answering pipeline architecture [46]. A sample question-answer set is as shown in Figure 1.

For several years, research in Natural Language Processing (NLP) has been inspired by

Question: What is the name of Durst's group?
Answers:
(+) Limp Bizkit lead singer Fred Durst did a lot before he hit the big time.
(+) "I hope he's OK", said Limp Bizkit singer Fred Durst.
(−) "Our world today is really emotional", says Durst.
(−) The group's DJ Lethal is now a member of Limp Bizkit.
(−) The Ozzfest is a proving ground but not, Durst hopes, a competition.

Figure 1.1: A sample question-answers pair for answer selection chosen from the TrecQA dataset [53]. The relevant and non-relevant answers are indicated by (+) and (−) respectively.

Machine Learning models that frequently involve the use of feature engineering techniques. The efforts of this research area has culminated in several systems such as Part of Speech (POS) taggers, Dependency parsers, and Named Entity Recognizers (NER) that exhibit human level accuracies. Not being an exception, until recently, the research in answer selection for question answering was also powered by machine learning models with feature engineering [53, 41, 60, 61]. While these models require hand-crafted features for better answer selection, one of the attractive characteristics of these models is that they are simple and are easily interpretable.

With the advent of neural networks (NN), however, nearly all NLP problems are addressed through them. Since NNs do not require any feature engineering, there also have been attempts to promote "Natural Language Processing from Scratch" in the NLP community, with emphasis on avoiding feature engineering altogether [6]. With the seemingly panacea use of NNs, decades worth of NLP research has been increasingly abandoned. Nevertheless, complex neural models have been proposed to achieve higher effective measures over the standard datasets, making it harder for the community to interpret these models.

Not being an exception to this trend, recent work in answer selection has largely moved away from feature engineering approaches that were dominant in the past decade. Nearly all work today takes advantage of NNs, with researchers proposing models of great sophistication and complexity. A common theme among all these NN models is that they almost always take word embeddings representing the questions and answers as input. Unlike the traditional machine learning models, these inputs are unaccompanied by linguistic features that are known to be useful to question answering: POS tags, NER output, linguistic relations, etc. Deep learning models on the other hand, purposely shun such features, expecting that NN will learn them automatically.

The increase in complexity of these NN architectures also leads to uninterpretable models. Moreover, the unnecessary complexity of the models tend to overfit the standard datasets. It is therefore desirable to strike a balance between NN architecture and some amount of feature engineering.

This work attempts to combine a simple NN model with linguistic features from NLP models. We begin with a baseline "Siamese" convolutional neural network (CNN) for answer selection that has word embedding matrix representing a question answer pair. We augment this word input matrix with rich linguistic features. We build a simple convolutional neural network with this linguistically annotated features (Lai-CNN for short) that rivals or exceeds the state-of-the-art results on the standard QA datasets.

Through Lai-CNNs, we are trading model complexity with mature NLP technology

and using linguistic features to help the network to learn better. We argue that, although Lai-CNNs might not yield the top place in the state-of-the-art leadboard, it will certainly help in gaining better effective measure while retaining model interpretability. Overall, this hybrid approach between NNs and linguistic features strikes a better balance between feature vs. network engineering.

We incorporate the linguistic features into the network by adding linguistic features as a part of the input to the network. Since the neural network usually inputs word embedding corresponding to each word in the natural language sentence, we first convert the linguistic features corresponding to each word into a one-hot representation, and concatenate this new encoded vector to the word embedding of the corresponding word. The convolutional neural network then convolves over the input word embedding to generate feature maps. These feature maps are automatically generated saving us the trouble of engineering features. Once the features maps are created, they are fed to a fully connected neural network and the network learns to classify whether an answer is relevant to the question or not.

1.1 Contributions

The main contribution of this work is that we combine traditional feature engineering approaches with neural methods. By retaining a simple CNN model and by adding linguistic features to its input matrix, we argue that this combination of linguistic features with the neural methods strikes a good balance between complex network architectures and feature engineering. In essence, following are the main contributions of this work:

- We present several baselines: word overlap, *idf*-weighted word overlap, textflow, and word mover's distance, of which the latter two haven't been employed in QA tasks before.
- This work proposes a way to incorporate linguistic features into CNNs by augmenting the input matrices of the NNs with one-hot vector representations of linguistic features.
- Ablation experiments that demonstrate gains with respect to addition of linguistic features ordered by difficulty—the complexity of their training.

1.2 Thesis Organization

The subsequent chapters are organized in the following manner: Chapter 2 presents the background elaborating on the task, dataset and the metrics used. In chapter 3 we go over the relevant related work and chapter 4 presents various baselines and convolutional neural network based models. Chapter 5 presents the linguistically annotated input approach and chapter 6 presents the experiments and results. Chapter 7 concludes the thesis.

Chapter 2

Background and Related Work

A typical question answering system has the following general strategy for retrieving the relevant answers corresponding to a question (as illustrated by Figure 2.1):

1. Analyze the question and identify the intent of the question. The question is classified according to the type of its answer. Borrowing an example from Voorhees et al. [49], the question beginning with *who* implies the person or organization being sought, and a question beginning with *when* indicates a time designation is needed.

2. Retrieve all relevant documents by treating the given question as a query.

3. Parse the returned documents to get sentences that have entities that are of same type as the answer. If a sentence with the required entity is found, it is deemed to be the right answer to the question, otherwise the system progresses to do passage matching techniques (between the query and candidate sentences) such as word overlap.

To illustrate this better, consider the question-answers pairs as shown in Figure 2.2. The correct answer in the ranked list is at index 3, "Amazon river 6,992km long" and an ideal answer selection system will place this answer at the top of the list. Note that since there can be more than one correct answer to the question, placing the correct answers as high as possible on top of the ranked list improves the overall quality of the answer selection system.

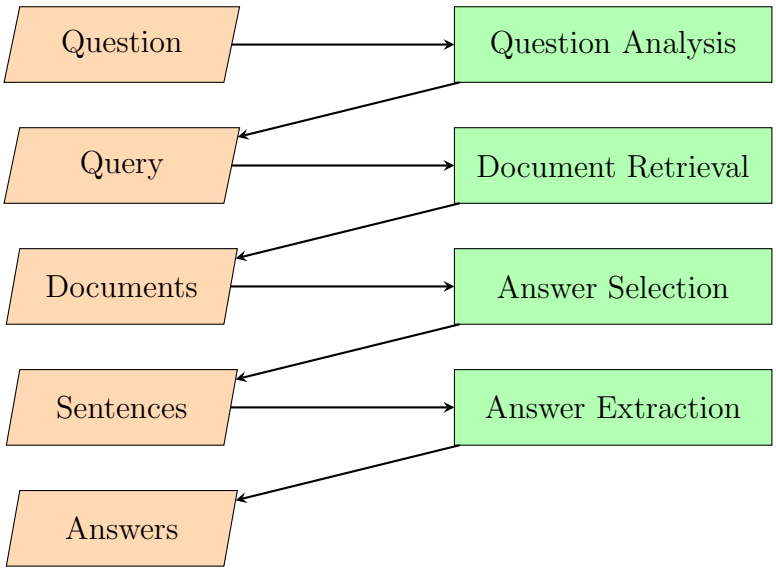

Figure 2.1: A typical question answering pipeline architecture, adapted from Tellex et al. [46]

Question: How long is the Amazon river?
Answers:
(−) Amazon rainforest is also known as Amazon Jungle
(−) Amazon river is the longest river in the world
(+) Amazon river is 6,992km long
(−) The Amazon river dolphin are also known as the pink river dolphin or boto
(−) The Amazon River is located in South America

Figure 2.2: An Answer Selection module would rerank the candidate answers such that the most relevant answers will appear at the top of the ranked list. The relevant and non-relevant answers are indicated by (+) and (−) respectively.

2.1 Related Work

In this section, we present the previous work in the field of factoid question answering; especially, the work done in the TrecQA and WikiQA datasets. We will present the evolution of the task, metrics, and various methods that were employed. First, we will discuss the non-neural approaches to factoid question answering, following which the recent advances in neural-methods will be discussed.

2.1.1 Non-neural methods

Non-neural approaches to answer selection for factoid question answering are mostly linguistically inspired approaches that try to capture the similarity of a candidate answer to a given question by formulating similarity as the distance between the question and the answer. That is, lower the distance between question and answer, likely is the answer to be relevant to the question.

Answer selection, naturally, is a text matching problem [61]. That is, the goal of the task is to measure how close an answer is to the question semantically. But, natural language is ambiguous and there are inordinate ways of expressing a same utterance. Traditional bag-of-words approaches cannot be employed since bag-of-words is invariant to the ordering of the words and therefore answers that have more overlaps with the questions, although not relevant, will be picked to be the most relevant answers. Consider, for instance, a question, *Who is the president of United States?*. A candidate answer, *The president of the United States says no to NAFTA*, although has high overlap, is not the right answer.

Overlap features for several of the NLP operations such as recognizing textual entailment, paraphrase identification, and question answering that input sentence pairs have shown to have a surprising degree of utility but the overlap features themselves are not enough for these tasks [52]. Similarly, there have been approaches, where simple word overlap methods are extended to include matching of synonyms, and varied expressions through resources like WordNet [26, 11]. A general approach in the QA community, however, has been to employ alternative representational techniques that extract useful information from the question-answers pair and to come up with a technique that matches the best candidate answer to the question.

Some of the early non-neural approaches include representing the dependency tree of both a question and a set of answers and choosing the answer that has the least edit distance to the question in terms of their dependency trees [30]. This way, both the syntactic and semantic information of the question-answer pairs is retained. The distance between the

representations were measured using an approximate tree matching algorithm that captures the number of additions, deletions, and changes to the tree. This work, however performed strict matching between the dependency trees. As an alternative approach, a stastical model based on fuzzy relation matching technique was proposedby Cui et al. [8].

Wang et al. propose "Jeopardy model", a model based on quasi-synchronous grammar formalism (developed by Smith and Eisner [40] for machine translation) [53]. The proposed model softly aligns a question sentence with a candidate answer sentence through syntactic and semantic transformations.

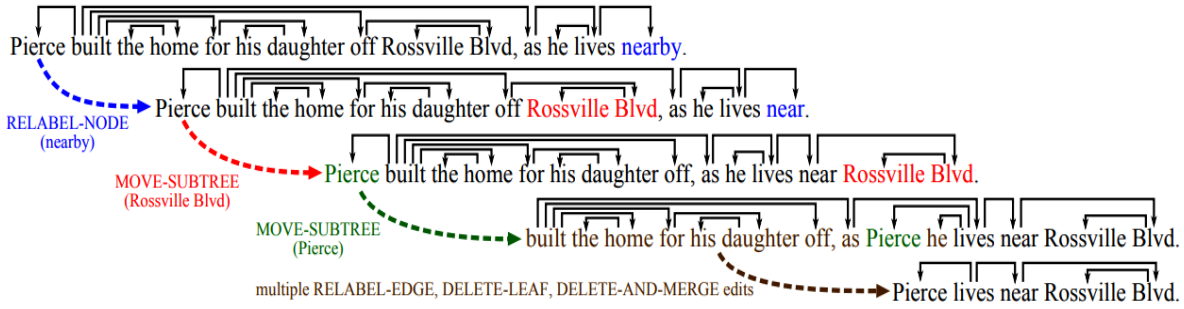

Figure 2.3: A tree edit sequence transforming a premise to an entailed hypothesis as adapted from Heilman and Smith [16]

Similar to dependency based tree edit distance models, Heilman and Smith proposed tree edit models for representing complex tree ordering phenomenon [16]. To better understand the working of tree models, consider the tree transformation in the Figure 2.3. The figure illustrates the entailment from a given premise to the corresponding hypothesis. The tree goes through a series of transformations to reach from the premise to the hypothesis by relabeling *nearby* to *near* and moving *Roseville Blvd* to be a child of *near*. To sequence the series of the edits efficiently, the authors use a logistic regression model that uses 33 syntactic features. The tree edit model was found to perform competitively for paraphrase identification, answer selection in question answering, and in recognizing textual entailment.

Yao et al. consider answer selection as a sequence tagging problem by constructing a linear-chain Conditional Random Field (CRF) based on question-answer pairs to learn the association between the question and answer types [60]. They use Tree Edit Distance (TED) to represent the knowledge of shared structure between the question and answer. One of the advantages of this approach is that the authors do not use manually created

templates for question and answers and thereby avoid human effort. Along with the TED as one of the feature for the classifier, the authors also use traditional features such as POS, NER, and dependency parsing that are known to aid answer selection in question answering.

Practically, however, the approaches discussed above are computationally expensive since they use syntactic and semantic parsers to be applied at runtime and a matching algorithm is often non-trivial requiring additional computational resources. Moving slightly away from the previous work that focusses on answer selection based on the syntactic analysis that is powered by dependency tree matching, Yih et al. improve the effective measures by considering rich lexical semantic resources [61]. Yih et al. focus on shallow semantic components—*lexical components*. That is, they use latent semantic models to capture the hidden word-alignment structures. The authors use WordNet to derive lexical semantic information.

Severyn and Moschitti use tree kernels to automatically learn complex structural patterns between question and answers [36]. For both question as well as a candidate answer, the authors design shallow syntactic trees and connect these trees through relational words (i.e., words that overlap between the question-answer pair). They then proceed to label the nodes of the trees with semantic information and discover more semantic links between the question and the answer based on the named entities. They also do not use any external resources such as WordNet.

One of the drawbacks of the tree edit models is that they have two separate phases: alignment-finding phase and resort to ad-hoc distance metrics. Wang and Manning propose a framework for performing these steps in a single phase by using alignments as structured latent variables [52].

2.1.2 Neural methods

The advantages of the neural methods for answer selection is that they do not involve any complex feature engineering techniques. Additionally, in contrast to the prior work with non-neural methods, the neural methods usually do not depend on any external resources such as WordNet.

The first deep learning approach to answer selection was proposed by Yu et al [64]. The authors created two distributional sentence models: bag-of-words model and a bigram model based on convolutional neural network (CNN). The authors use these models to represent the input question and answer and train a supervised model. The questions and answers are projected into vectors using these distributional models and the model learns

to semantically match the question with its candidate answer. On top of the semantic match, the authors also proposed an enhanced model that combines this semantic match signal with two word matching features. The neural architecture is simple and is illustrated by Figure 2.4.

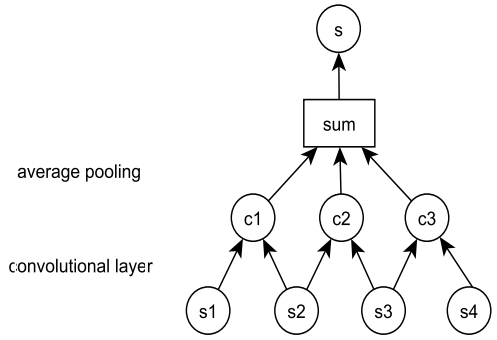

Figure 2.4: The architecture of the one dimensional convolutional neural network as adopted from Yu et al. [64]

Wang and Nyberg proposed stacked bidirectional LSTMs (biLSTM) [51] that sequentially reads words from question and answer sentences assigns a relevance score to the candidate sentences. The candidate answers with entities that are similar in embedding space can be misidentified as the right answers. For instance, *Japan* and *China* could be very similar in the semantic space but *China* may not be the right answer to the question. To discriminate between such answers, the proposed system does exact keyword match on top of the biLSTM relevance model. In contrast to Yu et al [64], this work uses Google's pre-trained 300-dimensional vectors that were trained on Google News corpora [25].

Severyn and Moschitti present a "siamese" structured CNN that learns the representations corresponding to the input question and answer and computes similarity between the learned representations [37]. We call this model SM model (after the authors) and is discussed in detail in the baselines section (see 3.2).

The models that followed the SM model gained complexity and sophistication. For instance, Tan et al proposed a biLSTM model that models question and and answer respectively, which are then connected to a pooling layer, and use a similarity metric to measure the matching between the question and the answer [42]. The simple pooling layer, however, may not retain linguistic information, and therefore the authors use CNN on top of the biLSTMs. The authors reason that this also results in a better embeddings corresponding to the questions and answers. In addition to using CNNs over the biLSTMs, the

authors also propose a simple attention model, which works as a framework for generating answer embedding according to the question context. The architecture of biLSTM CNN is as shown in the Figure 2.5

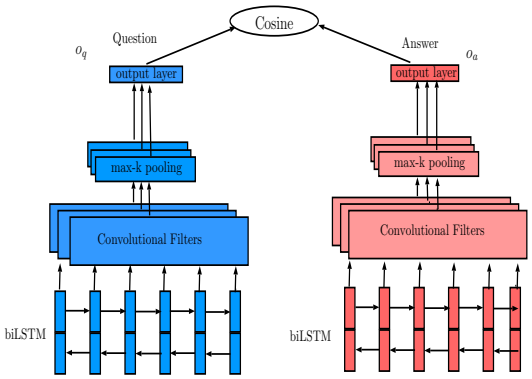

Figure 2.5: The architecture of biLSTM CNN as adopted from Tan et al. [42]

Santos et al. proposed a two-way attention mechanism called *attention pooling*, a discriminative model for pairwise ranking [9]. Through *attention pooling*, the pooling layers in the neural network can be aware of the current input pair (question and answer), and therefore influence the computation of each other's representations. In addition to the learned representations, attention pooling jointly learns a similarity measure over the projected segments of the input pair such as the trigrams of the input pairs and uses the measure to calculate attention vectors in both directions.

All the previous work that has been discussed thus far only consider similarity between the input sentence pair but not the dissimilarity between them. The dissimilarities between the sentence pairs, however, also give signals on the semantics of the sentences. Wang et al. consider both the similarities and disimilaries by decomposing and composing lexical semantics over all sentences [57]. That is, for each vector corresponding to a word, the model calculates a semantic matching vector based on all the other words in the corpus. Following which, each word is represented in terms of a similar and dissimilar vector. The authors use a CNN model to capture the features in the similar and dissimilar components and a similarity score is estimated over the composed feature vectors. Finally, this composed feature vector is used to predict sentence similarity between the input pair of sentences.

Building up on the idea of sentence similarity and dissimilarity by Wang et al. [57], He et al. propose a model that compares the a multiplicity of perspectives. The authors reason that through multiple perspectives, the ambiguity and variability of linguistic expression

can be captured [14]. The authors model questions and answers using a CNN model that can capture features at multiple levels of granularity with multiple types of pooling such as min, max, and average pooling. The model also uses multiple similarity metrics (cosine distance, Euclidean distance, and element-wise difference).

One of the common themes among all neural methods that are discussed in this section thus far is that in all methods model transforms the input questions and answers into a fixed-length vector and then processes these vectors by computing similarity matching on them. However, these representations are coarse-grained and may not capture the essence of the input sentences well. As a way to cope with such a problem, He and Lin propose a method for capturing fine-grained word interactions directly [15]. The authors also employ a novel similarity layer to focus on the important word interactions. The end-to-end model is as shown in the Figure 2.6.

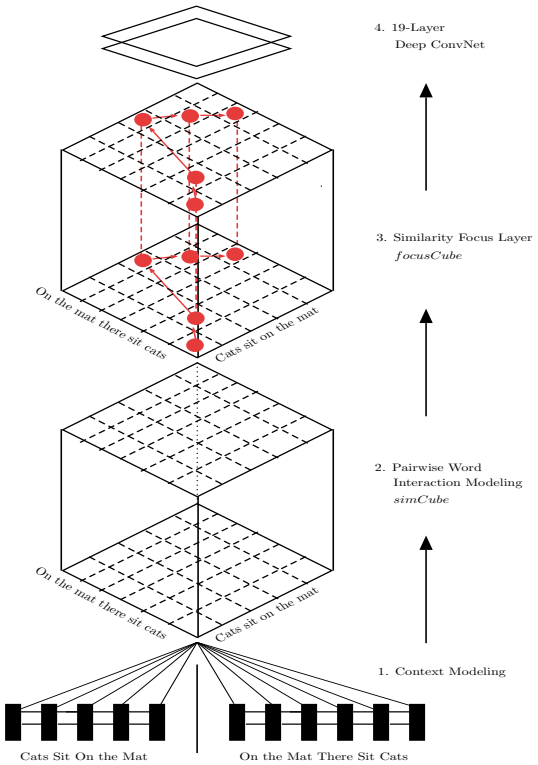

Figure 2.6: The architecture of the pair-wise interaction model as adopted from He et al. [15]

We observe ever increasing penchant towards complex neural network architectures as

a replacement to feature engineering. To quote Yang et al. [58], "Without combining additional features, could we build deep learning models that can achieve comparable or even better performance than methods using feature engineering?". This trend is disconcerting since the authors are trading simple overlap features with complex neural architectures. The authors propose a new architecture: attention based neural matching model (aNMM). The proposed model is equipped with a novel value-shared weighting scheme, and an attention scheme to focus on important question terms.

In contrast to the pointwise approach, where a single answer is learned to be relevant or not to a given a question, Rao et al. propose a pairwise approach to order a pair of answers, where the model learns to rerank the answers based a positive and a negative answer pair [31]. The model learns the joint representations of the triplet input: question, positive answer, and negative answer. The model stacks a triplet ranking loss function to learn from these representations with the goal to minimize the total number of inversions to the ranked list.

Some of the recent neural architectures that have been discussed so far use either convolutional neural networks or recurrent neural architectures to represent the input question and answers followed by complex similarity matching layers [58, 9, 14, 15, 31]. The problem with these architectures, however, is the similarity matching layers, which are new word interaction layers or attention based matching mechanisms, are not suited for practical applications due to the high memory consumption and computation cost incurred. Tay et al. propose an LSTM network with holographic composition, a novel technique to represent the relationship between a question and an answer [43]. The authors show that the holographic composition helps in providing scalable and rich representational learning without trading off for high parameter costs. The authors also propose Holographic Dual LSTM (HDLSTM) to model input sentences and for semantic matching. The model is trained end to end so that the parameters of the LSTM are optimized to best learn the relation between question and answer representations. The end-to-end system with holographic composition is as shown in the Figure 2.7.

Tay et al. explore if state-of-the-art effective measures can be obtained with simple neural architectures [44]. The authors propose a new neural architecture called HyperQA, that is parameter-efficient and outperforms architectures like Attentive Pooling BiLSTMs and Multi-Perspective CNNs. The parameters are reduced is by modelling the relationship between question and answers in Hyperbolic space rather than in an Euclidean space. HyperQA does not require any feature engineering , and does not requir complicated word interaction layers such as the ones described in Multiperspective CNNs [14].

Wang et al. propose a bilateral multi-perspective matching (BiMPM) model to improve

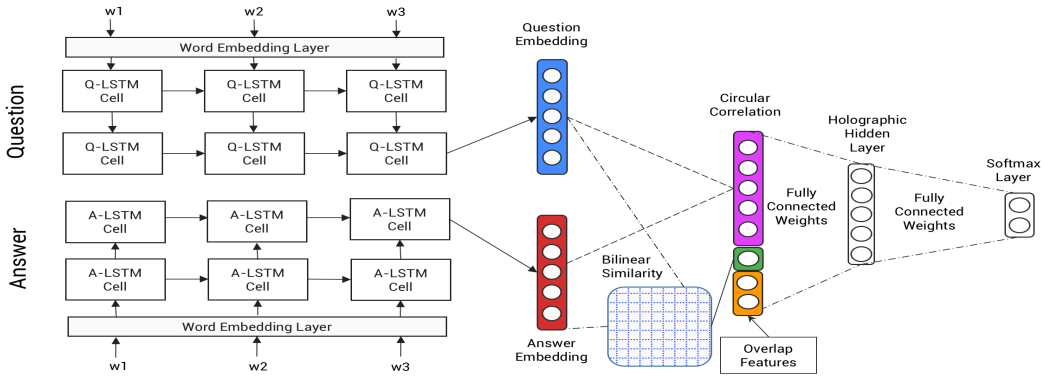

Figure 2.7: Architecture of the Holographic Dual LSTM model (HDLSTM) as adopted from Tay et al. [43]

on the existing similarity layers by matching the encodeded representations of question and answer in both the directions [55]. Given a question and answer as an input, the model first encodes the sentences with a biLSTM, and the encoded sentences are matched in both directions: in the direction of question and in the direction of the answer. For each direction, in each time step, the answer is matched against all timesteps of other answers from multiple perspectives. Finally, another biLSTM is used to aggregate the matching results into a fixed-length matching vector. The matching vector is fed into a fully-connected layer to decide whether the given answer is relevant to the question. The working of BiMPM is as illustrated by Figure 2.8.

Shen et al. try to address the answer selection problem by focussing on semantic similarity between the question and answers at word level [38]. The question and the corresponding candidate answer is first modelled using a biLSTM. The question and answer pair is then used to compute word-similarity matrix between the representations. Meanwhile, the authors reason that each word in a question and answer has different importance and they propose two strategies to capture importance of words: inter-attention layer and inter-skip layer. The weighted vectors are then fed to a fully-connected layer and then to a softmax layer to make final predictions.

Most recent work in answer selection includes work by Tran et al. [48] and Tay et al. [45]. Tran et al. propose a new family of Recurrent Neural Networks called a Context-dependent Additive Recurrent Neural Network (CARNN), which is aimed at developing better dialogue systems by considering external contextual signals (previous utterance in a dialogue system) in addition to the given textual input. The authors use this architecture to use question as the external signal to achieve the state-of-the-art effective measures

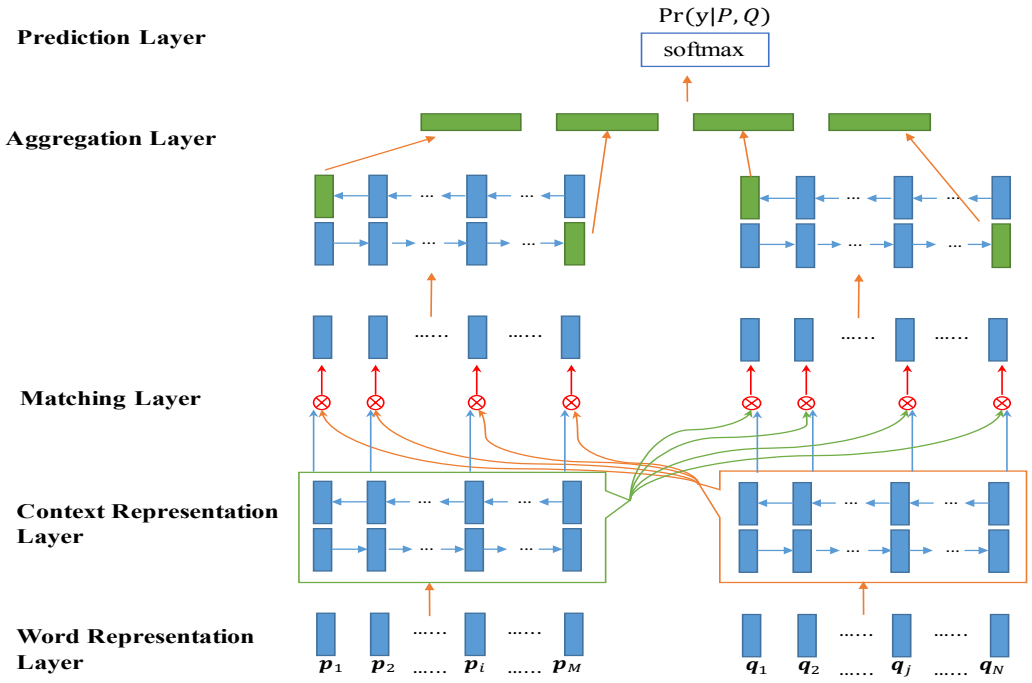

Figure 2.8: Architecture of the bilateral multi-perspective matching model (BiMPM) as adopted from Wang et al. [55]

on the TREC QA dataset. Tay et al. propose a new attention mechanism called Multi-Cast Attention Networks (MCAN) for ranking in conversational modelling and question answering tasks. At the time of writing, MCAN holds the highest state-of-the-art effective measures on the cleaner version of the TREC QA dataset.

2.2 Datasets

For our experiments, we consider two benchmark datasets that have been extensively used by the research community for factoid question answering: TREC QA and WikiQA.

2.2.1 TREC QA

The factoid question answering task was first introduced as a TREC 8 (Text REtrieval Conference) task on the TREC Disk 4 and 5 minus Congressional Record documents [49].

Dataset	Document Collections
TREC8	TREC disks 4&5 minus Congressional Record
TREC9 TREC10	AP newswire (Disks 1-3) Wall Street Journal (Disks 1-2) San Jose Mercury News (Disk 3) Financial Times (Disk 4) Los Angeles Times (Disk 5) Foreign Broadcast Information Service (FBIS) (Disk 5)
TREC11 TREC12 TREC13	AQUAINT disks

Table 2.1: Source document collections for TrecQA.

Set	# Question	# Pos Answers	# Neg Answers
Train	1,229	6,403	47,014
Dev	82	222	926
Test	100	284	1,233
All	1,411	6,909	49,173

Table 2.2: Statistics for various splits of TrecQA.

TREC 8 QA task was the first major large scale evaluation of systems based on the list answers returned and not on the set of documents with respect to the question asked.

Starting with the TREC 8 QA track, question answering systems were evaluated on their end-to-end performance. The participating systems would follow the aforementioned steps and return a single tuple per question; the tuple being the answer string and the document where the answer was found. These submissions were evaluated by TREC organizers based on a *pattern matching* scheme. That is, a simple regular expression matching with answers such as *1977* or *George Bush* accompanied by the *DocID* where this answer was found. This type of evaluation, however, is flawed since it can also accept an incorrect answer—that comes from the wrong sentence but the same document—to be the right answer.

To illustrate this better, consider an example discussed by Wang et al. [53]: for the question *"When did James Dean die?"* The correct sentence that has the right answer to this question is, *"In 1955, actor James Dean was killed in a two-car collision near*

Cholame, Calif." . Therefore, the correct answer is *1955*. But, the same document also had the sentence, *"In 1955, the studio asked him to become a technical adviser on Elia Kazans East of Eden, starring James Dean."* This sentence also has the answer *1955* and if a system had chosen this sentence to be the answer, TREC evaluation would have accepted such answer to be the right answer.

To overcome this problem, Wang et al. use the entire sentence as the answer instead of extracting the part of the answer. This way, the answer is informative to the user as well, as she or he can read the wider context of the sentence. They also focus only on the answer selection part of the question answering pipeline, assuming the candidate answers are given to a question.

Starting with Wang et al. [53], research on question answering has been mostly focussed on the answer selection part. That is, given a question q and "assuming" a candidate set of answers $\{c_1, c_2, \ldots c_n\}$ are given, the answer selection module will select the most relevant answers in the list. Alternatively, this problem can be formulated as a learning-to-rank problem, where the objective of the answer selection system is to the rerank the candidate answers $\{c_1, c_2, \ldots c_n\}$ such that the the most relevant answers appear at the top of the ranked list.

The TREC QA dataset was proposed and built by Wang et al. [53]. TREC 8-12 QA datasets were used for training, and TREC 13 QA dataset was used for testing purposes. The authors automatically chose the candidate answers by picking up the sentences from the document pool that have at least one non-stopword overlap with the question words. To augment the training set with the relevant answers, the authors also added answers with the correct pattern to the training set. The test set was completely manually judged and the first 100 questions of the TREC8-12 training set were also manually judged.

The entire TREC 13 set and the first 100 questions from the training set TREC 8-12 were manually judged. The source details and statistics of TREC QA dataset are as shown in the Table 2.1 and 2.2 respectively.

2.2.2 WikiQA

One of the drawbacks of the TREC QA dataset is that the candidate answers have at least one non-stop question words, and therefore the dataset involves a strong bias on the type of the answers included. Consequently, lexical overlap approaches between the question and answer words might perform better on the TREC QA dataset compared to a natural setting. Borrowing an example from Yang et al. [59], consider an example that illustrates this bias as shown in Figure 2.2.2.

Question: How did Seminole war end?
Answer: Ultimately, the Spanish Crown ceded the colony to United States rule.

Figure 2.9: An example of an answer that does not share non-stop question words

Set	# Question	# Pos Answers	# Neg Answers
Train	873	1,040	7,632
Dev	126	140	990
Test	243	293	2,058
All	1,242	1,473	10,680

Table 2.3: Statistics for various splits of WikiQA.

To overcome the problems that are intrinsic to the TREC QA dataset, Yang et al. [59] proposed the WikiQA dataset. The questions for the WikiQA dataset come from the Bing query logs and the candidate answers are chosen from Wikipedia pages. The candidate answers were labelled through crowd sourcing on Amazon MTurk. The statistics corresponding to the WikiQA dataset are as shown in Table 2.3.

2.3 Metrics

Since the task is a learning-to-rank task, we use metrics that are usually employed to guage the quality of a ranked list. The two standard metrics that are used are:

1. Mean Average Precision (MAP)
2. Mean Reciprocal Rank (MRR)

2.3.1 Mean Average Precision (MAP)

Mean Average Precision quantifies that quality of a ranked list is. That is, more the number of relevant documents at the top of the ranked list, higher will be the MAP score. MAP for a set of questions is calculated by finding the mean of the average precision scores for each query. Average Precision (AP) is defined as following:

$$AP = \frac{1}{|R|} \sum_{i=1}^{Q} P(i) \times rel(i) \tag{2.1}$$

where $|R|$ is the total number of relevant documents,
$P(i)$ is the precision at top i documents, and
$rel(i)$ is 1 if relevant, 0 otherwise.

Therefore, MAP is defined as:

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{2.2}$$

where Q is the total number of questions

To illustrate AP better, consider the ranked list shown in 2.3

$$1, 0, 0, 1, 1, 1 \tag{2.3}$$

Here "1" indicates a relevant answer and "0" indicates a non-relevant answer. The precision calculated at each point would be as shown in 2.5

$$1/1, 0, 0, 2/4, 3/5, 4/6 \tag{2.4}$$

The AP would therefore be the precision calculated at each point and averaged over total number of relevant documents:

$$AP = (1 + 2/4 + 3/5 + 4/6)/4 = 0.6917 \tag{2.5}$$

So, how do we compare two AP scores? That is, how good is, say, an AP score of 0.5 over an AP score 0.1 and how can this be reflected in terms of the quality of the ranked list?

Consider a list that has relevant answers at alternative indices:

$$0, 1, 0, 1, 0, 1, 0, 1, ... \tag{2.6}$$

Clearly, the AP of this list would be 0.5. Alternatively, consider another ranked list with relevant answers at every third index:

$$0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, ... \tag{2.7}$$

The AP score of this list would be 0.333. To recapitulate, if the AP is 0.5, the relevant answer occurs at every second position. If the AP score is 0.3, the relevant answer occurs at every 3rd position and an AP score of 0.1 would mean that every 10th answer is correct.

2.3.2 Mean Reciprocal Ranking

Mean Reciprocal Ranking (MRR) indicates how well the first relevant answer is placed. That is, MRR can be thought to be equivalent to MAP with only the first relevant answer considered. More formally,

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank(Q_i)} \tag{2.8}$$

where N is the total number of questions
$rank(Q_i)$ is the index at which relevant answer is found for question Q_i

The MRR corresponding to the ranked list in equation 2.6 and equation 2.7 are 0.5 and 0.3333 respectively.

Chapter 3

Baselines

In this chapter, we explore various baselines both neural and non-neural, and present results using them.

3.1 Non-nerual Baselines

The non-neural baselines for question answering for TREC QA and WikiQA datasets are fairly competitive and exhibit MAP/MRR scores that are very close to the state-of-the-art systems.

Word overlap is the simplest baseline that captures what fraction of non-stop question words are present in the candidate answer. The baseline ranks the candidate answer that has higher number of non-stop question words over the candidate sentences that have lower non-stop question words. The baseline, therefore, inherently flawed since i) there can be two candidate sentences with the exact number of non-stop question words ii) a candidate answer can still be a relevant answer without having an exact match with non-stop question words. That is, word overlap only captures lexical similarity and not semantic similarity between the words.

3.1.1 Weighted Word Overlap

As discussed in the earlier section, the word overlap baseline considers candidate answer sentences with equal number of non-stop question words equally; thereby, does not discriminate between relevant and non-relevant answers.

One of the ways to deal with this problem is to weight the non-stop question words based by their importance. This way, two sentences containing the same number of non-stop question words will be ranked differently depending on the importance of the non-stop question words they possess.

The weighted-word-overlap baseline works exactly on the same principles, where the non-stop question words are weighted by the Inverse Document Frequency (IDF) score. IDF is defined in Equation 3.1. We notice that rarer a word, lower is its document frequency *df*, and therefore more informative or important is the word.

$$idf(t, D) = \log \frac{N}{|1 + \{d \in D : t \in d\}|} \tag{3.1}$$

where N: total number of documents in the corpus

$|\{d \in D : t \in d\}|$ is the number of documents where the term t appears. Also, known as the document frequency *df*.

The working of weighted-word-overlap baseline is as shown in Figure 3.1.

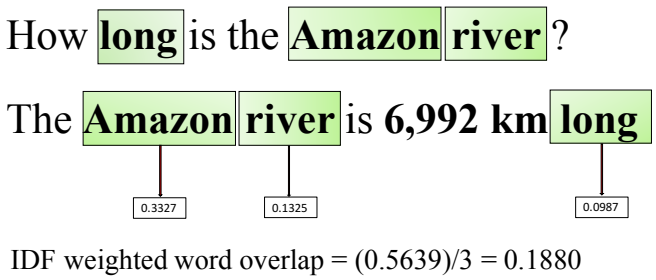

Figure 3.1: Illustration of IDF weighted word overlap baseline.

3.1.2 Word Mover's Distance

Word Mover's Distance (WMD) is a metric that is similar to the well-known Earth Mover's Distance metric (proposed by Rubner et al. [33]) and measures the distance between two documents. The WMD between two documents A and B is defined as thee minimum cumulative distance that words from document A need to travel to match exactly the point cloud of document B [20]. More formally, WMD is as defined as in Equation 3.2.

$$WMD(q, a) = \sum_{w \in q} \min_{w' \in a} dist(\vec{w}, \vec{w}') \tag{3.2}$$

where q is a question
a is a candidate answer
and *dist(.)* is Euclidean distance

WMD can perform both lexical and semantic matches and therefore the method overcomes the drawbacks of lexical matches that are inherrent in the word overlap baselines The method is also a simple hyper-parameter-free model that is easy to use and to interpret.

We use WMD to capture the distance between two sentences and therefore the similarity between a question and a candidate answer. Lower the distance the words in question have to move to be similar to the answer, more relevant the candidate answer would be to the question. We represent each question and the answer with word vectors by using the pretrained word vectors from Mikolov et al. [25]. We then calculate the WMD between a question and the corresponding candidate answers and based on the score the candidate answers are reranked. The working example of WMD is as illustrated by the Figure 3.2.

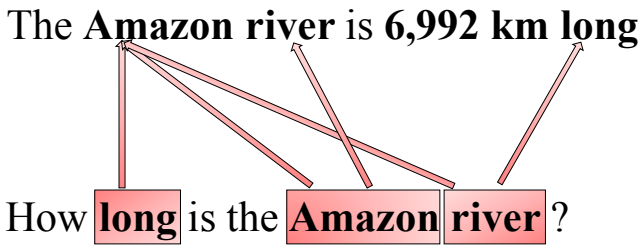

Figure 3.2: Working of Word Mover's Distance

3.1.3 TextFlow

TextFlow is a text similarity measure that computes the distance between two input sentences by representing the input text pairs as continuous curves and by using both the actual position of the words and sequence matching to compute the similarity value [27].

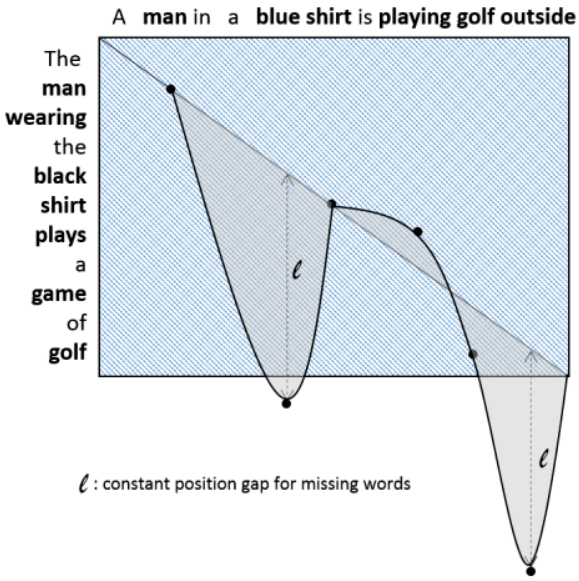

Figure 3.3: Working of TextFlow as adopted from Mrabet et al. [27]

It is a stand-alone similarity measure, whose parameters can be pre-tuned when needed with the help of a small dataset. The working of TextFlow is as illustrate by Figure 3.3. The word sequences corresponding to the input text pair is represented along the X and Y axis. The area under the curve oscillating around the diagonal indicates the similarity between the text pairs. Greater the area under the curve, lower is the similarity between the input text pairs. TextFlow between two input sentences is a real number and is in the interval [0,1]; 0 indicating no match and 1 indicating exact match. It is an asymmetric measure and therefore TextFlow(X, Y) is not same as TextFlow(Y, X).

We treat a question and the corresponding candidate answer to be input text pairs and we rerank the candidate answers based on their TextFlow scores.

3.2 Neural Baseline: SM Model

SM model is a Convulational Neural Network based model that was proposed by Severyn and Moschitti [37]. The model follows a Siamese structure with two wings, a wing for

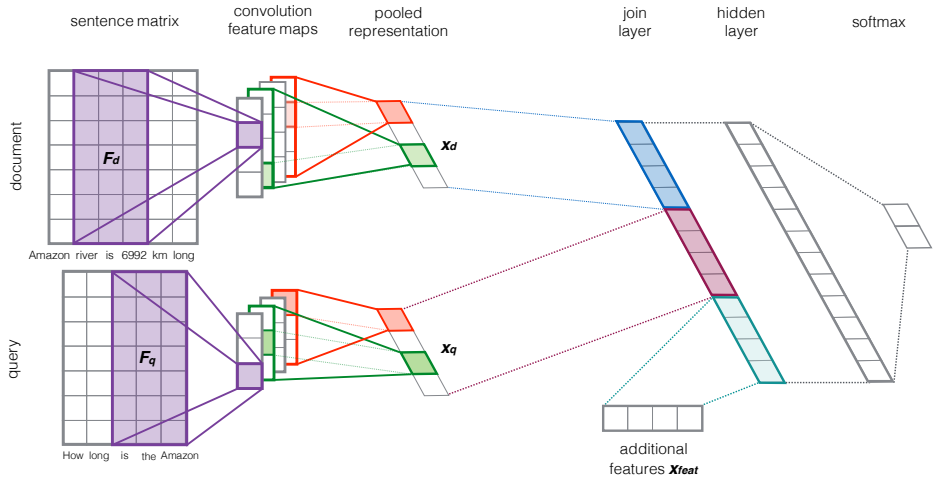

Figure 3.4: The architecture of SM model as adopted from Sequiera et al. [35]

question and answer respectively. In this structure, a question and a candidate answer is processed at the same time.

The word embeddings of the question and answer is fed as input to each of the wings. That is, each word in the question and the answer is first converted into a continuous distributional vector such that if two words are similar in semantic space, their corresponding vectors would also be similar in the vector space (through cosine similarity, euclidean distance, etc.)

The convulational filters then convolve over these input matrices, followed by a RELU activation and max-pooling yielding feature maps corresponding to the question and the candidate answer. In the figure 3.4, x_d corresponds to the feature maps of the answer and x_q refers to the feature map of the query. Note that, unlike traditional machine learning methods, these features are automatically generated.

In the original paper, the feature maps x_d and x_q are concatenated into a single vector:

$$\mathrm{x}_{\mathrm{join}} = [\mathrm{x}_q^T; x_{\mathrm{sim}}; \mathrm{x}_d^T; \mathrm{x}_{\mathrm{feat}}^T] \tag{3.3}$$

where x_{join} indicates the bilinear join layer and x_{sim} refers to the bilinear similarity between x_q and x_d:

$$\mathrm{sim}(\mathrm{x}_q, \mathrm{x}_d) = \mathrm{x}_q^T M \mathrm{x}_d \tag{3.4}$$

This similarity matrix M is a parameter of the network that is optimized during training. Rao et al. found that the existence of this matrix did not affect the effective measures

(and hurt the effective measures slightly) and therefore we have removed the matrix from the subsequent implementation of the SM model [32].

The feature maps are then fed to a fully connected neural network along with four additional external features (represented by x_{feat} in the picture):

1. word overlap before removing stop words
2. word overlap after removing stop words
3. IDF weighted word overlap before removing stop words
4. IDF weighted word overlap after removing stop words

Table 3.1: Baseline resultson TREC QA dataset

Method	**MAP**	**MRR**
Word overlap	0.650	0.681
Weighted word overlap	0.701	0.769
WMD	0.733	0.791
TextFlow	0.704	0.762
SM Model	0.768	0.821

Table 3.2: Baseline resultson WikiQA dataset

Method	**MAP**	**MRR**
Word overlap	0.494	0.496
Weighted word overlap	0.560	0.564
WMD	0.604	0.614
TextFlow	0.531	0.535
SM Model	0.688	0.709

The external features x_{feat} are very useful and it was found that external features along with the convolutional filter maps result in the effectiveness of SM model [32]. Finally, the output of the fully connected layer is fed to a SoftMax layer and a score for each class i.e., relevant or non-relevant is generated. This score corresponding to the candidate answers is then used to rerank the list so that most relevant answers appear at top of the ranked list.

In addition to removing the bilinear matrix from the join layer, we also made some other simplifications/modifications to the original SM model:

1. The original SM model had static embeddings in the input layer. That is, the gradients during the training were not back-propagated to the input layer. With our implementation, we back-propagate the gradients to the input space for all our experiments.

2. The original SM model uses $ReLU$ as the activation function but we use *tanh*. We did not see any difference in terms of effective measures when one of the two activation functions were used and we retained the *tanh* activation function.

3. Following Rao et al., we removed the similarity matrix M since its existence does not affect the effective measures [32].

The MAP, MRR scores obtained for each of above discussed baselines for TREC QA and WikiQA dataseta are shown in Table 3.1 and 3.2 respectively.

Chapter 4

Proposed Model

In this chapter, we describe in detail our proposed model. We build up the linguistically annotated input for CNNs (Lai-CNN) over the SM model that was discussed in the previous chapter. But, the question is how do we incorporate the linguistic features into the neural network? Chen et al., for instance, proposed an elaborate set of features to be added as external features to SM model [5]. The authors propose these features at three different levels: lexical and semantic matching, readability, and focus features. These sets of features—21 of them in total—capture either the interaction between the question and answer or are overlap features like BM25. We, on the other hand, want to annotate each word in the question and the answer with its linguistic features and therefore these features cannot be incorporated as external signals.

Therefore, we would have to modify the input that is fed to SM model by linguistically annotating it. The SM model takes an input matrix of word-embedding as the input, where each word is represented as a vector. We linguistically annotate the input matrix before feeding it to the neural network.

As discussed in the previous chapter, the SM model follows a "Siamese" structure with two wings, a wing for a question and another for an answer, which are processed simultaneously. The illustration of each arm is as shown in Figure 4.1.

Following Peng and Lu [28], in addition to the word embedding, we concatenated additional linguistic features in each input "arm". However, there are a plethora of linguistic features to choose from and a neural network, the network could easily overfit if an inordinate number of features were added affecting the effective measures.

We only choose the linguistic features that have known to help in the answer selection task (see Yao et al. [60]). We also order the linguistic features to be added to the input of

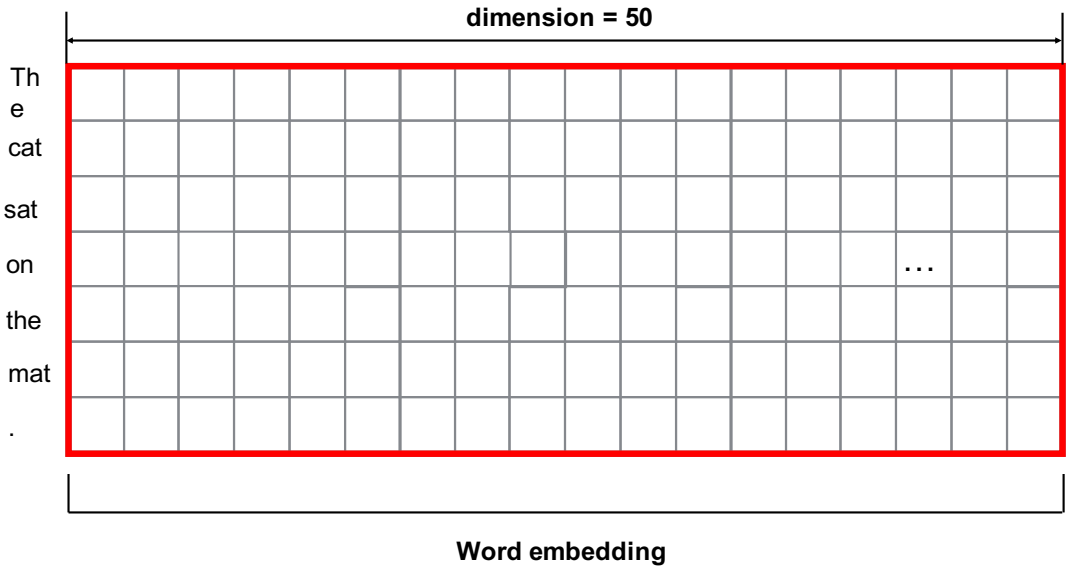

Figure 4.1: The input word embedding to SM CNN. Note that this figure is the rotated form of the input word embedding matrix in Figure 3.4

the network based on the difficulty of the task i.e., how hard is it to train the corresponding natural language processing task. Accordingly, we enumerate the annotations in the following order:

1. Part-of-speech Tagging (POS)
2. Named Entity Recognition (NER)
3. Dependency parsing

4.1 Part-of-speech Tagging

POS tagging is a task of identifying the grammatical units of words in a given sentence. The POS of a word gives details on the grammatical function of the word in an utterance and helps in disambiguating the word's purpose. It not only gives a lot of information about the word but also about its neighbouring words. Examples of POS include nouns, verbs, adjectives etc.

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Figure 4.2: PTB tagset as adopted from Jurafsky et al. [17]

English has a standardized POS tagsets called the Penn-Tree-Bank tagset (PTB for short). PTB tagset has 45 tags in total and are shown as by Figure 4.2. These tags are fine-grained and since our task just needs general signal of what the grammatical class of the word is and to avoid overfitting of the model with too many tags, we map the PTB POS tag set to a universal tag set proposed by Petrov et al. [29]. The universal POS tagset is shown in Figure 4.3.

The rationale behind using POS tags for answer selection is that the POS tags accentuate the signals that could be important for matching factoid question answers. Consider, for instance, the following question:

Q: what is the ***longest river*** *in the world ?*

Here the words *longest* and *river* form an ADJECTIVE-NOUN pair and a relevant answer will have a similar pair. Helping the neural network to recognize such patterns helps in

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Figure 4.3: Universal POS tagset as proposed by Petrov et al. [29]

the ranking of relevant answers better.

POS tagging has been extensively studied by the NLP research community with word-level accuracy of 97.33% on the WSJ dataset [39]. POS tagging is treated as a sequence tagging task and even simple HMM based models yield word-level accuracy as high as 96.7%. With the state-of-the-art accuracy as high as 97.33%, POS tagging is a fairly easier task; especially, considering the human error on the task is 3% [23].

4.2 Named Entity Recognition

A named entity is, roughly speaking, anything that can be referred to with a proper name: a person, a location, or an organization [17]. However, named entities also include temporal expressions such as dates, time, and numerical expression such as prices. The list of generic named entities along with the examples is as shown by Figure 4.4.

Named Entity Recognition (NER) is a task that recognizes the entities and identifies the categories of the named entities. Consider the following sentence that is annotated with its named entities:

Example: [Germany']$_{LOCATION}$s representative to the [European Union']$_{ORGANIZATION}$ veterinary committee [Werner Zwingman]$_{PERSON}$ said on [Wednesday]$_{DATE}$ consumers should ...

In the above example Germany is the name of a country, European Union is the name of an organization and so on.

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	The **Mt. Sanitas** loop is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states, provinces	**Palo Alto** is raising the fees for parking.
Facility	FAC	bridges, buildings, airports	Consider the **Tappan Zee Bridge**.
Vehicles	VEH	planes, trains, automobiles	It was a classic **Ford Falcon**.

Figure 4.4: A list of named entities as adopted from Jurafsky et al. [17]

NER is a difficult task due to the ambiguity of segmentation while recognizing the entity since a named entity recognizer would have to identify where the entity begins and where it ends. Additionally, identifying the type of the entity on its own can also be ambiguous since the named entity can seemingly belong more than one category. Consider, for example, the word *JFK*, which might refer to a person, an airport, a school, a bridge etc.

The use of NER features in the input to the neural network for answer selection can be justified easily. Since the task focuses on factoid answer selection, most of the questions and answers will have named entities in them and helping the neural network find the named entities will rank the relevant answers better. To illustrate this, consider a sample factoid question:

*Q: When was **Albert Einstein** born ?*

The question has the words Albert Einstein, which is a named entity. The answer to this question will also have a date as the answer which is also a named entity and a neural network that is trained to use named entities as a signal for answer selection would therefore perform well on answer selection.

NER is also treated as a sequence labelling task where the assigned tags identify both boundary and type of the entity. Sequence taggers like CRF, MEMM are trained on the labelled data. Named Entity Recognition as a tagging task is not as easy as POS tagging and the state-of-the-art system exhibits 89.31% accuracy over the CoNLL 2003 dataset [1].

4.3 Dependency Parsing

Dependency parsing gives the dependency information between two words in a sentence. It is the binary relationship between two words in an utterance. Several NLP systems use dependency information to understand how two words are related that cannot otherwise

be understood through bag-of-words approaches. The syntactic structure of the sentence can solely be explained by the words and the binary grammatical associations between them [17]. The dependencies are called typed dependencies since the type of the dependency is drawn from the pre-established set of grammatical relations. The dependency tree includes a *ROOT* relation, which is the head of the entire structure. A dependency tree three parts: head, dependent, and a grammatical function. A set of grammatical relationships are as shown in Figure 4.5.

Clausal Argument Relations	**Description**
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Figure 4.5: Examples of dependency relaltionships as adopted from Jurafsky et al. [17]

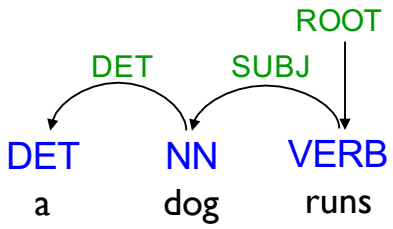

Figure 4.6: Dependency parsing of a simple sentence

Consider the example sentence as shown in the Figure 4.6. In this example, we can see that the word *a* is related to the word *dog* by *a* being the DETERMINER of the word *dog*. Here, DETERMINER is a grammatical function, *dog* is a dependent word, and *a* is the

head. Similarly, *dog* is the SUBJECT of the verb *runs* and *runs* is the ROOT verb in this sentence and *runs* is the head of the word *dog* and ROOT is the head of the word *runs*.

There could also be longer dependencies that can be captured by the dependency parse as shown in the Figure 4.7. Adding dependency features as a linguistic feature to our CNN based model helps the network in capturing the long distance dependencies that are usually missed by CNNs since model shorter passages.

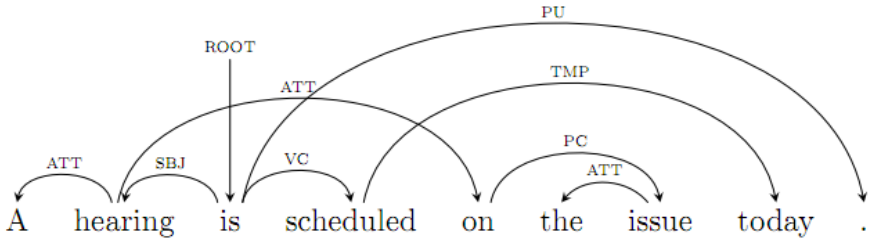

Figure 4.7: Longer dependencies while dependency parsing

Dependency parsing is the hardest of the NLP tasks that we had enumerated earlier. The state-of-the-art (non-neural) dependency parse achieves a UAS score of 88% [3].

Each of the linguistic features mentioned in this chapter will be added to the input layer of the CNN. The sans-linguistic features, however, are word embeddings in semantic space of d dimension. Therefore, the linguistic features that are added to this input layer should also be mapped to an embedding space. We do this by representing the linguistic features in one-hot representation. That is, we begin with an array of 0's and we assign 1 to the index of the linguistic feature. This sparse representation of the input can be paralleled to the vector representation of the corresponding linguistic feature.

After concatenating the word embedding with the linguistic features, the input to the network is as shown in Figure 4.8. The row corresponding to each word represents the word embedding of the word, and the representation of its linguistic features. Consider, for instance, the word *cat*. The POS of the word is *NN* (a Noun), the head of the word is *the*. The word *the* is related to *cat* by a dependency relation *det* (ie.,*the* is the determiner of the word *cat*). The word *cat* is not a named entity and therefore is represented by *O* (no entity).

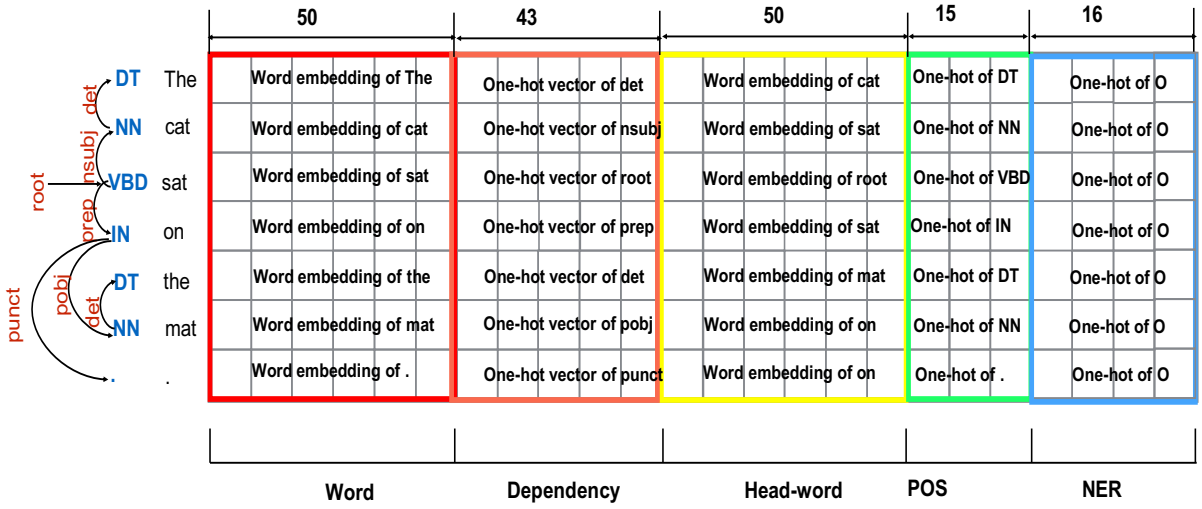

Figure 4.8: The constituents of LAI-CNN. Note that this figure is the rotated form of the input word embedding matrix in Figure 3.4

The numbers corresponding to each embedding represent the dimension of the corresponding embedding space. The figure also indicates the corresponding representation of linguistic features: word and head-word features are represented with word embedding features (word embedding in the figure) whereas POS, NER, and dependency features are represented using one-hot representation (one-hot vectors in the figure). Going back to the example word *cat*, the first feature Word is represented as a word embedding, the dependency feature *det* is represented as a one-hot vector. The headword *the* is represented as a word embedding. Finally, the POS and NER features *NN* and *O* are represented by one-hot vectors respectively.

4.4 Experimental Setup

In this section, we present the implementation details pertaining to our proposed model that was discussed in detail in the earlier chapter. We annotate the SM model with linguistic features by modifying the word embedding matrix to the neural network. In the end, the word embedding matrix that was input to the neural network included the following features:

4.4.1 Word Embedding

We used a 50-dimensional *word2vec* word embedding built on the Wikipedia dump and the AQUAINT collections. The word embedding was trained by using Google's *word2vec* model [25] and was provided by the authors of SM model [37]. If an embedding corresponding to a word was not found during the initialization (out of vocabulary words), the word embedding for such words was initialized with embeddings sampled from a uniform distribution U[-0.25, 0.25].

4.4.2 Part-of-speech

Before the training our model, we run the datasets through Stanford's Log-linear POS tagger [47]. The POS tagger tags the input text with a set of PTB tags. The PTB tagset has 45 tags and therefore a sparse representation of the tags would result in larger word-embedding matrix, possibly resulting in overfitting. Moreover, we are only interested in the signals such as an ADJECTIVE-NOUN pair that can also be generated through course POS tagset.

We, therefore, mapped the POS tags of each word into the universal POS tag set [29]. Finally, we represented the universal POS tags as a 15-dimensional one-hot vectors.

4.4.3 NER

The sentences in the dataset are tagged with Named Entities using Stanford's CRF sequence models [12]. The NER tagset had the following tags: PERSON, LOCATION, ORGANIZATION, MISC, MONEY, NUMBER, ORDINAL, PERCENT, DATE, TIME, DURATION, and SET. We represented NER tags as a 16-dimensional one-hot vector.

4.4.4 Dependency + Headword

Dependency parsing can be thought of as a binary relationship between two words in a sentence connected by a grammatical relationship. Therefore, while dependency parsing a sentence, we get a tuple of the word's headword and the grammatical relationship. We use Stanford's dependency parser to obtain a set of universal dependencies [34].

We used the basic universal dependencies obtained from CoreNLP dependency parser. The dependency feature of each word is represented as a 43-dimensional one-hot vector.

As for the headword, since the headword is yet another word belonging to the sentence, we use the same initial 50-dimensional *word2vec* word embedding to represent the headwords.

We implemented the proposed neural network with PyTorch deep learning toolkit (verion: 0.2.0). We performed parameter tuning on both the TREC QA and WikiQA datasets separately. We used Adadelta as the optimizer with the initial learning rate of 1.0 for TrecQA dataset and 0.96 for WikiQA. For both datasets, we used a batch size of 64. We used *tanh* as the activation function with a dropout rate ($p = 0.5$) in the fully connected layer. For both models we used a single kernel with a filter width of size 5. As for parameter regularization, we used $\lambda = 10^{-5}$ and $\lambda = 0.95 \times 10^{-4}$ for TREC QA and WikiQA respectively.

We first train the model with individual linguistic features: POS, NER, and dependency features (Headword + Dep). We combine these features to provide two sets of cumulative features:

1. POS + NER
2. POS + NER + Headword + Dep

Chapter 5

Experimental Results

The results on the TREC QA and WikiQA datasets are as shown in the Table 5.1 and 5.2 respectively. The effective measures corresponding to previous work in answer selection for question answering. For the TREC QA dataset, we enumerate the results by referring to the ACL Wiki page for question answering [1]. For WikiQA, we manually discover the previous work on the WikiQA dataset and cite the effective measures.

In the results corresponding to the TREC QA dataset in Table 5.1, we see the effective measures by Punyakanok et al. [30] and Cui et al. [8] are considerably lower. This is because the TREC QA dataset was not proposed until 2007 and Wang et al. [53] revisited their approaches on the TREC QA dataset.

The other surprising fact is that simple lexical baselines in Table 3.1 perform better than several linguistic and Machine Learning approaches. We can easily reason the utility of word-overlap methods for the TREC QA dataset by revisiting how TREC QA dataset was created (see Section 2.2.1). Historically, question answering task was evaluated in an end-to-end fashion with ad-hoc retrieval at the beginning of the pipeline and answer selection at the end of the pipeline. The TREC QA dataset was also created from TREC QA tracks which followed the above-mentioned evaluation strategy. Since ad-hoc retrieval methods such as BM25 and QL (Query Likelihood) work based on word-overlap mechanisms, the dataset favours word-overlap techniques. This also explains the gains obtained by the external feature x_{feat} layer that is fed into the SM model, which essentially consists of word overlap features. Additionally, the WMD baseline from Table 3.1 is not far behind compared to the state-of-the-art system for TREC QA dataset, indicating that WMD captures the semantic overlaps that could be missed by the word-overlap methods.

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

Table 5.1: Results on the TREC QA dataset

Reference	**MAP**	**MRR**
Punyakanok et al. (2004) [30]	0.419	0.494
Cui et al. (2005) [8]	0.427	0.526
Wang et al. (2007) [53]	0.603	0.685
Heilman and Smith (2010) [16]	0.609	0.692
Wang and Manning (2010) [52]	0.595	0.695
Yao et al. (2013) [60]	0.631	0.748
Severyn and Moschitti (2013) [36]	0.678	0.736
Yih et al. (2013) [61]	0.709	0.770
Yu et al. (2014) [64]	0.711	0.785
Wang and Nyberg (2015) [56]	0.713	0.792
Feng et al. (2015) [10]	0.711	0.800
Yang et al. (2016) [58]	0.750	0.811
He et al. (2015) [14]	0.762	0.830
He and Lin (2016) [15]	0.758	0.822
Tay et al. (2017) [43]	0.770	0.825
Rao et al. (2016) [31]	0.780	0.834
Chen et al. (2017) [5]	0.782	0.837
SM model	0.768 [0.766, 0.769]	0.821 [0.811, 0.827]
Linguistic features		
POS	0.774 [0.768, 0.778]	0.832 [0.816, 0.848]
NER	0.776 [0.768, 0.786]	0.837 [0.832, 0.840]
Headword + Dep	0.777 [0.775, 0.781]	0.837 [0.825, 0.848]
Cumulative features		
POS + NER	0.779 [0.777, 0.785]	0.846 [0.840, 0.853]
POS + NER + Headword + Dep	0.781 [0.776, 0.793]	0.843 [0.831, 0.854]

The WMD baseline also performs competitively with respect to the WikiQA dataset with effective measures comparable with Yu et al. [64].

Table 5.2: Results on the WikiQA dataset

Reference	**MAP**	**MRR**
Yu et al. (2014) [64]	0.619	0.628
Yang et al. (2015) [59]	0.652	0.665
Dos Santos et al. (2016) [9]	0.689	0.696
Miao et al. (2016) [24]	0.689	0.707
Yin et al. (2016) [63]	0.692	0.711
He and Lin (2016) [15]	0.709	0.723
Rao et al. (2016) [31]	0.701	0.718
Tay et al. (2017) [43]	0.712	0.727
Wang et al. (2017) [55]	0.718	0.731
Yin and Schtze (2017) [62]	0.712	0.723
Chen et al. (2017) [4]	0.721	0.731
Wang et al. (2016) [50]	0.734	0.742
Wang and Jiang (2016) [54]	0.743	0.755
SM Model	0.688 [0.682, 0.692]	0.709 [0.706, 0.711]
Linguistic features		
POS	0.701 [0.695, 0.704]	0.721 [0.715, 0.725]
NER	0.696 [0.688, 0.704]	0.716 [0.706, 0.726]
Headword + Dep	0.696 [0.689, 0.707]	0.712 [0.702, 0.723]
Cumulative features		
POS + NER	0.700 [0.696, 0.705]	0.718 [0.712, 0.723]
POS + NER + Headword + Dep	0.704 [0.698, 0.711]	0.720 [0.710, 0.731]

We notice that while some of the sophisticated and computationally expensive models only give slight improvements in the effective measures over the existing systems [58, 9, 14, 15, 31], our proposed model shows similar increments although by following a simple architecture. Consider, for instance, Yang et al.[58], who use an attention based neural matching model (aNMM), a computationally expensive model with a value-shared weighting scheme as well as the attention scheme to focus on important question words. The proposed model, however, begins with a simple "Siamese" architecture and changes the input fed into the neural network but does obtain considerable gains in the effective measures.

While conducting our experiments, we noticed that the effective measures would vary by a margin of 3 to 4 points. This sort of variance in the results caused by the selection of random seeds is well known in the community [7]. We, therefore, repeated the experiments

on 5 different random seeds and tuned the hyper-parameters for each seed. The format of the results are $average[min, max]$. Comparing our approach with the previous work in question answering area, we are the first ones to report results on k different random seeds. Therefore, it is fair to compare the maximum values of effective measures obtained with Lai-CNNs with the existing systems. We, however, show that the results averaged over a set of random seeds also outperform the existing approaches.

We begin with the modified version of SM model that has an average (MAP, MRR) scores of (0.768, 0.821) for TREC QA. The maximum MAP and MRR scores for the dataset are 0.769 and 0.827 respectively. Just by adding the POS linguistic features to the input matrix of the SM model, we notice that Lai-CNN beats the state-of-the-art system in terms of MRR by obtaining a maximum MRR score of 0.848—a 2% improvement over the baseline. We see similar effects with the addition of NER or dependency features. With the addition of NER features, the maximum MAP score exceeds the MAP score of the state-of-the-art system. Choosing the maximum effective measures obtained in this configuration alone would rival state-of-the-art systems. Similarly, we cumulate the above-mentioned linguistic features. With the addition of POS + NER features, the MRR score averaged over 5 random seeds exceeds the state-of-the-art MRR score. Meanwhile, the maximum effective measures in this configuration show 2% and 3% improvement over the baseline with respect to MAP and MRR respectively. Finally, with the addition of POS + NER + Headword + Dep features, the average MAP obtained is 0.781, which is comparable with the state-of-the-art system and average MRR obtained is 0.843, which beats the state-of-the-art MRR. The maximum effective measures, once again, far exceed the state-of-the-art measures .

Similarly, for the WikiQA dataset, we begin with SM model that has an average (MAP, MRR) score of (0.688, 0.709). The SM model has a maximum MAP and MRR score of 0.692 and 0.711 respectively. Just as we saw in the TREC QA dataset, a simple addition of one of the linguistic features results in considerable gains. We obtain a maximum MAP of 0.707 and a maximum MRR of 0.726 just by adding one of the linguistic features. Similarly, by adding the cumulative linguistic features POS + NER + Headword + Dep, we obtain a maximum MAP of 0.711 and an MRR of 0.731 which is a 2% improvement over the baseline. The improvements with WikiQA dataset do not outperform the state-of-the-art effective measures but nevertheless contribute to improvements over the baseline. We notice that the gains in the effective measures over the baseline for both the datasets are $\tilde{2}$% but the WikiQA dataset has stronger state-of-the-art systems.

To recapitulate, we notice by adding individual linguistic features alone, gains in effective measures are evident in both the datasets. In case of the TREC QA dataset, the gains are huge and just adding one of the linguistic features rivals the existing state-of-the-art

MRR score. Furthermore, adding the cumulative linguistic features, extends the gains with POS + NER + HEADWORD + DEP rivalling the existing state-of-the-art effective measures.

In contrast to the previous systems in this area, which report results on a (probably hand picked) random seed, we report our results on 5 different random seeds and show that the gains are observable even while averaging the effective measures across the random seeds. Similar to the TREC QA dataset, the gains in the WikiQA dataset are also easily noticeable with just the addition of any of the aforementioned linguistic features. Although the gains on the WikiQA dataset do not beat the state-of-the-art effective measures, the gains achieved by adding the linguistic features are, nevertheless, commendable.

Chapter 6

Conclusion

In this thesis, we proposed a novel way to combine the linguistic features into the word embedding matrix of the neural networks. We begin by modifying the SM model, which is a "Siamese" structured model both question and answer input word embedding matrices processed simultaneously. We carefully choose a set of features to be added to the SM model so as to help the model learn better for an answer selection task.

We argue against the common trend of resolving to more complicated and sophisticated neural networks, which have increasingly been used in the question answering community. We also make the best use of decades old research spent on developing models like POS taggers that exhibit human-like accuracies over standardized datasets, that have been abandoned with the advent of neural networks and with a goal of doing NLP "from scratch". We add the the following linguistic features that are known to aid answer selection: POS, NER, and dependency features. We come up with simple ways of representing these feature and adding them to the input matrix of the neural network. By using just three simple and intuitive linguistic features, we strike a good balance between feature engineering and complex neural models.

We show that the linguistic features not only contribute to increment in effective measures over the baseline, but just the addition of one of the linguistic features rivals the state-of-the-art effective measures for the TREC QA dataset and also considerable gains over the WikiQA dataset. We, therefore, hold that Linguistically Annotated Input CNNs may not get a system a place in the leader-board but will definitely yield considerable increments in the effective measures.

The proposed Lai-CNN model adds considerable improvements over a simple neural baseline for answer selection in question answering systems. As a future work, we aim to

extend Lai-CNN for other NLP tasks such as sentence classification and text entailment. Several existing models for these approaches are CNN based. For example, models like Kim-CNN [18] and Mp-CNN [14] use CNNs for sentence classification and text entailment. These models can can easily be modified to just by linguistically annotating the input to the models. We predict that by linguistically annotating the input to the neural models, noticeable improvements in the effective measures can be obtained.

References

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

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

[3] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[4] Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, Liang He, and Weijie An. Enhancing recurrent neural networks with positional attention for question answering. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 993–996. ACM, 2017.

[5] Ruey-Cheng Chen, Evi Yulianti, Mark Sanderson, and W Bruce Croft. On the benefit of incorporating external features in a neural architecture for answer sentence selection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1017–1020. ACM, 2017.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[7] Matt Crane. Questionable answers in question answering research: Reproducibility and variability of published results. *Transactions of the Association for Computational Linguistics*, 6:241–252, 2018.

[8] Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 400–407. ACM, 2005.

[9] Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *CoRR, abs/1602.03609*, 2(3):4, 2016.

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

[11] Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52, 2008.

[12] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[13] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

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

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

[16] Michael Heilman and Noah A Smith. 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*, pages 1011–1019. Association for Computational Linguistics, 2010.

[17] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

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

[19] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[20] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[21] Leslie Lamport. *LATEX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[22] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[23] Christopher D Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International conference on intelligent text processing and computational linguistics*, pages 171–189. Springer, 2011.

[24] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736, 2016.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[26] George Miller, Christiane Fellbaum, Judy Kegl, and Katherine Miller. Wordnet: An electronic lexical reference system based on theories of lexical memory. *Revue quebecoise de linguistique*, 17(2):181–212, 1988.

[27] Yassine Mrabet, Halil Kilicoglu, and Dina Demner-Fushman. TextFlow: A text similarity measure based on continuous sequences. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 763–772, 2017.

[28] Yifan Peng and Zhiyong Lu. Deep learning for extracting protein-protein interactions from biomedical literature. *arXiv preprint arXiv:1706.01556*, 2017.

[29] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*, 2011.

[30] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Mapping dependencies trees: An application to question answering. In *Proceedings of AI&Math 2004*, pages 1–10, 2004.

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

[32] Jinfeng Rao, Hua He, and Jimmy Lin. Experiments with convolutional neural network models for answer selection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1217–1220. ACM, 2017.

[33] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE, 1998.

[34] Sebastian Schuster and Christopher D Manning. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *LREC*, pages 23–28. Portorož, Slovenia, 2016.

[35] Royal Sequiera, Gaurav Baruah, Zhucheng Tu, Salman Mohammed, Jinfeng Rao, Haotian Zhang, and Jimmy Lin. Exploring the effectiveness of convolutional neural networks for answer selection in end-to-end question answering. *arXiv preprint arXiv:1707.07804*, 2017.

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

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

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

[39] Libin Shen, Giorgio Satta, and Aravind Joshi. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 760–767, 2007.

[40] David A Smith and Jason Eisner. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 23–30. Association for Computational Linguistics, 2006.

[41] Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. Learning to rank answers on large online QA collections. *Proceedings of ACL-08: HLT*, pages 719–727, 2008.

[42] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[43] Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. Learning to rank question answer pairs with holographic dual LSTM architecture. *arXiv preprint arXiv:1707.06372*, 2017.

[44] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 583–591. ACM, 2018.

[45] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Multi-cast attention networks for retrieval-based question answering and response prediction. *arXiv preprint arXiv:1806.00778*, 2018.

[46] Stefanie Tellex, Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003)*, pages 41–47, Toronto, Canada, 2003.

[47] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[48] Quan Hung Tran, Tuan Lai, Gholamreza Haffari, Ingrid Zukerman, Trung Bui, and Hung Bui. The context-dependent additive recurrent neural net. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1274–1283, 2018.

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

[50] Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1288–1297, 2016.

[51] Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 707–712, 2015.

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

[53] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[54] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

[55] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[56] Zhiguo Wang and Abraham Ittycheriah. FAQ-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*, 2015.

[57] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*, 2016.

[58] Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. ANMM: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 287–296. ACM, 2016.

[59] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, 2015.

[60] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 858–867, 2013.

[61] Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1744–1753, 2013.

[62] Wenpeng Yin and Hinrich Schütze. Task-specific attentive pooling of phrase alignments contributes to sentence matching. *arXiv preprint arXiv:1701.02149*, 2017.

[63] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*, 2015.

[64] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.