THE UNIVERSITY *of* LIVERPOOL

Sentence Matching for Question Answering with Neural Networks

Thesis submitted in accordance with the
requirements of the University of Liverpool
for the degree of Doctor of Philosophy

in

Electrical Engineering and Electronics

by

Jinmeng Wu, B.Sc.(Eng.)

September 2018

Sentence Matching for Question Answering with Neural Networks

by

Jinmeng Wu

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors Prof. Yannis Goulermas, Dr. Jeyan Thiyagalingam and director Dr. Tingting Mu , for their patient, encouragement, and professional instructions throughout this research. I could never progress toward the ability to reach to this stage without their consistent and illuminating support.

I would also like to extend my appreciation to my annual progress committee members: Dr. Jiafeng Zhou and Dr. Saqib Khursheed, for their invaluable advice during my research. And my thanks also go to the Department of Electrical Engineering and Electronics at the University of Liverpool, for providing the research facilities that made it possible for me to carry out this research.

I would like to thank my colleagues, Austin Brockmeier, Andrew Jones, Liyun Gong, Long Yan, Yanbin Hao, Xingjian Gao and Yu Wu. For their wonderful camaraderie I enjoyed and many helpful discussions that challenged me to think deeper and wider about my research.

I offer my regards and blessings to all my friends in Liverpool for the experiences and pleasures we have had together. Special thanks to Zhihao Tian, Peng Yin, Chen Xu, and Wenzhang Zhang for all their wonderful camaraderie and suggestions during my time at the University of Liverpool.

Last but not least, I would like to extend my deepest gratitude to my family, especially to my parents AiGuo Wu and GuoMei Xu, and my brother JinRui Wu. Their unconditional love, belief, and support always encourage me and give me strength.

Abstract

Natural Language Processing is an important area of artificial intelligence concerned with the interactions between computers and human language. Semantic matching requires accurately modeling the relevance between two portions of text and is widely used in various natural language processing tasks, such as paraphrase identification, machine translation and question answering. In the past years, several works have continually progressed towards improving the ability to capture and analyse the text matching information (e.g. lexical, syntactic, etc.) through application of various techniques. In this thesis, we examine the problem of developing efficient and reliable semantic matching methods for question answering and exploring the effects of the different deep learning algorithms. We split our work into three pieces, where each part refers to a different sentence matching structure.

In the first part of thesis, we propose a deep semantic similarity model to learn a distributed similarity representation for sentences pairs. Text matching can use lexical information without any consideration of semantics. Semantic similarities can be determined from human-curated knowledge, but such knowledge may not be available in every language. The novelty of the proposed architecture lies in an abstract representation of the pairwise similarities created by deep denoising stacked auto-encoders. Model training is accomplished through a greedy layer-wise training scheme, that incorporates both supervised and unsupervised learning. The proposed model is experimentally compared to state-of-the-art approaches on two different dataset types: the TREC library and the Yahoo! community question datasets. The experimental results show the proposed model outperforming other approaches.

In the second part of thesis, we focus on designing a new question-answer matching model, built upon a cross-sentence, context-aware, bi-directional long short-term memory architecture. Semantic matching between question and answer sentences involves recognizing whether a candidate's response is relevant to a particular input question. Given the fact that semantic matching does not examine a question or an answer individually, context information outside the sentence should be considered with equal emphasis to the within-sentence syntactic context. An interactive attention mechanism is proposed which automatically select salient positional answer representations, that contribute more significantly towards the relevance of an answer to a given question. In the experiments, the proposed method is compared with the existing models, using four public community datasets. The results state that the proposed model is very competitive. In particular, it offers 1.0%-2.2% improvement over the best performing model for three out of four datasets, while for the remaining one performance is around 0.5% of the best performer.

In the last part of this thesis, a much more complex deep memory network structure is considered. In particular, we aim at developing a novel memory network for storing and reading the relevant question answer pairs from internal corpus. Traditionally, to provide more related sentence information, the external resources (e.g. knowledge base, related documents) are widely used in question answering leading to large computational costs. Thus, our goal in this work is to build a reliable model that can utilize the input corpus to build the memory network without the knowledge-based resources. In the experiments, our proposed method indeed improves the matching performance on three library/community question answering datasets, when compared with those methods relying on memory structures with document resources, it achieves better performances compared with the state-of-arts.

Declaration

The author hereby declares that this thesis is a record of work carried out in the Department of Electrical Engineering and Electronics at the University of Liverpool during the period from October 2014 to September 2018. The thesis is original in content except where otherwise indicated.

Contents

List of Figures

List of Tables

Abbreviations

IR Information Retrieval. 1, 8, 38, 105

NLP Natural Language Processing. 1, 6, 8, 25

QA Question Answering. 3, 4, 6, 8, 11, 13, 25, 38, 104, 135, 136

MRC Machine reading comprehension. 4, 8, 11, 25, 35, 45, 104, 136

cQA Community Question Answering. 4, 5, 8, 11, 13, 35, 38, 48, 136

AI Artificial Intelligence. 6, 8

NMT Neural Machine Translation. 7, 25, 37

CNN Convolution neural network. 7, 9, 22, 35

RNN Recurrent Neural Network. 7, 9, 10, 25, 26, 35

LSTM Long-Short Term Memory. 7, 10, 136, 138

GRU Gated Recurrent Unit. 7, 42

BRNN Bi-directional recurrent neural networks. 7, 10

AE Auto-encoder. 16, 20, 22, 35

DAEs Deep Auto-encoders. 16

PCA Principal Component Analysis. 17

DAE Denoising Auto-encoder. 19, 35, 69, 70, 136

SAE Stacked Auto-encoders. 21, 35, 58

ANNs Artificial Neural Networks. 25

RNN LM RNN-based language model. 26

MLP Multi-Layer Perceptron. 26

NTM Neural Turing machine. 30, 33, 35, 112

CNNs Convolutional Neural Networks. 37

BOW Bag-Of-Words. 39

LSA Latent Semantic Analysis. 39

TED Tree Edit Distance. 39

RBM Restricted Boltzmann Machine. 40

DMN Dynamic Memory network. 45

DSSM Deep Semantic Similarity Model. 49, 50, 61

SDA Sparse Denoising Auto-encoder. 58

MRR Mean Reciprocal Rank. 62, 136

MAP Mean Average Precision. 62, 136

VQA Visual Question Answering. 137

GAN Generative Adversarial Nets. 138

Chapter 1

Introduction

1.1 Semantic Sentence Matching

Semantic matching represents the relationship between objects in an ontology technique to identify information which is semantically related. The object includes but is not limited to image, sentence, topic, etc. Semantic matching as a measurement operation that is used in various applications of fields, such as Information Retrieval (IR) [3], data integration, Natural Language Processing (NLP) [2, 4, 5], peer to peer networks, conversational agents [6, 7], ontology merging. Semantic matching demands machine has understanding, reasoning, decision making and natural language generation in order to replicate or simulate the behavior of human mind in the real world. In this research, we focus on researching the semantic relationship between sentences as referred to semantic sentence matching.

Semantic sentence matching as a changeling problem in NLP, it aims to understand words and sentences meaning, and measure the relevance between word pairs by the similarity formulation. For instance, given the two sentences as an input pair is shown in Figure 1.1, the semantic matching is an operator which measures those words in the two sentences which semantically correspond to one another. In Text 1, the word "car" is semantically equivalent to "automobile" in the Text 2, since they are in English corpus. Moreover, the word "famous" in Text 1 shares the same meaning with the word "popular" in Text 2. These information of synonymous word pairs is

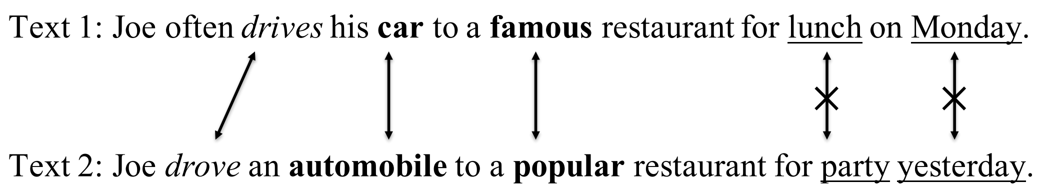

Text 1: Joe often *drives* his **car** to a **famous** restaurant for lunch on Monday.
Text 2: Joe *drove* an **automobile** to a **popular** restaurant for party yesterday.

Figure 1.1: An example of matching sentence pair. Successfully matched word pairs are highlighted in bold. Mismatched word pairs are marked by underline.

optional collected from the external linguistic resources [8], e.g. WordNet, Wikipedia, Freebase. However, there exist several word pairs are mismatched between the two sentences, for example, the word "lunch" has a distinguishable entity meaning, differs from the word "party". Similarly, the mismatched words "Monday" and "yesterday" indicate the distinct dates, although these words belong to the same category about time. In particular, the words "drive" and its paradigmatic type "drove" represent the same semantic meaning but in grammatical tenses. From the example, we analyzed the semantic sentence matching based on word-level. When viewed from a sentence-based perspective, the two sentences present different semantic meanings with a low similarity based on the expressions of the entire sentences. Therefore, the semantic sentence matching relies on word or sentence-level is a challenging problem to be explored.

Actually, the semantic matching method has been proposed to solve the semantic compatibility problem as a valid solution, is referred as the handling linguistic diversity. For example, the mutual intelligibility of multiple languages appear different meanings and cultures, the use of different terminology has been a serious problem. Recent the machine translation works may solve the semantic heterogeneity problem by converting the language on time. Following with the advent of the internet and the consequential information explosion, the richness of the language is continually increased, some novel words and phrases occur on the community forums on the web, are called network-based language. Since the increase of language, IR includes question answering and information integration from a variety of sources to be challenging tasks. The accurately semantic sentence matching becomes more important and challenging for solving the semantic compatibility problem.

The traditional approaches of semantic sentence matching methods detect the similarity between sentences, e.g. lexical matching, knowledge-based and IR. The lexical matching [9] method addresses the characters similarity computation without considering the synonymous problem between word pairs. The knowledge-based method [8, 10] extracts semantic information from the external resource bases for the neural network and dependency tree with syntactic structure. Using these traditional methods, it is hard to capture the semantic relationship between sentences, and identify the similarity relevance based on word or sentence-level. Recently, a series of deep neural networks [2, 11] have been proposed to provide a solution for the challenging problem, since the unique architecture and properties of these deep neural networks.

1.2 Question Answering Tasks

Question Q_1

- Where was the Joe?

Candidate Answers: A_1 and A_2

- A_1: When Joe comes back from school, he will go to the kitchen first, then go to the bathroom, I guess he should be there.
- A_2: Joe often forgets to take his lunch to school. Fred, can you remind him next time?

Figure 1.2: Example scenario for QA.

Question Answering (QA) is an important task within the fields of information retrieval and NLP, which is concerned with building systems that automatically answer questions posed by human beings in natural language. The example matching question and answer (Q-A) pair is shown as Figure 1.2. It can be seen that question and answer sentences are tidy and clear in the structured QA dataset, they share the mutual information so that it is easy to identify the A_1 is the correct answer for question Q_1. In general, the structured QA dataset, also called library QA is manually collected from the regular websites, e.g. TREC [12] and WikiQA [13].

QA is widely derived in a variety of tasks in NLP, for instance, dialogue system [6, 7], Machine reading comprehension (MRC) [14, 15], Community Question Answering (cQA) [11, 16], and IR engine on website/mobile [17]. QA tasks are characterized as challenging research topic in computer science, despite language being one of the easiest things for humans to learn. Since human language is rarely precise and plainly spoken, QA tasks becomes difficult for machines to deal with because of the ambiguity of language. To simulate human language, machine is not only required to understand the meanings of words, but also to comprehend the various concepts of phrases or sentences composed of words combinations. Hence it is crucial to build sentence and word representations capable of capturing the relationship and semantic information between question and answer.

Question Q_1 and Q_2

- Q_1: Where was Joe before the office?
- Q_2: Where is Joe?

Supported Contexts: C_1 and C_2

- C_1: Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
- C_2: Joe traveled to the office. Joe left the milk. Joe went to the bathroom.

Target Answers: A_1 and A_2

- A_1: kitchen
- A_2: bathroom

Figure 1.3: Example "Joe's story" scenario for QA with inference.

Recently popular MRC task has been proposed used in various deep neural networks. MRC [18] is a challenging matching problem which requires an ability to understand the context meaning, consisting of a large number of sentences. The given input of MRC is a triple type that consists of question, context and labeled answer, as shown in Figure 1.3. From the figure, two input triples are given for generating answer spans of the contexts. Unlike Q-A pair matching in Figure 1.2, the contexts C_1 and C_1 perform a key impact for supporting its corresponding questions as the inferences. With respect to the logical question, the two contexts can be combined to generate an answer, the preceding context information is helpful for reasoning. For instance, when the question in the second triple changes from 'Where is Joe?' to 'Who picked up the milk?', the supported information 'Joe picked up the milk' occurs in context C_1, not C_2. Therefore, capturing a certain amount of related sentences in the context may benefit the question and context (Q-C) pair matching.

Question Q_1

- Hi, Fred!! emmm...Do you know where is Joe going???

Candidate Answers: A_1 and A_2

- A_1: No, emmm..I guess he may back to home, why not contact with him??? LOL...
- A_2: wow, He left his card on park, can you please return it to him???

Figure 1.4: Example scenario for sentence matching in cQA.

cQA has seen a spectacular increase in popularity in the recent past. With the advent of web and popularity of computer, a number of cQA datasets are produced collected from web forums, such as Yahoo! [19], Quora [20], and Stack Overflow [21]. A large amount of users use these web forums to obtain answers for the posted questions. Meanwhile, users are allowed to post their question on these community forums, then waiting it answered by several experts, who are also on-line users. The answers can help the other users by posting comments under the answers. The increases of Q-A pairs in such web platforms enables the ability to accomplish QA tasks, e.g. sentence retrieval, answer selection and relevant question searching.

The sentence matching example of cQA is illustrated in Figure 1.4. Unlike the structured Q-A pair, two semantic components of cQA may lead to the challenging problem. The Q-A pairs in cQA belong to open-domain without much access limitation, and the non-factoid pair causes amount of disordered and misunderstood words in sentences, such as the slang "emmm..." and "LoL" in the figure. These unstable pairs in cQA may result in a poor matching quality. An alternative problem is that, the answer sentence in cQA is generally composed by a large number of words, which lead to a long answer sentence, especially compared with short question. Thus, the information of Q-A pair is not equivalent and insufficient for semantic matching.

1.3 Deep Learning Methods

Deep learning is a novel branch within machine learning system. A series of research works have been proposed in recent years to learn algorithms towards various deep architectures with multiple levels of representations. Theoretical arguments in [22] claimed deep learning architectures are necessary to efficiently represent the kind of high-level abstractions through learning the family of functions in Artificial Intelligence (AI). The successful research work of deep learning has already attracted much attention from academic field and commercial industries.

The origin of deep learning goes back to the discovery of biology, neocortex is a key founding for deep architecture research based on mammal brain that has cognitive capacity. It allows sensory signals to pass through a complex hierarchy and observations are represented regularity follows with learning [23]. Abstractions at multiple layers represent given input from the lowest to the highest level and each layer corresponds to the relevant cortex area. Therefore, this discovery leads to the appearance of deep structure in machine learning field [23]. The innovation of deep architecture performance with an unsupervised pre-training approach for training deep belief networks [24]. The pre-training approach focuses on training each layer in the architecture with unsupervised learning algorithm followed by global fine-tuning. With respect to supervised training, this pre-training method offers a better generalization, regularization, capacity control, and low variance.

Typically, deep learning includes two main properties denote depth and flexibility, they employ the possibility of features at each level by creating the representation in a layer-wise approach. Depth represents the numbers of layers in the architecture, which has at least two training layers of distributed representations. Flexibility property permits the deep learning structure contains a series of prior information and achieves a state-of-art performance in multiple tasks, e.g. speech recognition [25], computer vision [26, 27], NLP [28, 29].

Recently, a variety of deep neural networks have been developed to QA task based on presenting word to a distributed representation type [30,31]. Deep learning models have shown effectiveness to characterize latent relationship between words, producing distributed embedding representation for words [5, 29] and sentences [32, 33], since the properties of deep learning model allow the structure contains a series of prior information and learns multiple tasks in a meanwhile [30]. Deep neural networks have been widely applied to various NLP tasks such as Neural Machine Translation (NMT) [4, 34], paraphrase identification [35], and summarization [36].

Convolution neural network (CNN) [37, 38] is a common semantic sentence matching method, aiming to integrate the word patterns into a representation that expresses the semantics of a given sentence. In general, CNN uses a convolution operator to reduce the dimension of distributed sentence representation from a matrix to vector type, this step simplifies similarity matching so the similarity score is computed based on two vectors. However, there are two main drawbacks of these models: the input information such as the cumulative meanings of words, the relationship between word pairs and the order of words would be loss and unclear. Sentences representations are separately learned by two CNNs in [33], which increases the number of variables need to be tuned and costs a larger model computation.

Recurrent Neural Network (RNN)-based language models [39, 40] recently perform well on retaining and stacking the key information of words spread throughout a sequence using variant architectures such as the Long-Short Term Memory (LSTM) [41] and Gated Recurrent Unit (GRU) [42], since the long-term monomeric property. Bi-directional recurrent neural networks (BRNN) [43] has been proposed to learn the sentence representation through taking into account information from both the preceding and following words, supported by an alignment model to produce a vector to aggregate word information into sentence. As for QA tasks, BRNN is used to learn similarity based on word-level considering with adjacent words in the sentence. Since BRNN has the long-term mnemonic property, we study an interactive attention model that uses BRNN to generate the question representation by involving sentence's context information. This context information is automatically created by question or answer representation and represented by a vector, which instead of external resources provided by manual feature engineering in previous works [8, 44].

1.4 Motivation and Main Contributions

1.4.1 Motivation

In this thesis, the goal of the research work is to design the various deep neural networks to learn the high-level semantic text matching for QA tasks, including answer selection, cQA, and MRC. The semantic text matching task is a challenging problem in NLP, it demands machine has understanding, reasoning, decision making , and natural language generation in order to replicate or simulate the behavior of the human mind in the real world. Nevertheless, the traditional methods based on statistical language models are not able to capture the meaningful relationship between words by using shallow learning. The deep neural network is efficient to learn the meaning of word according to its deep layered architecture and flexibly designed algorithms. In the current, text matching task is widely applied in modern industries, for instance, automatic conversation on mobile, IR on the website, and image description.

AI is a popular field focuses on creating novel technologies allow computers and machines to simulate minds of human being in an intelligent manner. The general fields of simulating intelligence include machine learning [45], machine visual [46] and NLP [47]. As one of central tasks of AI, NLP explores how machines are able to understand and manipulate natural languages such as sentence or speech, to accomplish multiple real-world applications. Several popular tasks of NLP include automatic sentence summarization [36], sentiment analysis [48], and machine translation [34]. QA is a computer science area within the fields of information retrieval and NLP, which is concerned with building systems that automatically answer questions posed by human beings in a natural language. QA is widely used in variety of applications, for instance, dialogue system [6, 49], social network [50] and search engine on website/mobile [17].

The traditional approaches in QA typically adopted a statistical method that is to use lexical matching to count the number of matching words or stems [9], however, it fails to understand the meanings of words and capture the correlation between words, which leads less relevance, namely semantic information between question and answer. Although feature-based models have been proposed in [51–53] by use semantic features to improve the similarity metric, their feature engineering stage can be expensive. Moreover, these approaches cannot be adapted to match language for which there is no external semantic resource available, as previously noted [32]. Replacing the manual feature engineering with a neural language model, various recent works have been developed to compute semantic similarity between sentence by learning distributed representations of words via neural networks [30, 31, 33]. The strategy of the neural network is to characterize each word with a vector with less dimensionality. Popular ways for generating a word vector embedding include bag-of-word representation based on the contextual words around the targeted word [54], latent semantic analysis [28], and distributed embedding representation generated by a probabilistic neural language model [55].

Above neural networks provide effective ways to represent a word in a certain amount of corpus, nevertheless, machine computes the similarity over phrases/sentences representation level is still a precise issue for QA. In recent years, deep learning models have shown effectiveness to characterize the latent relationship between words, returning distributed embedding representation for words [5, 29] and sentences [32, 33], due to the two main properties of deep learning models: depth and flexibility that allows the structure of deep architecture contains a series of prior information and learns multiple tasks in a meanwhile [30].

Recently, CNN [37] and RNN-based models [39] gain the state-of-art performance on semantic sentence matching. In the previous subsection 1.1, we mentioned that learning semantic sentence matching based on word-level and sentence-level provide a different similarity between sentences, which is a challenging problem for QA. Most CNN [33, 35] and RNN works likely to learn the semantic relationship between sentence representations. More precisely, CNN demonstrates the similarity between Q-A pair by integrating the word patterns into a combined representation that expresses the semantics of a given sentence. These models consider the complexity of CNN, converting the distributed sentence embedding representation from matrix to the vector at the early step, which simplifies similarity matching procedure through computing score over two vectors. However, there are two main drawbacks of these models: the input information such as the cumulative meanings of words, the relationship between word pairs and the order of words would be lost and unclear. The sentences representations are separately learned by two CNNs in [33], which increases the number of variables need to be tuned so that larger model computation would cost. In Chapter 4, we propose the deep matching model based word-level similarities to solve these problems. The proposed model represents each word positional similarities by using a distributed similarity matrix, which captures word pair importance factor between a pair of the sentence.

Other than treating a sentence representation as a unordered words, RNN retains and stacks the key information of words spread throughout a sequence using architectures similar to LSTM [41]. BRNN [34] has been proposed to strengthen the sentence representation by taking into account information from both the preceding and following words, and can be applied with the support of an alignment model to produce a vector for aggregating word information in a sentence for translation task. As for QA tasks, previous work [56] computed a similarity matrix between the answer and question sentences, utilizing their multiple positional representations learned by a BRNN. Although this work has been achieved a satisfying matching result, BRNN learns the similarity based on sentence-level in an individual learning approach, which lacks the interaction between question and answer. In Chapter 5, with respect to the long-term memory property of LSTM , we propose the interactive attention model that uses bi-directional LSTM to generate the sentence representation by involving the contextual and interactive information using pair-wise attention mechanisms: co-attention and similarity-attention. This contextual information is automatically created by combing word-positional representations, which instead of external resources provided by manual feature engineering in previous works [51–53].

A knowledge-based resource [44, 57] has been recently adopted in deep neural networks for semantic sentence matching. Despite the external database can supply the relevant information to improve matching, it is hard to master the semantic relationship between a pair of sentence. Moreover, it costs an expensive computation by collecting the database from a web or large corpus. Instead of external resource with a neural language model, memory network based models [58,59], a specific type of recurrent networks with dynamic and addressable memory mechanism, have been recently proposed to store the input content information for MRC task. The content consists of multiple sentences has enriched information to be stored, whereas for pure QA matching also called answer selection, the question and answer has a few words, which are not able to be utilized by memory. In Chapter 6, we design the memory network to store the relevant questions and answers for each Q-A pair as possible. Besides, the relevant Q-A pairs are collected from internal dataset corpus related to the input Q-A pair. Except for answer selection task, the proposed model is also able to accomplish MRC task by using attentive memory as a matrix to store content, input, and relationship between Q-C pair, with a dynamic storage process. The large capacity of the proposed memory overcomes the other memory networks [59, 60] that suffer from the storage limitation problem. To summarize, the key hypotheses of the thesis are listed as follows:

- The semantic deep neural networks are proposed to maintain a better performance than the existing methods for question answering task.
- The set of word-level similarities is sufficient for the semantic matching task. Essentially, the proposed neural network is able to learn how to distill a set of pairwise similarities to a single relevance measure.
- Interactive learning of the question and answer representations is beneficial to semantic sentence matching by considering positional words information.
- A multi-dimensional memory network has the ability to store relevant sentence pairs, then improve sentence matching by releasing these sentence pairs.

1.4.2 Contribution

In this thesis, we design the deep neural networks for handling the semantic matching problem in three types of QA tasks – answer selection with structured QA and cQA, and MRC. Accordingly, three new models are adaptive for three types of tasks and these models are briefly introduced as follows:

- **Greedy Word-level Semantic Similarities**: In Chapter 4, we examine the problem of semantic information loss caused by compressing the sentence representation at an early stage for semantic structured QA and cQA matching. In the proposed work, we first learn the distributed interaction representation of sentence pair directly, and gradually compress the distributed similarity representation, instead of learning individual sentence representation. To investigate the effect of deep learning network to semantic sentence matching problem, we compare the performance between two learning presentation approaches, the experimental result show that the proposed work learns semantic similarity based on word-level outperforms the one on sentence-level. Considering the amount of trainable parameters in complex deep neural networks, we employ a pair-wise training method to train the proposed model. The training approach is divided into multiple steps is individually trained by the specific loss objectives. At last, the model is supported by an effective greedy layer-wise training strategy to fine-tune the entire model, and this decomposable training approach is experimentally shown to outperform the previous models.

- **Context-aware Neural Network**: In Chapter 5, we consider strengthening the sentence representation based on a cross-sentence context-aware bi-directional LSTM architecture for examining structured QA and cQA matching. The new pair-wise attention mechanisms are designed to enrich the generative sentence representation: the co-attention mechanism takes the sentence content, interactive attention involves the semantic information of word pairs between question and answer. The quantity context information jump is proposed by considering informativeness of multi-positional words, it helps to improve the computation of attention mechanism. Besides, the proposed model pre-trains the sentence representation based on Bi-directional LSTM in unsupervised training. The generative sentence representation reduces the computation without sacrificing the representation quality.

- **Attentive Memory Network**: In Chapter 6, we deal with the storage limitation problem of memory for semantic QA and MRC matching. The proposed work employs the memory as a matrix type to store all semantic similarities of word pairs to improve the content representation in memory, which extracts the semantic information for supporting the semantic matching. In proposed work, the new memory refinement is designed to store the contextual information of sentences in separately two steps at one iteration, with simplify recording the information using row and column-based vectors. Attention memory network improves the interactive content in the memory, it is helpful for generating multiple words with weakly unsupervised training. The proposed memory network is capable of completing QA and MRC tasks by defining a different matching operation and loss functions in the output layer.

1.5 Thesis Outline and Related Publications

We organize the thesis outline as follows:

CHAPTER 2 introduces the preliminaries by introducing various generic deep learning methods for semantic matching by learning the input representation. The chosen methods studied in this chapter represent the basic concepts and methodologies in this area. In fact, most existing works in deep learning are largely based on recognizing some of these ideas and create novel extensions to them.

CHAPTER 3 reviews the related works connect to semantic sentence matching. The related works present the general conceptions and various architectures in different period. The related works provide a better understanding of this field and the proposed works in this thesis. In addition, it describes the ideas of state-of-art models compared with the proposed model.

CHAPTER 4 introduces the deep semantic similarity model with a pair-wise training approach in structured QA and cQA. It is based on the hypothesis that learning interactive similarity based on word-level. The proposed model structure consists of three parts: distributed bilinear similarity computation, the stacked AE with unsupervised training learns the high-level deep similarity representation, an overall similarity score computation and matching prediction. The pair-wise training approach based on a supervised ranking score is used to optimize the variables in parts of the proposed model. In analysis of experiments, the proposed model achieves better with this training approach. Performance comparison with the state-of-art models, its superiority demonstrates in various performance comparisons.

CHAPTER 5 addresses the interactive learning of sentence representation. We propose a cross-sentence context-aware bi-directional LSTM architecture. This model contains interaction-based and sentence-based attention mechanisms for answer representation learning, and augment the proposed approach to consider the relationship between adjacent words. The context-aware model with pair-wise attention mechanisms offers the best performance among all the comparing methods.

CHAPTER 6 studies multi-dimensional memory network with attention mechanisms. The proposed attentive memory network is to efficiently store and extract the semantic information related to question and answer or context pairs as an internal knowledge resource for the memory network. Multiple performance comparisons with the state-of-art approaches also demonstrate the superiority of the proposed model.

CHAPTER 7 concludes the whole thesis. We underline the contribution of this thesis, recapture the key ideas, and propose several potential directions for future research.

The publications produced from this research work are listed as follows:

- J. Wu, T. Mu, J.Thiyagalingam and J. Y. Goulermas, "Building Interactive Sentence-aware Representation based on Generative Language Model for Question Answering", Neurocomputing, Elsevier, minor revision.
- J. Wu, T. Mu, J.Thiyagalingam, and J. Y. Goulermas, "Multi-dimensional Memory Model for Question Answering and Machine Comprehension", IEEE

Transactions on Knowledge and Data Engineering, to be submitted.

- J. Wu, T. Mu, J.Thiyagalingam, and J. Y. Goulermas, "Deep Semantic Answer Matching by Greedy Word-level Similarity", IEEE Transactions on Knowledge and Data Engineering, to be submitted.

Chapter 2

Deep Learning Methods for Sentence Matching

In this chapter, we explore the deep learning methods since they are building models for developing the more complicated methods or to handle more complex semantic matching problem. Almost deep learning methods for semantic sentence matching are implicitly developed based on the designs of these networks. For instance, the proposed methods expressed in Chapters 4- 6 are partly based on the theoretical conceptions and algorithms related to generic neural networks. Therefore, it is vital to investigate these neural networks in great depth.

2.1 Auto-encoders

2.1.1 Theoretical Foundations

Auto-encoder (AE) is a symmetrical model [31] in neural networks, that learns features through an unsupervised leaning algorithm that computes back-propagation, assuming that the target labels are equivalent to the inputs. The basic structure of AE illustrated in Figure 2.1 is composed of a single hidden layer, while an AE contains the multiple hidden layers is called Deep Auto-encoders (DAEs) that is commonly applied in paraphrase detection [31] and the document analysis [61, 62]. Regardless of AE or DAEs, the two main components of these models are the

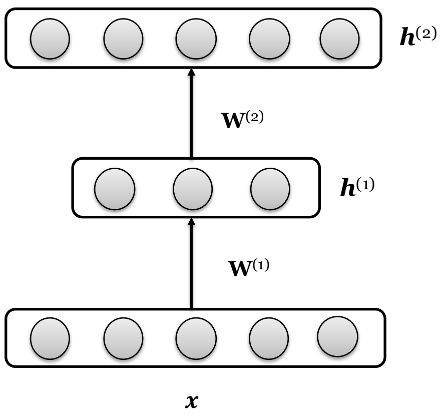


Figure 2.1: An illustration of AE architecture.

encoder and the decoder. The encoder is responsible for encoding the input $\mathbf{x}$ into the mapped representation, whereas the decoder aims to reconstruct this intermediate representation to the input.

Consider an AE with d units denoted as the input vector $\boldsymbol{x} = [x_1, ..., x_d]^T$, where x_i (0<i<d) is defined as a scalar value in the input vector. Here, we study a simple case of using a single hidden layer with k units. Let $\mathbf{W}^{(1)} = [w_{ij}^{(1)}]$ and $\mathbf{b}^{(1)} = [b_j^{(1)}]$ denote the parameters associated with the connection between unit i in input layer and unit j in hidden layer. The encoding formulation under AE is computed by

$$\boldsymbol{h}^{(1)} = f\left((\mathbf{W}^{(1)})^{\mathrm{T}}\boldsymbol{x} + \mathbf{b}^{(1)}\right), \tag{2.1.1}$$

with

$$(\mathbf{W}^{(1)})^{\mathrm{T}}\boldsymbol{x} = \sum_{i=1}^{d} w_{ij}^{(1)} x_i, \tag{2.1.2}$$

where the weight w_{ij} and bias b_j parameters are used to calculate the weighted sum of input to unit j in hidden layer. Here, we use a non-linear activation function $f(\cdot)$ of hidden layer, such as sigmoid function [63], that adapts to binary input by mapping the output value to the interval (0,1). The hyperbolic tangent [64] and rectified linear unit [65], that map the output to a certain domain are commonly chosen in the other cases. In particular, if activation is a linear function, a number of hidden units project the input in the span of the first principal components of data, so that the AE is similar to the Principal Component Analysis (PCA) [66]. Since the non-linear function applied to the AE, the encoded feature $\boldsymbol{h}^{(1)} = [h_j^{(1)}]$ in the hidden layer has the ability to obtain the main factors of variation from given input data.

In decoder, the encoded features are mapped to the reconstruction from hidden unit j to unit i in the output layer. Let the decoding formulation is defined as

$$\boldsymbol{h}^{(2)} = f\left((\mathbf{W}^{(2)})^{\mathrm{T}}\boldsymbol{h}^{(1)} + \mathrm{b}^{(2)}\right), \tag{2.1.3}$$

with

$$(\mathbf{W}^{(2)})^{\mathrm{T}}\boldsymbol{h}^{(1)} = \sum_{j=1}^{k} w_{ji}^{(2)} h_j^{(1)}, \tag{2.1.4}$$

where the weight matrix $\mathbf{W}^{(2)} = [w_{ji}^{(2)}]$ and bias vector $\mathrm{b}^{(2)} = [b_i^{(2)}]$ denote the parameters from hidden layer to output layer. Because the decoder aims to reconstruct the input, the decoded feature $\boldsymbol{h}^{(2)} = [h_i^{(2)}]$ has the same d-dimensional units as the input vector $\boldsymbol{x}$.

During the training procedure, AE is trained to minimize the re-constructive error between the output $\boldsymbol{h}^{(2)}$ at the decoding layer and the input $\boldsymbol{x}$ at the encoding layer.Consider a set of m training samples $\{\boldsymbol{x}_i\}_{i=1}^m$, with an overall parameter $\boldsymbol{\theta} = [\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}]$ in the l-th layer (l=1,2 for one hidden layer). The common choice for an arbitrary re-constructive error formulation is the mean square error over the input samples, the objective cost function is defined as

$$L(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} ||\boldsymbol{x}_i - \boldsymbol{h}_i^{(2)}||^2 + \frac{\lambda}{2}\sum_{l=1}^{2} ||\boldsymbol{W}^{(l)}||_2^2, \tag{2.1.5}$$

note the latter term of cost function $\frac{\lambda}{2}\sum_{l=1}^{2} ||\boldsymbol{W}^{(l)}||_2^2$ is called the regularization term, where $||\cdot||_2$ means Euclidean norm and $\boldsymbol{\lambda} > 0$ is the regularization parameter, also called weight decay term, which tends to decrease the magnitude of the weights and helps prevent over-fitting phenomenon. In the following, we exploit the variations of the AE in order to capture significant input information and enrich the representations, including the sparsity, denoising and multiple layer properties.

2.1.2 Sparse Auto-encoder

In general, an AE is used to map the input representation to a fix-sized hidden representation with dimensionality reduction. However, the compression of the input becomes difficult when the number of hidden units is large. In particular, a sparsity constraint on the hidden units makes it possible to vary the effective number of units. A sparse auto-encoder does not suffer from these issues: the computational complexity (of inferring the codes), the stability of the inferred codes, and the numerical stability and computational cost of computing gradients on the first layer in the context of the global fine-tuning of a deep architecture.

In sparse AE [67,68], a non-linear mapped over-fitting of the input vector $\boldsymbol{x}$ tends to employ the sparsity to the activation in hidden layer. The objective minimizes reconstructed error with a sparsity constraint, the overall cost function becomes

$$L_{sparse}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \beta \sum_{j=1}^{k} KL(\rho||\widehat{\rho}_j), \tag{2.1.6}$$

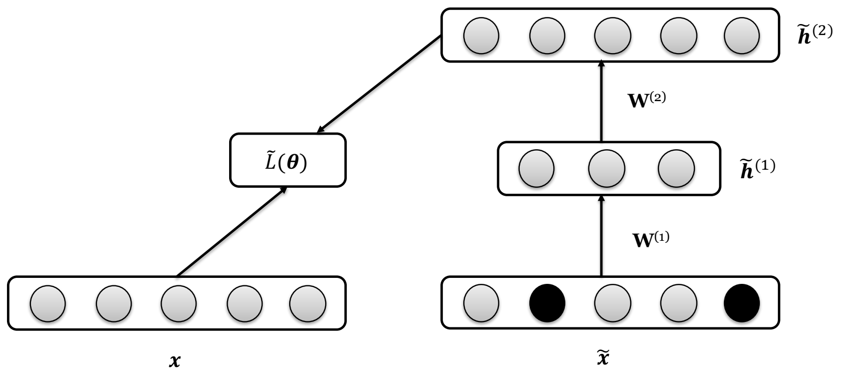
where $L(\boldsymbol{\theta})$ is the cost function defined in Eq.(2.1.5), and β is sparsity penalty weight. Notice $KL(\rho||\widehat{\rho}_j)$ is Kullback-Leibler (KL) divergence [69] between a Bernoulli random variable with a target activation ρ and the other type of averaged activation $\widehat{\rho}_j$ of hidden unit j. The penalty term of cost function based on KL divergence is given as:

$$\sum_{j=1}^{k} KL(\rho||\widehat{\rho}_j) = \sum_{j=1}^{k} \rho \log \frac{\rho}{\widehat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\widehat{\rho}_j}, \tag{2.1.7}$$

where ρ is the sparsity parameter whose a small value nears to zero. The averaged activation $\widehat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m} h_j^{(1)}||\boldsymbol{x}_i||_1$ of the j-th hidden unit is calculated over m training sets, where $||\boldsymbol{x}_i||_1 = \sum_j^d |x_{ij}|$, and the activation $h_j^{(1)}$ of hidden layer is computed from Eq.(2.1.1). In particluar, if $\rho = \widehat{\rho}_j$, $KL(\rho||\widehat{\rho}_j)$=0. Otherwise, when the value of $\widehat{\rho}_j$ is not equal to ρ, the penalty term becomes larger whatever the increase or decrease of $\widehat{\rho}_j$. This variation influences sparsity constraint that non-redundant over-fitting features are learned when ρ is small enough [70].

2.1.3 De-noising Auto-encoder

Denoising Auto-encoder (DAE) is a stochastic variant of a basic AE that focuses on reconstructing the original input from a partially corrupted transformation of the input. The structure of the DAE is shown in Figure 2.2. It learns a robust



$\widetilde{\boldsymbol{h}}^{(2)}$
$\mathbf{W}^{(2)}$
$\tilde{L}(\boldsymbol{\theta})$
$\widetilde{\boldsymbol{h}}^{(1)}$
$\mathbf{W}^{(1)}$
$\boldsymbol{x}$
$\widetilde{\boldsymbol{x}}$

Figure 2.2: An illustration of the DAE architecture. The two circles in black colour indicate the corrupted units in the input.

representation from a corrupted input that is able to restore the original undistorted input [71].

Intuitively, DAE tends to prevent the redundant information of the input whilst offsetting the influence of a *stochastically corruption* procedure is applied to the input of AE. The corrupted input is obtained through the stochastic corruption mapping: $f_R(\widetilde{\boldsymbol{x}}|\boldsymbol{x}) \to \widetilde{\boldsymbol{x}}$, where the f_R is the desturuction mapping function that has multiple approaches, e.g. additive isotropic Gaussian noise, salt-and-pepper noise and random noise [72]. Here, we consider the random noise that consists of randomly selecting a set of elements in the initial input to zero. Let the corrupted input $\widetilde{\boldsymbol{x}}$ instead of the original input $\boldsymbol{x}$, as the input mapped to AE, we obtain the encoded representation $\widetilde{\boldsymbol{h}}^{(1)}$ via the encoder formulation Eq.(2.1.1), then the encoded representation as input applied to the decoder formulation Eq.(2.1.3) from which we reconstruct the decoded representation $\widetilde{\boldsymbol{h}}^{(2)}$. Denote the final hidden representation $\widetilde{\boldsymbol{h}}^{(2)}$ is used to reconstruct the original input $\boldsymbol{x}$, the overall parameter $\boldsymbol{\theta} = [\mathbf{W}^{(l)}, \mathrm{b}^{(l)}]$ in the l-th layer (l=1,2). The cost function of DAE is expressed over the training samples:

$$\widetilde{L}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} ||\boldsymbol{x}_i - \widetilde{\boldsymbol{h}}_i^{(2)}||^2 + \frac{\lambda}{2}\sum_{l=1}^{2} ||\mathbf{W}^{(l)}||_2^2. \tag{2.1.8}$$

The strategy of DAE is to robustly extract a high-level representation whilst enriches the hidden features under the corrupted input. The main innovation of DAE is highly stable and roubust as the unsupervised pre-training for a deep neural network [72].

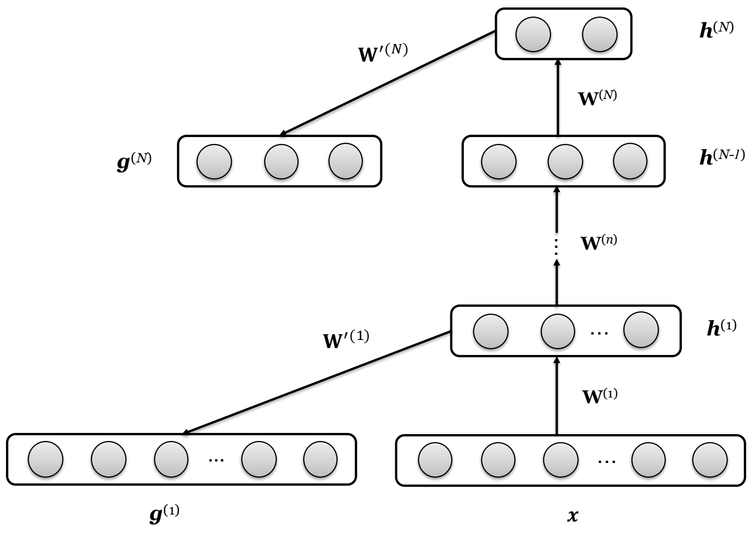


Figure 2.3: An illustration of the SAE architecture.

2.1.4 Stacked Auto-encorders

Stacked Auto-encoders (SAE) is a neural network with multiple layers consisting of a set of auto-encorders, where the output of each layer is connected to the input of the next layer [71], as shown in Figure 2.3. Formally, suppose a stacked auto-encorder with N auto-encorders to avoid confusion with hidden layer number L ($L \geq N$) in SAE, we adopt the same number of hidden layers and auto-encorders $L = N$ as a simple example shown in the figure. In the encoder of n-th AE, using the parameters from basic AE denote the weight $\boldsymbol{W}^{(n)} = [w_{ij}^{(n)}]$ and bias $\boldsymbol{b}^{(n)} = [b_j^{(n)}]$. The encoded feature is given by running the encoding step of each layer in the forward way

$$h_j^{(n)} = f(\sum_{i=1}^{d} w_{ij}^{(n)} h_i^{(n-1)} + b_j^{(n)}). \tag{2.1.9}$$

In particular, $\boldsymbol{h}^{(n-1)} = \boldsymbol{x}$ when $n = 1$. The encoder representation $\boldsymbol{h}^{(n)} = [h_j^{(n)}]$ of n-th hidden layer as the input passed to next layer $n+1$ in SAE network. Prior to the above step, we first use the encoder representation to reconstruct the input from the last layer, and learn the higher level hidden representation contains input information. The decoding formulation of the n-th AE is expressed by

$$g_i^{(n)} = f(\sum_{j=1}^{k} w'^{(n)}_{ji} h_j^{(n)} + b'^{(n)}_i), \tag{2.1.10}$$

where the decoding parameters weight $\mathbf{W}'^{(n)} = [w'^{(n)}_{ij}]$ and bias $\mathrm{b}'^{(n)} = [b'^{(n)}_j]$ connect the units between the encoding and decoding layer. Given the decoded representation $\boldsymbol{g}^{(n)}$, denote the overall encoding parameter $\boldsymbol{\theta}_n = [\mathbf{W}^{(n)}, \mathrm{b}^{(n)}]$ and decoding parameter $\boldsymbol{\theta}'_n = [\mathbf{W}'^{(n)}, \mathrm{b}'^{(n)}]$, the reconstruction minimization function is defined to restore the input from last layer $\boldsymbol{h}^{(n-1)}$

$$L'(\boldsymbol{\theta}_n, \boldsymbol{\theta}'_n) = \frac{1}{m}\sum_{i=1}^{m} ||\boldsymbol{h}_i^{(n-1)} - \boldsymbol{g}_i^{(n)}||^2 + \frac{\lambda}{2}\sum_{n=1}^{2} ||\mathbf{W}^{(n)}||_2^2. \qquad (2.1.11)$$

In SAE network, a set of auto-encorders are stacked in a greedy layer-wise type for pre-training the parameters of a deep network. The greedy layer-wise approach for pre-training a deep network by training each layer in turn. Finally, the pre-trained encoded representation is initialized as the input to the next layer in SAE.

2.2 Convolution Neural Network

2.2.1 Theoretical Foundations

CNN is a hierarchical neural network with alternating convolutional layers and sub-sampling layers as shown in Figure 2.4. The operation of CNN contains three main architecture properties: a local receptive field, shared weights and sub-sampling, to ensure invariance in the shift, scale and rotation for connecting the local receptive fields to neurons. The application of CNN shows a good performance on the visual tasks (e.g. pattern recognition) [73].

Suppose the input is a large size of 2-dimensional image (e.g. a 1000*1000 pixels). The AE network mentioned in the previous section can learn features through a fully connected network (spanning the entire image) but this is computationally expensive, and this method increases the number of parameters to be learned. For instance, assuming a number of 10^6 input pixels or variables, the order of 10^8 parameters must be learned to 100 features from the input. The computational speed of the feed-forward and back-propagation algorithm becomes much far slower with a large size input.

A locally connected network provides a solution due to its well-known advantage of being able to learn local features. Different from AE, CNN learns features through

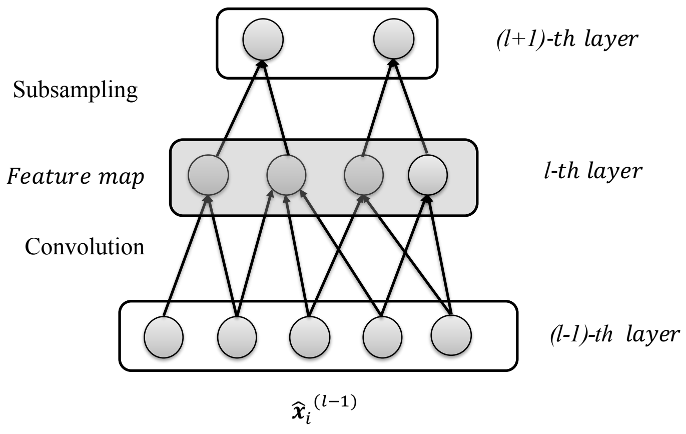


Figure 2.4: An illustration of the CNN architecture.

restricting the respective fields of hidden neurons to be a local connection. It allows the hidden unit to connect with a subset of input units. The idea of a locally-connected network refers to the the primary visual system (perceptron) in biology, and neurons in the visual cortex correspond to localized receptive fields with stimuli in certain locations [74].

Neurons extract features in contiguous region pixels with a different position in the local respective fields, such as comers and edges. Input units in the same located respective field are mapped to a plane called “feature map”. These units share a set of weights and bias during the mapping process. The output units in the feature map perform the same operation when working on different parts of the input, thereby constituting translation invariance. Next, we analyze the structure of a CNN that consists of convolutional layers and pooling layers.

2.2.2 Convolutional Layer

The convolution operation is the key component to CNN. The convolution kernel works as a local filter, it moves through the entire input and do the convolution operation on each local components. This process extracts feature representation of the local input, which works similar to common machine vision filters. The convolutional layer of CNN employs the convolution operation to the portions of input map using a kernel, and the convolved outputs denote states of the neuron are stored in a feature map. The layer's parameters consist of a set of learnable filters or kernels, which correspond to a local receptive field used to convolve the input maps or the feature maps in the previous layer. Generally, the output feature map combines convolutions with input maps by kernels. Denote a set of d-dimensional vectors of the input maps $\mathbf{X} = [\widehat{\boldsymbol{x}}_i^{(l-1)}]$, where $\widehat{\boldsymbol{x}}_i^{(l-1)}$ is the feature map for location i in the $(l-1)$-th layer. Here the connection weights $\boldsymbol{W}^{(l)} = [\boldsymbol{w}_j^{(l)}]$ and bias $\boldsymbol{b}^{(l)} = [b_j^{(l)}]$ are specified to be the j-th type of kernels, where the parameters $\boldsymbol{w}_j^{(l)}$ and $b_j^{(l)}$ connect the input map and output map by neurons in the feature map. The convolution output of feature map for location i in the l-th layer is defined as

$$\widehat{x}_{ij}^{(l)} = f(\boldsymbol{w}_j^{(l)}\widehat{\boldsymbol{x}}_i^{(l-1)} + b_j^{(l)}), \forall j \in [1, 2, \dots, K_l], \tag{2.2.1}$$

with its matrix form is defined by

$$\widehat{\boldsymbol{x}}_i^{l} = f(\boldsymbol{W}^{(l)}\widehat{\boldsymbol{x}}_i^{(l-1)} + \boldsymbol{b}^{(l)}), \tag{2.2.2}$$

where K_l is the number of kernels used for convolution at layer l, and total number of layers is set as L. In particular, $\widehat{\boldsymbol{x}}_i^{(0)}$ is a segment part of the primary input when l=1. Assuming the width size of sliding windows is set to k for a kernel, thus the convolved input map is scanned from the i-th position to $(i+k-1)$-th position across the entire input denotes by $\widehat{\boldsymbol{x}}_i^{(0)} = \boldsymbol{x}_{i:i+k-1} = [\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \dots, \boldsymbol{x}_{i+k-1}]^{\mathrm{T}}$, where $\widehat{\boldsymbol{x}}_i^{(0)}$ is the k-th dimensional vector. Since the convolution network has a robust shift invariance property, the same amount of change occurs in the feature map output as that in the input changes.

2.2.3 Subsampling Layer

The precise positions of the detected features are irrelevant in identifying the patterns after the convolution process. The property of being stationary implies that convolved features in one region are still useful in the other regions. A subsampling layer is introduced to reduce the resolution of the feature maps, and increases the sensitivity of output shifts and distortions.

A subsampling layer is also called a pooling layer. This displays a pooling function by aggregating the statistical features in the various positions of the input

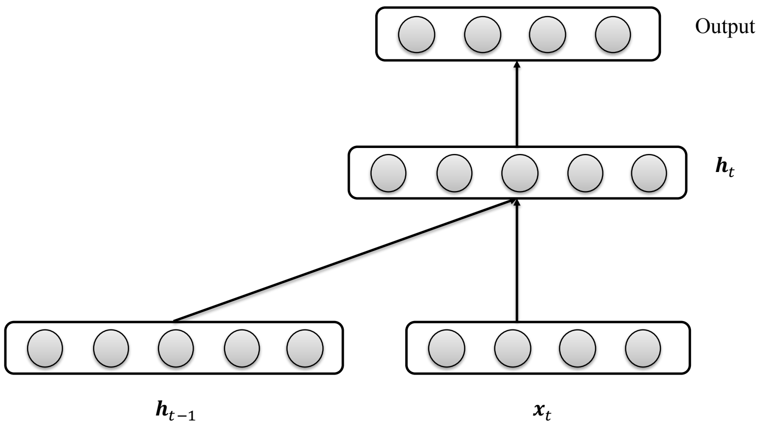


Figure 2.5: An illustration of the RNN architecture.

map. Generally, the pooling function includes a local averaging and maximizing operations. The layer's unit computes the average (mean) or maximum (max) of the feature maps of the input in a region. The pooling function is the non-overlapping approach that maps the max or mean of the input into a feature map in the next layer, and the dimension of the feature map output becomes lower than the original input dimension. Supposing the last L-th layer is the subsampling layer, the pooling function after the convolution process is

$$\boldsymbol{x}_i^{(l+1)} = f_{pooled}(\widehat{\boldsymbol{x}}_i^{(l)}), \forall l \in [2, 4, \ldots, L], \tag{2.2.3}$$

where f_{pooled} represents the pooling function, denoted as max or mean operation. In general, these features after pooling go through the convolution layer as the input.

2.3 Recurrent Neural Networks

2.3.1 Theoretical Foundations

In the previous sections, we introduced typical Artificial Neural Networks (ANNs) (e.g. CNN, AE) whose connections belong to a end-to-end approach. RNN is a cyclical connections system, considering the order of element in input sequence. It is successfully used in a variety of NLP tasks, including NMT [75, 76], MRC [77] and QA [78].

The RNN-based language model (RNN LM) is called Elman network [39, 79]. This structure contains previous input, hidden and output layers. In general, RNN is an instance of extension of Multi-Layer Perceptron (MLP), whereas the core operation is quiet different. The MLP focuses on mapping from input to the output units, whereas RNN in principle map the whole history from previous inputs to each current output. The key component of RNN is that the recurrent connections permit a memory of previous inputs to persist in the internal hidden state, which is used to affect the network output. In this section, we focus on a basic RNN containing a self connected hidden layer, as shown in Figure 2.5.

2.3.2 Forward Propagation

The forward propagation of an RNN is similar as that of an MLP with a single hidden layer, different from MLP, the activations of the hidden layer based on both the current external input and the previous hidden layer activations one step back in time. Assuming a length T input sequence $\mathbf{X}$ presented to RNN, and a sequential of elements is expressed by $[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T]$, where $\boldsymbol{x}_t$ is a d-dimensional input vector at current time t. In particular, the previous hidden state $\boldsymbol{h}_{t-1}$ changed to be a part of inputs with the current input vector $\boldsymbol{x}_t$. Since the same operation is executed for the previous layers, hidden layer output sequence is $[\boldsymbol{h}_1, \cdots, \boldsymbol{h}_T]$, where $\boldsymbol{h}_t$ is the n-dimensional output hidden state for the current layer. The m-dimensional output vector is denoted as $\boldsymbol{y}_t$. Let x_t^i to be a scalar value of the i-th input unit at the current time t, and h_t^j denotes a value of the j-th hidden unit at time t. The defined parameters $\mathbf{W} = [w_{ik}]$ connect the input and the current hidden layer, $\mathbf{U} = [u_{jk}]$ connects units from the previous hidden layer to the current hidden layer. The hidden state at current time t is computed by the combination between input vector $\boldsymbol{x}_t$ and the previous hidden layer state $\boldsymbol{h}_{t-1}$, is defined as

$$a_t^k = \sum_{i=1}^{d} w_{ik} x_t^i + \sum_{j=1}^{n} u_{jk} h_{t-1}^j, \tag{2.3.1}$$

with the hidden state at current time t is computed with a nonlinear and differentiable activation function, given as

$$h_t^k = f(a_t^k), \tag{2.3.2}$$

where the output hidden activation $\boldsymbol{h}_t = [h_t^k]$ is a n-dimensional vector at time t. The complete sequence of hidden activations is computed by starting from $t = 1$, then recursively applying 2.3.1 and 2.3.2 with the increase of time t at each step. The parameters $\mathbf{W}$ and $\mathbf{U}$ are shared between the input vector and previous hidden state when both dimensions are the same. Note that the required initial values h_0^j are commonly set to 0 or random distribution, [80] has shown that using nonzero initial values improves the robustness and stability of RNN to noise. Denote the parameter $\mathbf{V} = [v_{kh}]$ is a $n \times m$ dimensional matrix. The output $\boldsymbol{y}_t$ is calculated by the hidden state at the current time t as

$$y_t^h = g(\sum_{k=1}^{n} v_{kh} h_t^k), \tag{2.3.3}$$

where the output y_t^h is a scalar at h-th element of vector $\boldsymbol{y}_t$ at time t. The activation function $g(\cdot)$ can be set to a linear or non-linear functions (e.g. softmax function and logistic sigmoid) depends on different tasks. For instance, sequence prediction task [39] uses the softmax function as their activation function.

2.3.3 Backward Propagation

Apart from forward propagation for RNN, the backward pass for each training epoch is significant for optimizing the network. Supposing the partial derivatives of the objective function is given with respect to the outputs from RNN, it is important to compute the derivatives with respect to the forward parameters $\mathbf{W}, \mathbf{U}$ and $\mathbf{V}$ of RNN. There are two commonly methods have been proposed to successfully compute the weight derivatives of RNNs, one is called back-propagation through time (BPTT) [81], the second one is real time recurrent learning (RTRL) [82]. In the backward pass for RNN, we employ BPTT method to compute the derivatives of weights since it is more efficient and conceptually simpler with a less computation.

Similar to a standard backward propagation for MLP, BPTT is composed by a repeated process of the time chain rule. It is worth noting that for recurrent networks like RNN, the objective function Eq.(2.3.2) relies on the activation in the hidden layer not only affect the output layer, but also the hidden layer at the next time step

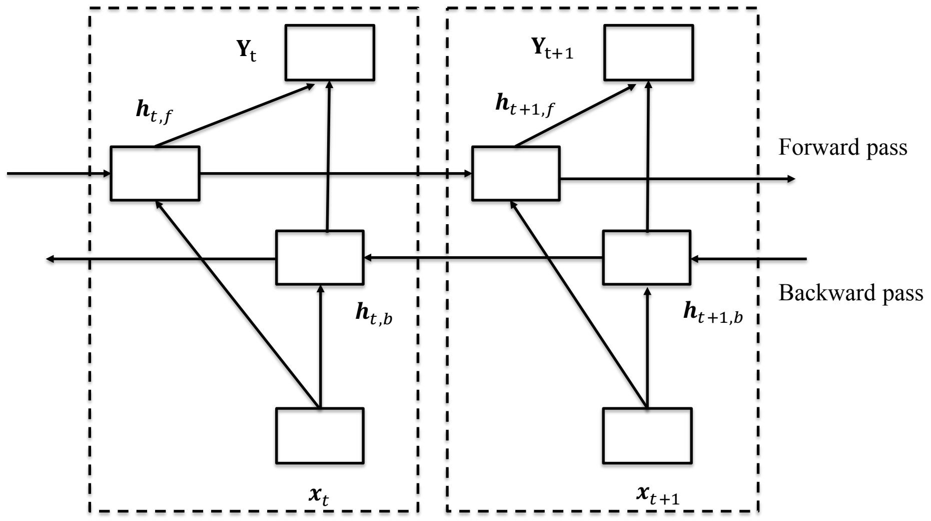


Figure 2.6: An illustration of the BRNN architecture.

in a recurrent network. Therefore, the partial derivative of the activation $\boldsymbol{a}_t$ in hidden layer is obtained by the hidden layer at the next time step $t+1$, defined as

$$\delta_t^k = f'(a_t^k)\left(\sum_{h=1}^{m}\delta_t^h v_{kh} + \sum_{j=1}^{n}\delta_{t+1}^j u_{kj}\right), \tag{2.3.4}$$

where

$$\delta_t^i \overset{\text{def}}{=} \frac{\partial L}{\partial a_t^i}, \tag{2.3.5}$$

with the cross entropy loss of RNN is denoted as L, and the symbol $\overset{\text{def}}{=}$ denotes the equal definition function. The δ_t^i term is defined by the derivative of i-th element of activation at time t, this term is calculated by starting at final time step $t = T$, and recursively computing a partial derivative function following with the decrease of t.

2.3.4 Bi-directional RNNs

Bi-directional recurrent neural networks (BRNN) [83] are composed by a training sequence forwards and backwards of two separate recurrent hidden layers, both of which are connected to the same output layer, as shown in Figure 2.6. Normally, we compute the current output according to the past context information by a forward propagation for RNN. However, the context in future is significant as well as that in past for a series of tasks, such like sequence labeling. For instance, when recognizing a special hand written letter, it is useful to obtain the subsequent letters coming after the target letter as well as those before. While a standard RNN process sequences only considers a forward order of letters, regardless of the subsequent context.

An efficient approach is to pad the subsequent context to the input sequence so that increasing the dimension of the input, this method called time-window. Although it can provide the future context to predict the current letter, it suffers from the issue that a range of useful context is generally unknown, it may cause the required time-window size becomes very large. Thus, a number of unnecessary input weights is increased to cause expensive computation. Another approach is given a delay between the inputs and the output through providing the network a few time steps of future context. This method remains the robustness to distortions of RNN, but it still requires the range of future context to be determined by hand. Furthermore it places an unnecessary burden on the network by forcing it to remember the original input, and its previous context, throughout the delay. In above approaches, neither of these approaches remove the asymmetry between past and future information.

Differently, BRNN offers a better solution to solve the above issue, it provides a symmetrical network to collect the past and future context together for each step time in the input sequence, without changing the input based on the relevant output. In recent, BRNN brings an improvement results in various fields, such as machine translation [34], machine comprehension [43] in NLP tasks and protein secondary structure prediction [84] in the biological area.

The forward propagation for the BRNN hidden layers is the same as for the traditional RNN. In particular, the input sequence is pointed in the opposite directions to the two hidden layers, and the output layer of the BRNN stops updating until both two hidden layers have processed the entire input sequence. The total hidden state at time t is represented by $\boldsymbol{h}_t = [\boldsymbol{h}_{t,f}; \boldsymbol{h}_{t,b}]$, where the notation $[\cdot;\cdot]$ represents the aggregation between forward and backward hidden states. The activation of the output layer is computed by using the two hidden layers, given as

$$\boldsymbol{y}_t^h = g(\sum_{k=1}^{n} v_{kh}[h_{t,f}^k; h_{t,b}^k]), \tag{2.3.6}$$

where the activation $\boldsymbol{y}_t^h$ is a 2-dimensional vector at h-th element of matrix $\mathbf{Y}_t$ at time

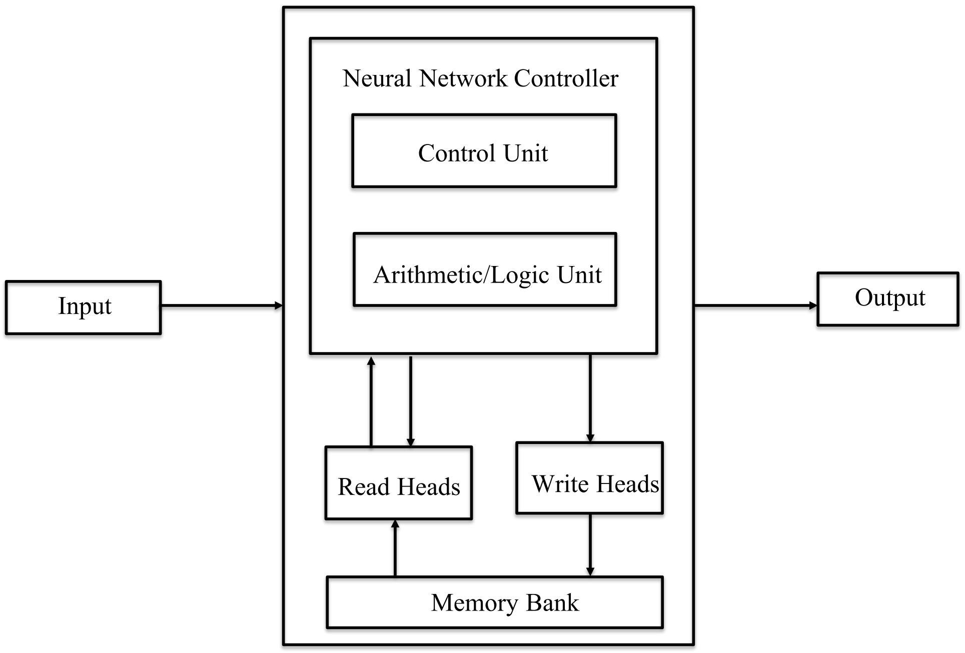


Figure 2.7: An illustration of the NTM architecture.

t. The forward and backward hidden states are separately computed from Eq.(2.3.1) and Eq.(2.3.2). Additionally, the backward propagation for BRNN is the same as it for a basic RNN trained with BPTT method. In particular, the BPTT backward propagation first computes the derivative terms for the output layer, then calculates that for two hidden layers in an opposite directions for BRNN.

2.4 Neural Turing Machines

2.4.1 Theoretical Foundations

Neural Turing machine (NTM) is a structure of recurrent neural networks proposed by [85], it allows to selectively store the information for a variable length of time. The NTM architecture consists of two main components as illustrated in Figure 2.7, where one is the neural network controller, and the remaining one is a memory bank. In fact, the controller of NTM is an optional neural network that is defined according to the specific task, e.g. MLP, RNN and LSTM. Similar to the standard neural networks, the one of usages for the controller is that interacting to external resources through attention mechanisms for input vectors. In particular, an additional function of the controller is to connect with a memory bank by adopting read and write operations, where the controller can selectively collect required information from memory, meanwhile it also leads to a change of content in memory.

The interactive architecture of NTM belongs to the differentiable end-to-end system, hence it is possible to efficiently optimize the parameters using a standard optimizing method including gradient descent algorithm [24]. To achieve it, the read and write operations are used to interact with a certain number of elements in memory. An attentional focus mechanism determines the number of elements from memory that connects to the controller. In general, the attention mechanism constrains both read and write operations to interact with a small portion of the memory, since the NTM tends to store the output data from the controller without interference so that the interaction is set to highly sparse when connecting the elements in memory.

Differentiable neural computers are an outgrowth of NTMs, with attention mechanisms that control where the memory is active, and improved performance. The memory location brought into attentional focus is determined by specialized outputs emitted by the heads. These outputs define a normalized weighting over the rows in the memory matrix. Each weighting, one per read or write head, defines the degree to which the head reads or writes. Several memory models derived from NTM offer given improved results in machine comprehension [59, 86]. For example, given question and context as the inputs, the memory provides information required to predict the target answer for a given question.

2.4.2 Data Extraction

In this section, we describe the read operation that extracts data from the memory bank to controller. Assuming M_t denotes a $N \times M$ dimensional memory matrix at time t, where N is defined as the number of positions in memory matrix, and M is the dimension of each position. Let the parameter $\boldsymbol{w}_t = [w_t^i]$ is the vector of weightings for N positions released by a read head at time t, where w_t^i denotes a value for the i-th position of memory matrix at time t. The read vector returned by the head is defined as $\boldsymbol{r}_t$ at time t. The weight w_t^i of $\boldsymbol{w}_t$ attends to each position of memory M_t^i for computing the read vector $\boldsymbol{r}_t$, given as

$$\boldsymbol{r}_t = \sum_{i=1}^{N} w_t^i \mathbf{M}_t^i, \tag{2.4.1}$$

where $\boldsymbol{r}_t$ is a M-dimensional vector by the weighted sum function, and it is clearly differentiable with respect to both the memory and the weighting. The normalized weightings $\boldsymbol{w}_t$ over the N positions of rows in memory matrix, are constrained by

$$\sum_{i=1}^{N} w_t^i = 1, \forall i \in [1, \ldots, N] \tag{2.4.2}$$

where the N elements weighting w_t^i of $\boldsymbol{w}_t$ is belong to [0,1] interval. The read vector $\boldsymbol{r}_t$ contains data from memory is utilized to the controller. In addition, the controller is capable of deciding the magnitude of the weightings applied to memory.

2.4.3 Memory Updating

Memory update mechanism of NTM focuses on modifying the content in memory based on each position of the previous memory matrix $\mathbf{M}_{t-1}$ by using the write operation. It consists of two operations: erase and add. Given a vector w_t of weightings returned by a write head at time t. Assuming $\boldsymbol{e}_t$ denotes a M-dimensional the erase vector, whose elements belong to the range [0,1]. The vector $\mathbf{M}_{t-1}^i$ of i-th position of memory matrix at previous time $t-1$, is refined as follows

$$\tilde{\mathbf{M}}_t^i = \mathbf{M}_{t-1}^i [\boldsymbol{v} - w_t^i \boldsymbol{e}_t], \tag{2.4.3}$$

where $\boldsymbol{v}$ is defined as a row vector with all element values of 1, and it acts point-wise operation with respect to the multiplication of memory locations. In particular, the elements of a memory location are equal to zero when both the weighting w_t^i at the i-th location and the elements of erase vector $\boldsymbol{e}_t$ are one. The elements of memory are unchanged when either the weighting or the erase is zero.

Besides, the write head also defines a M-dimensional add vector $\boldsymbol{a}_t$ for refining memory. The add vector focuses on the memory matrix $\tilde{\mathbf{M}}_t^i$ computed from the erase step, thereby the add function is defined as

$$\mathbf{M}_t^i = \tilde{\mathbf{M}}_t^i + w_t^i \boldsymbol{a}_t, \tag{2.4.4}$$

where the vector M_t^i for the i-th position of memory matrix at time t has been updated by the combined erase and add operations of the write head. Subsequently, the memory matrix M_t contains the final content is applied to the read head for providing the newest information for the controller. Note that the write operation is differentiable end-to-end since both erase and add functions are differentiable. The erase and add vectors contains M elements, which is the same as the read vector, NTM allows them to refine over the specific elements in each memory location.

2.4.4 Addressable system

In the previous sections, we have shown the operations of reading and writing, respectively. Here, we briefly introduce the weightings produced by read and write heads of the controller. In NTM, two addressing mechanisms with complementary facilities focus compute the read and write weightings: content-based addressing and location-based addressing. The content-based addressing mechanism focuses on the interactive relationship between the current memory matrix and the outputs from controller by computing the semantic similarity. By using the similarity operation, the goal of the content-based addressing mechanism is to make the memory to produce a portion of data is approximate to the output produced by the controller. Suppose $\boldsymbol{k}_t$ denotes the M-dimensional output vector from a head of the controller. The content-based addressing mechanism computes the normalized weighting between the output $\boldsymbol{k}_t$ and a memory vector M_t^i defined as

$$\tilde{\boldsymbol{w}}_t = \frac{exp\left(\beta_t cos[\boldsymbol{k}_t, \mathrm{M}_t^i]\right)}{\sum_{j=1}^{N} exp\left(\beta_t cos[\boldsymbol{k}_t, \mathrm{M}_t^j]\right)}, \tag{2.4.5}$$

where the parameter β_t represents a positive key strength for controlling the magnitude of similarity between $\boldsymbol{k}_t$ and M_t^i. The operation $cos[\cdot,\cdot]$ is represented by the cosine similarity function by

$$cos[\boldsymbol{k}_t, \mathrm{M}_t^j] = \frac{\boldsymbol{k}_t \cdot \mathrm{M}_t^j}{||\boldsymbol{k}_t|| \cdot ||\mathrm{M}_t^j||}, \tag{2.4.6}$$

where the notation $\cdot$ is an Euclidean dot product, and $||\cdot||$ represents the length of the vector, which is commonly computed by the Euclidean norm. The cosine similarity output determines the weighting of content-based addressing.

The content-based addressing mechanism computes the weighting over each position of matrix based on the content of data from memory and the controller. Followed by this, an alternative approach is location-based addressing mechanism updates the weighting over entire locations of memory, where this design is suitable for storing the arbitrary content of data to memory. The content-based addressing mechanism contains several transformations of a weighting, e.g. interpolation, shift, and sharpening. Assuming g_t denotes a interpolation gate. The weighting vector $\tilde{\boldsymbol{w}}_t$ from content-based addressing is interpolated with the weighting w_{t-1} from the head at the previous time $t-1$ according to the value of interpolation gate g_t, thereby the gated weighting$\hat{\boldsymbol{w}}_t$ is defined by

$$\hat{\boldsymbol{w}}_t = g_t \tilde{\boldsymbol{w}}_t + (1 - g_t)\boldsymbol{w}_{t-1}, \tag{2.4.7}$$

where $\hat{\boldsymbol{w}}_t$ is a N-dimensional vector. In particular, the gated weighting $\hat{\boldsymbol{w}}_t$ only depends on the weighting $\boldsymbol{w}_{t-1}$ at the previous time when g_t is set to zero. Oppositely, the gated weighting $\hat{\boldsymbol{w}}_t$ equals to content weighting $\tilde{\boldsymbol{w}}_t$ by ignoring $\boldsymbol{w}_{t-1}$. Let $\boldsymbol{s}_t$ defined as a shift weighting vector. The gated weighting $\hat{\boldsymbol{w}}_t$ is used to shift operation by indexing the memory positions from 0 to $N-1$, defined as

$$\bar{w}_t^i = \sum_{j=0}^{N-1} \hat{w}_t^i s_t^{(i-j)}, \tag{2.4.8}$$

where the number of shift index is under N. The shift transformation may lead to the leakage or dispersion of weightings over time when the shift weighting is not sharp. To solve this, the head denotes a scalar γ_t to normalize the shift weighting as

$$w_t^i = \frac{\bar{w}_t(i)^{\gamma_t}}{\sum_j \bar{w}_t(j)^{\gamma_t}}, \tag{2.4.9}$$

where the value of γ_t is bigger than 1. Overall, the weighting from read and write heads is refined by the combined addressing system including the content and location-based addressing mechanisms.

2.5 Conclusion

In this chapter, we have explored a number of deep learning methods in the field of learning high-level representation. These generic methods lay the foundations for developing new neural network architectures and their key ideas have been repeatedly applied and developed in the entire field of machine learning. We summarise their main ideas here: **a)** AE aims to compress and reconstruct input with an *unsupervised* training. In compression procedure, an encoder is used to map the input to a hidden representation with dimensionality reduction. In reconstruction process, a decoder reconstructs the input from the hidden layer. The variation models of AE includes DAE and SAE. The key idea of DAE is to learn a robust representation from the disturbed input. SAE conducts an unsupervised pre-training in a layer-by-layer approach. The pre-trained layer conducts feature selection and extraction on the input from the preceding layer, followed by supervised fine-tuning. **b)** CNN consists of three kinds of structural ideas: a local receptive field, weight sharing and temporal or spatial subsampling to obtain a certain degree of displacement, scale and deformation invariance. The *locally* connection approach between two layers benefits to reduce the intermediate parameters in network. **c)** RNN is an instance of *recursive training network*. It learns a high-level representation by aggregating the whole information of a input sequence. Note that in the forward pass, the current hidden state depends on the hidden state at the previous time, whereas in backward pass, the derivative of it depends on the hidden state at next time. **d)** An NTM is a *differentiable memory network*. It defines a readable and writable memory to absorb the external resources and return the useful information to the controller. In particular, similar to RNN, the content of the memory is updated on time according to that in the previous state.

Recently, above neural network models have been applied to numerous fields, e.g. machine translation in NLP [34], image caption generation in visual QA [27], and speech recognition in audio processing [25]. However, we have come across the semantic matching problem using this generic models for related QA tasks, eg. cQA, MRC. This problem is important to almost QA models since the semantic matching between question and answer determines the model performance. In this chapter, we have closely studied the properties and designs of various neural network models, and clarifying their mathematic algorithms will certainly be helpful to new algorithm designs. In the next Chapter 3, we will describe the background of semantic matching and its literature review.

Chapter 3

Semantic Sentence Matching

In this chapter, we review the related works of semantic sentence matching in cQA. Multiple designs of architectures are investigated for the relevance between a pair of sentences. Generally, most of works studied the semantic sentence matching based on the generic neural network models, e.g. CNN, RNN. Semantic matching measures the model performance that is significant to be explored in details.

3.1 Semantic Matching

The most basic technique for detecting semantic similarity between text objects is lexical matching, e.g., by evaluating the string similarity between words [9]. The main drawback of such techniques is that they are unable to recognize similarities between synonymous words [87–89]. To tackle this problem, word co-occurrence information obtained from large text corpora (e.g., Wikipedia, newswire text) or hierarchy information drawn from semantic networks (e.g., Wordnet) can be utilized to formulate the semantic similarities between words [8, 9]. Another strategy is to characterize each word with a vector and compare the words through a similarity function, e.g., inner product and cosine similarity [90]. Popular ways for generating a word vector embedding include bag-of-word representation based on the contextual words around the targeted word [54], latent semantic analysis [91], and distributed embedding representation generated by a probabilistic neural language model [28,55].

Given the established similarities between words, sentence similarity can be further established obtained based on an element-wise comparison of words resulting in a similarity matrix between two sentences [92]. Such a matrix no longer contains much of the syntactic or global structure of sentences [31].

In recent years, deep neural networks have been shown to be effective in characterizing the latent relationships between words, returning distributed embedding representation for words, phrases, sentences, paragraphs, and even documents. They have been widely applied to various NLP tasks such as paraphrase identification to detect whether two sentences share the same meaning [35,93], and NMT to match a source sentence in one language to a target sentence in another [34,94]. This is a highly dynamic research area and new architectures are being proposed and refined. In particular, Convolutional Neural Networks (CNNs) have been used to integrate the word patterns into a representation that preserves the semantics of a given sentence [33,35,93]. Additionally, recursive auto-encorders can learn sentence representations by encoding the parse tree structures of sentences [31], and recurrent neural networks (RNNs) can retain the key information of words spread throughout a sequence [94] using architectures such as the Long-short Term Memory (LSTM) [41]. Bi-directional RNN takes into account information from both the preceding and following words, and can be applied with the support of an alignment model to produce a context vector to aggregate word information in a sentence [34]. Recently, a hierarchical modification of a RNN has been developed jointly to model words, sentences, and paragraphs by treating a paragraph as a sequence of sentences and a sentence as a sequence of words [61]. There has been increasing interest in using an attention mechanism in NLP. An attention mechanism would enable a network selectively to focus on the key parts of a sentence, for instance, through the use of attention driven RNN [4,95] or CNN [37,96]. In addition to comparing the similarities between text objects, semantic matching can also be performed between an image and a portion of text, e.g. sentence description generation for images [27,97]. A popular approach for this is to build a compositional network that contains a CNN (for image processing) and a RNN (for text processing) to model the similarity between image and text representations [98].

3.2 Community Question Answering

QA is the task of producing a machine that can automatically answer questions posed by humans using natural language. Given a question, QA consists of two typical schemes. The first approach is to select the best answer from an existing pool of answer candidates, known as cQA. The other is a machine dialogue task which requires the computer to automatically to generate a novel answer, based on a natural language model [49]. On the other hand, cQA is structured around a website that allows users to post their questions and for other users to submit answers [99, 100]. The answer selection task for cQA can be treated as a semantic matching task between the question and answer text; this matching task is the focus of this study.

Commonly, cQA has undergone a spectacular increase in popularity in recent years. With the advent of relevant websites (e.g. Stack Overflow [21] , Yahoo! [19]), an increasing number of people can freely post any question and expect a variety of answers. With the influx of new questions and the varied quality of the answers provided, it is very time-consuming for a user to inspect them all. Therefore, developing automated tools to identify good answers to a question is of practical importance [101]. The problem of redundancy and noise is prevalent in CQA [102]. Both questions and answers contain auxiliary sentences that do not provide any meaningful information.

In the past, the traditional QA approaches treated a series of documents as a type of memory to retrieve answers from these documents corresponding to the question. A set of recent works attempt instead to generate a graph of facts called a knowledge base [8, 57], as the memory, and map the questions to logical queries.

3.3 Traditional Approaches

The traditional approach is IR-based techniques [3] adopts the retrieval methods to collect the specific information from a set of documents. The retrieval methods search the answers from the website through detecting the entities for question. Considering the other traditional technique, lexical matching is employed to detect semantic similarity between text objects. For instance, Islam et al. [9] has evaluated the string similarity between words, and Chen et al. [103] has developed a feature-based system, computing the similarity distances between words using a variety of statistical methods. One of the main drawbacks of using such techniques is that the similarity between synonyms cannot be well captured directly from the text [87, 89].

This synonym-specific problem, however, can be addressed through a number of methods. One approach is to pre-compute or pre-load word co-occurrence information based on one or more large text corpus, such as Wikipedia. Another method is to leverage word hierarchy information drawn from semantic networks, such as WordNet, as in [8, 9]. Characterizing each word with a vector and comparing the words through a well-defined similarity function, such as cosine similarity [90], can also counteract this specific problem. A number of techniques exist for generating an embedding vector for a word. Popular methods include Bag-Of-Words (BOW) representation based on the contextual words around the target word [37, 54], Latent Semantic Analysis (LSA) [91], distributed word embeddings generated by a probabilistic neural language model [28, 55], and Gaussian distribution embedding [104].

Once the similarities between words have been established, the similarity between a pair of sentences can be derived based on an element-wise comparison of words using techniques like the syntactic tree kernel [105, 106], Tree Edit Distance (TED) [107] and its multiple variations [53, 108]. In particular, TED is a flexible, lexical-based matching technique, which compares two sentences based on the number of operations required to transform one sentence into the other, where the operations include inserting a word, substituting a word, moving the position of a word within the sentence, and deleting a word [107]. There are multiple variations of TED models, for instance, ones that use tagging sequences [109] and ones that consider both lexical semantic resources and distributional representations [53]. Another variation for transforming complex sentences is to first extract shorter and more intuitive sentences using a tree kernel heuristic [110]. The probabilistic TED model [108] uses tree-edit operations to transform a question into an answer, and decides whether an answer is relevant according to whether or not the given question can be successfully transformed into this answer. There are also cQA approaches aimed at extracting structural patterns that contain syntactic information in sentences, e.g., the feature-based one [106] and [105] based on a syntactic tree kernel. These techniques return a similarity matrix [92] between two given sentences. Nevertheless, despite measuring the similarity, the similarity matrix may not reflect the syntactic or global structure of the sentences [31].

3.4 Neural Semantic Models

Deep neural networks have been proven to be effective for generating distributed embedding representations of text objects (e.g., words, phrases and sentences) and characterizing the latent relationships between them. In the context of cQA, they have been widely applied to handle a number of problems, such as identifying paraphrased sentences [31, 111], detecting shared meaning between sentences [35, 112], and for syntactic parsing to capture semantic relationship between phrases [113, 114]. In this section, we review neural network architectures for answer selection in cQA, we not only explain the basic ideas of the existing works but also note their limitations, which has motivated the design of the proposed network architecture and its training.

A simple way to use a neural network for answer selection is through feature learning. For instance, [16] employs a Restricted Boltzmann Machine (RBM) to combine bag-of-word features and non-textual features for a given sentence and then feeds the fused features to a classifier to decide the best answer. An alternative way is the CNN-based approaches that have been very successful in image representation learning and very popular in text representation learning. By characterizing a sentence as a set of word embedding vectors (the set of word embedding vectors constitutes a *distributed representation*) stored in a sentence matrix [11, 32], a CNN can be typically employed to compute a vector representation for the sentence from its input matrix. A CNN applies the convolution operator between a weight matrix and the sentence matrix to extract contributing patterns corresponding to word sequences in sentences [37, 115]. Each sentence is represented by a new vector in another distributed representation, from which a scalar similarity score can be computed for a sentence pair. The advantage of such a design is that the convolution feature map enables the modeling of (1) the semantic information between words within the same sentence and (2) the syntactic information of a sentence via extracting the contributing word sequences. Similar to a CNN, an auto-encoder can be applied to learn the sentence representation from word embeddings [19]. Overall, all of these techniques formulate sentence representation without consideration of the matching task, and although the sentence representations will preserve semantic information, the detailed word-level information that may be crucial for matching the two sentences, will be lost.

There are other approaches that attempt to encode the semantic links between the questions and answers that are important for matching. For instance, a latent sentence representation can be learned in order to reconstruct a question from its answer using a deep belief network [116]. To model more accurately the semantic relevance between sentences, some architectures have begun to combine similarity information directly with representations of sentence pairs. In particular, the similarity score computed from a CNN-based sentence representation can be treated as an intermediate feature and combined with sentence representations themselves prior to further processing [33]. That is, the sentence representations returned by CNN for each sentence are concatenated with the scalar similarity score. The concatenated vector is compressed to a dense vector of lower dimension by a fully connected neural network. Although this model learns a distributed similarity representation tailored to the matching task, it starts from the CNN-based sentence representation which, as mentioned earlier, has already suffered a information loss of the detailed word-level information and thus is not optimal for improve the matching.

Focusing on the word-level semantics, Lu and Li propose a model that recognizes the similarity between each pair of words using a topic-based neural network architecture [29]. Specifically, each first-layer neuron is associated with a topic and the neuron is active if both sentences have at least one word associated with this topic. The cross-domain topics [117] are modeled using latent Dirichlet allocation (LDA) [118]. The output of an active neuron is a function of a linear weighting of the bag-of-words representation for each topic. In total, an overall relevance score is computed through a further two hidden layers, where the second hidden layer is also controlled by the topic hierarchy. Although this model directly considers the word-level similarity, the information flow in the model is limited by the initial architecture, which compresses the representation based on the topic hierarchy, which may not be optimal for the matching task.

Another approaches work on word-level semantics between sentences, where CNN learns the distributed similarity representation from sentence embedding matrices [2, 5]. The network architectures use a 2D convolution feature map to encode the detailed segment-level similarity between sentences, but employs a 1D convolution feature map to encode word-level similarity between the sentence segments. The input of this 1D map is a simple concatenated vector of two individual segment vectors of word occurrences. In this case, local word interactions between sentence segments will not be considered thoroughly, thus detailed word-level similarity may be lost under the segment scale. To address these issues, we propose an alternative deep matching model in Chapter 4 that directly utilizes the word-level similarities to extract a robust similarity measure.

However, none of the above approaches account for the order of the words in a sentence. In recent years, RNN [39] have become a popular choice for processing natural language due to their effectiveness in modeling the word order information within a sentence. For instance, [75] uses a bi-directional RNN facilitated by an alignment model [98] to compute the sentence representation for the NMT task. In QA related tasks, [56] characterizes the sentence by using a stacked bi-directional LSTM, while [78] uses a bi-directional LSTM. In [78], the multiple hidden representations returned by a bi-directional LSTM at different states are used to compute a similarity matrix between the question and answer sentences. It is a common choice that learns a sentence representation using RNN or the variation model of RNN (e.g. LSTM [41], GRU [42]), and Wang et al. [119] has utilized a bi-directional LSTM to learn the word position representation of each time step in a sentence.

3.5 Attention Mechanisms for cQA

The attention mechanism, first proposed in [75] for the NMT task, enables a neural network to identify the salient components of a sentence. It tends to rely on a weighted sum of a set of component representations, where the attention weights control the contributions of the components. The softmax function is typically used to convert a set of importance scores to a set of positive attention weights that sum to unity. Different ways of designing attention mechanisms correspond to different strategies of defining the components and formulating their importance scores. In Chapter 5, we refer to a function that is used to compute these importance scores as an attention function.

A typical way of incorporating an attention mechanism in an RNN- or LSTM-based cQA system, is to relate the different components to the different hidden states of the network, which correspond to the different word positions in a sentence. The final sentence representation can be expressed as a weighted sum of the hidden representations computed at these states. In [95], the importance score is formulated as a function of each hidden representation itself, and focuses solely on the contribution of the word position within the target sentence. In [120], the attention mechanism is applied to the answer sentences, where the importance score is computed from not only the hidden representation of the answer states, but also the question representation returned by a bi-directional LSTM. This results in an interactive attention mechanism between answers and questions. Similar strategies to [120] are also proposed in [121, 122]. More sophisticated attention mechanisms are developed by considering more factors that may affect the importance score. For instance, [97] takes into account the previous episode memory, while [102] considers the question topic and question type in cQA, as well as the question and answer interaction information.

Instead of using attention mechanism in the learned representations from specific network, an alternative way to set the attention mechanism is to examine the importance of the word pairs that appear in the given sentence pair. For instance, given a question sentence containing n words and an answer sentence containing m words, each element in the $n \times m$ attention weight matrix indicates how much a word pair contributes to the relevance of the two given sentences. The importance score of each word pair can be computed from their corresponding word embeddings [37] or the hidden representations at the corresponding word positions returned by an LSTM [123], through the use of Euclidean distance or dot product. [1] measures the semantic interactions of word pairs from similarity matrix between the encoded sentences representations, which come from bi-directional LSTM. A soft alignment representation is computed for each word in sentence using an attention mechanism in word-level similarity matrix.

Variations of attention mechanism can be developed in a bespoke manner to suit a specific task, for instance, by taking into account an external knowledge base [44], by implementing an attentive max-pooling operation for CNN [38, 96], or by joining the internal documents into given question using co-attention attention in MRC task [123], etc. Typically, [40] explores an sentence-aware word attention on each word position representation of a sentence before computing the RNN representations. Besides the CNN or RNN-based attention models, a recent auto-encoder with attention model [124] applies a hidden representation from the encoder to reconstruct sentence representations in the decoder for question retrieval. Recently, self-attention has emerged as an attention mechanism aimed at aligning the multiple positions of a sequence, which has been widely used in a variety of the related QA tasks, for instance, machine reading comprehension (MRC) [77, 125], NMT [76] and abstractive summarization [36]. For instance, [126] provides the fusion functions to combine self-attention and similarity matrix based attention to complete the related MRC task. In cQA, [101, 127] apply a multi-dimensional self-attention mechanism to question and answer embeddings, and an attention weight vector instead of a single attention scalar is computed to learn word-level alignment representation. In Chapter 5, we extend the self-attention mechanisms by involving more contextual information in cQA datasets.

3.6 Memory Networks

General neural network memory models such as NTM consists of the differentiable memory and controller that reads and writes to specific locations [85]. Recently, a series of deep neural network models adopt the memory architectures for machine comprehension [128, 129], image caption generation [97] and dialogue system [6]. Memory network has been employed for MRC with internal resources, such as supporting contexts or facts, in most cases. The MRC is derived by QA task, the difference being that the knowledge-base of external domain is typically required to answer questions [13, 128], whereas in machine comprehension answers are inferred from a given text.

A Memory network called MemNN has been proposed in [58], first introduced the concept of a long-term memory component for MRC. MemNN aims to store useful contextual information from supporting facts, and produces relevant memories to support given input. Followed by this, as a strongly supervised model, MemNN generates a single word relies on the supporting facts as the answer for given question. A series of attentive memory networks [86, 130] are recently proposed to strengthen memory by storing the relevant information. Dynamic Memory network (DMN) [97] improves over it by employing an end-to-end trainable network with an attention mechanism. The memory iteratively produces a vector to store the relevant input information, which is used in answer generation. End-to-end memory networks (MemN2N) [86] encodes sentence to a continuous vector representation depends on a recurrent attention mechanism instead of sequence aligned recurrence, it has been shown to perform well on simple-language machine comprehension and language modeling tasks. Different from the MemNN and DMN architectures, MemN2N uses an end-to-end mechanism with a weakly supervised training. However, the disadvantage of MemN2N is that it only generates answers with one single word. Multi-layer Embedding with Memory Network (MEMEN) [59] provides a hierarchical attentive memory to learn an alignment memory matrix, which contains the syntactic and semantic information of the words returned by skip-gram model. The proposed model in Chapter 6, exploring to use memory network in QA task without external sources and knowledge-base.

3.7 Conclusion

In this chapter, we have introduced a variety of neural network methods crossing different periods for tackling semantic matching problem. The traditional model is also called shallow learning use the statistical methods to capture lexical information including BOW, TED and LSA. The variants of deep learning models focus to learn the high-level semantic similarity representation through multiple layers. Additionally, the attention mechanism as an efficient tool improves the relevance between question and answer, by involving the contextual information to the model. A typical memory network utilizes its core memory not only store the content related to the question or answer, and selectively release that to support the semantic matching. It is helpful for improving the model performance, especially for a large scale of the dataset. Although there exist multiple types of models can complete QA matching, we still come across a series of problems, such as the model complexity, the small storage of memory, and external resources.

In the next following sections, the proposed models will provide the solutions with respect to above mentioned problems. For instance, in regard to the complexity and computation speed, the proposed method in Section 4 solves the problems by training model in a decomposition way. The proposed memory network model in Section 6 is able to store relevant information in specific locations of memory, and generate internal interactive information to support semantic matching.

Chapter 4

Greedy Word-level Semantic Similarities

4.1 Introduction

In Chapter 2 and Chapter 3, we have reviewed some of conventional deep learning methods and the background of semantic sentence matching. Semantic matching requires accurately modeling the relevance between two portions of text and is widely used in various NLP tasks, such as paraphrase identification [31, 35], machine translation [34, 61, 94], image caption generation [27, 98], answer selection [14, 86, 131], etc. For instance, in the answer selection task, given the query "where is the cat?", it is clear that "the cat sits on the mat" is a relevant answer. A semantic similarity score can be computed based on how well two portions of text are matched.

The simplest similarity scoring approach is to use lexical matching to count the number of matching words or stems, but this fails to notice the similarity between the question word "where" and the preposition "on" or the location "mat". A knowledge resource can be used to inform the matching about the relationship between words or phrases; however, this requires additional effort to manually create or curate a knowledge resource, and still requires a mechanism to combine the word or phrase-level similarity. An alternative is to use machine learning to train a model to quantify the semantic similarity. Recently, there has been a burgeoning interest in using neural language models—neural networks adapted for representing natural language – for estimating the semantic similarity between portions of text, and improved matching performance over traditional approaches has been demonstrated [29, 32, 56, 132].

In this chapter, we focus on semantic matching for answer selection in cQA [133], for which the task is to select highly relevant answers from a candidate sentence pool given a question posted by the user. It has a wide range of applications such as information retrieval, web search ranking [132], and dialogue systems [49, 134]. In order to compute an accurate measure of relevance, it is crucial to take into account the syntactic, lexical, and semantic information of the text pair. To achieve this previous works have proposed feature-based models utilizing semantic information provided by external resources, e.g., WordNet [51–53]. Although these approaches use semantic features to improve the similarity metric, their feature engineering stage can be expensive. Moreover, these approaches cannot be adapted to match language for which there is no external semantic resource available, as previously noted [32]. Replacing the manual feature engineering with a neural language model, various works have been developed to compute semantic similarity between text by learning distributed representations of words via neural networks [30, 31, 33].

The state-of-the-art models for this task have used various architectures for the task. Fundamentally, the models can be divided into those that derive the similarity score from sentence level representations [32, 33] and those that use directly the word-level similarities [5, 29]. Here we propose an architecture that uses word-level similarity, based on a vector-space embedding of words, and then builds an abstracted representation of this similarity using stacked auto-encoders, with the aim of both preserving and denoising the information contained by the set of word-level similarities. Our hypothesis is that, if properly assessed, the set of word-level similarities is sufficient for the semantic matching task. Essentially, the proposed neural network is able to learn how to distill a set of pairwise similarities to a single relevance measure.

Specifically, we propose a deep matching model that represents the similarity between a pair of sentences using a distributed similarity vector to capture the semantic similarity between all words in the sentences. We refer to the approach as Deep Semantic Similarity Model (DSSM). In the proposed DSSM, given an input sentence pair, the similarity between each pair of words is represented by a matrix computed from the individual representations of the words in a distributed representation. The similarity matrix itself may contain redundant information that can unnecessarily increase the model complexity. Thus, a pooling procedure is applied to extract the similarity between the word pairs that are most discriminative to the relevance identification. Multiple stacked auto-encoders (AEs) [62] are used to denoise and refine the similarity representation across multiple layers. Unlike previous approaches [31, 33, 135] that follow the procedure of first reducing the distributed sentence representation from matrix to vector and then computing a scalar similarity score, DSSM never directly reduces the sentence representation, but gradually compress the distributed similarity representation. This can potentially avoid the problem of input information loss caused by compressing the sentence representation at an early stage [5]. We do not compress similarity information based on a pre-defined scale, e.g., topics in [29] and sentence segments [5], but use AEs to automatically encode the important information.

The DSSM model is parameterized by a substantial number of variables including the word embeddings and the network weights. All of these parameters are difficult to train jointly. In order to obtain a well-trained model, a greedy layer-wise training scheme is proposed that trains one part of a model at a time, using the previously pre-trained parameters to generate the input for the next part [136]. This scheme not only eases the training complexity, but also has been shown to yield significantly better local optimum than random initialization and offers better generalization [72].

Specifically, in the proposed greedy layer-wise training scheme for DSSM, we divide our model into three parts: (1) a distributed bilinear similarity computation, (2) a high-level deep similarity representation returned by the stacked AE, and (3) a overall similarity score computation and matching prediction. Initially, the word embeddings are formed using an unsupervised neural language model. Then during the initial training stage, a pair-wise training approach based on a supervised ranking score is used to optimize the variables in the bilinear similarity and pooling computation for part (1). An unsupervised training based on a regularized reconstruction error is then used to optimize the AE weights for part (2). The weights in the matching prediction (3) are trained based on a cross-entropy loss function. During the final training stage, an additional fine tuning of all the model parameters is conducted based on the same cross-entropy loss.

In the experiments, we show empirical comparisons of the proposed method and various state-of-the-art matching models that do not utilize any external information resources or manual feature engineering. The text retrieval conference (TREC) QA dataset and Yahoo! database of cQA are used to evaluate the performance. In addition, we analyze the performance of variations of the proposed model to empirically justify model design choices. The results show that the proposed DSSM model outperforms the state-of-the-art methods and demonstrate that the model provides a robust approach for text matching.

4.2 The Proposed Semantic Matching Model

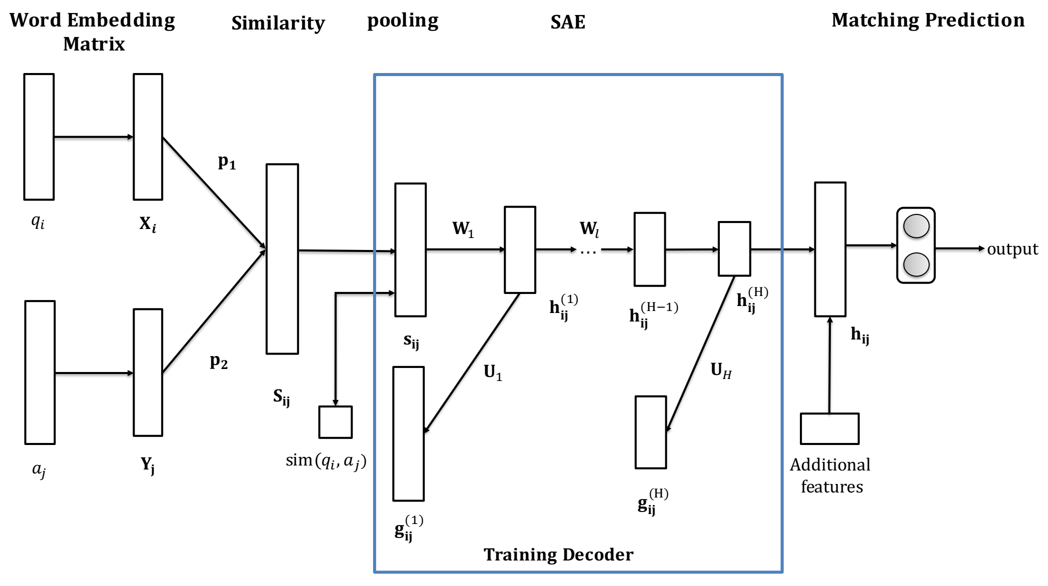


Figure 4.1: An illustration of deep architecture for matching text pair.

Given a query described by a sentence, the goal is to rank the relevant answer sentences from a pool of candidates. Success on this task hinges on the metric used to quantify the relevance between a query and a candidate sentence. For this purpose, we propose to train a deep similarity model that automatically learns not only the distributed representations of sentences and words, but also a parametric similarity measure between sentences to capture more accurately their semantic similarity, and can be used as a confidence score to determine whether the candidate answer is relevant to the query. The layered algorithm architecture is illustrated in Figure 4.1. It includes an input layer parameterized over the sentence and word representations, a similarity computation layer, a pooling layer, multiple similarity denoising layers, and a finally a prediction layer. A well-trained model should effectively quantify the semantic relevance between a variable length query and variable length answer.

4.2.1 Distributed Similarity

Bilinear Similarity Construction

Given a query q_i and a candidate answer a_j, it is reasonable to assume the answer's relevance depends on the semantic similarity between the words they contain. We employ a distributed vector representation to model the semantic similarity between words—each word is represented by d-dimensional vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_d]$ and the similarity between two words $\boldsymbol{w}$ and $\boldsymbol{w}'$ is proportional to $\boldsymbol{w}^T\boldsymbol{w}'$ or the cosine similarity $\frac{\boldsymbol{w}^T\boldsymbol{w}'}{\|\boldsymbol{w}\|\|\boldsymbol{w}'\|}$. Assuming the vocabulary size is V, all the word vectors are stored as the rows of the $V \times d$ matrix $\mathbf{W}$. Subsequently, if the query has m_i words it can be represented as an $m_i \times d$ matrix $\mathbf{X}_i$, likewise given the answer candidate has n_j words it is represented by an $n_j \times d$ matrix $\mathbf{Y}_j$, where the rows of each matrix correspond to the vector representations of the words appearing in the sentence. Defining the integers m and n as the maximal lengths of a query sentence and an answer sentence, variable length sentences can then be characterized by fixed-size matrices by adding zero rows to fill up empty positions for shorter sentences: the $m \times d$ matrix $\mathbf{X}_i$ denotes a query and the $n \times d$ matrix $\mathbf{Y}_j$ denotes an answer candidate.

Working with the matrix representations of sentences, the semantic relatedness of a sentence pair (q_i, a_j) can be formulated by a bi-linear model [29], given as

$$\mathbf{S}_{ij} = f\left(\mathbf{X}_i^T\mathbf{P}\mathbf{Y}_j + \mathbf{b}_s\right), \tag{4.2.1}$$

where the $d \times d$ matrix $\mathbf{P}$ and the $m \times n$ matrix $\mathbf{b}_s$ are network variables to be optimized, and $f(\cdot)$ is an activation function that operates on each element of the input matrix and can be set as hyperbolic tangent, sigmoid or rectified linear function [64, 137]. Eq. (4.2.1) computes the similarity for each pair of words, one from each sentence, and this results in the $m \times n$ similarity matrix $\mathbf{S}_{ij}$ between the query q_i and answer candidate a_j.

The parameter matrix $\mathbf{P}$ contains a total of d^2 elements. Instead of optimizing over arbitrary matrices, we can restrict the matrix structure in order to reduce the number of parameters and improve the computational efficiency—often, without sacrificing any performance. For instance, one extreme case is to fix $\mathbf{P}$ as an identity matrix. Another commonly used option is to set $\mathbf{P}$ as an diagonal matrix [5, 29, 33],

$$\mathbf{S}_{ij} = f\left(\mathbf{X}_i^T \text{diag}([p_1, p_2, \ldots, p_d])\mathbf{Y}_j + \mathbf{b}_s\right), \tag{4.2.2}$$

so that the computation is reduced to optimizing d variables instead of d^2. In this work, we follow a cross attention model [2] to restrict $\mathbf{P}$ to be rank-1, given as

$$\mathbf{S}_{ij} = f\left(\mathbf{X}_i^T \mathbf{p}_1 \mathbf{p}_2^T \mathbf{Y}_j + \mathbf{b}_s\right), \tag{4.2.3}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are two d-dimensional column vectors. One main advantage of such a design is that $\mathbf{X}_i^T\mathbf{p}_1$ projects each word in the query to a one-dimensional score, and a simple examination of the score value can provide a preliminary view on the semantic contribution of the query words, and likewise $\mathbf{Y}_j^T\mathbf{p}_2$ can be used to to indicate the contribution of the answer words. The multiplication of the scores of the query and answer words indicates a word pair that contributes significantly to the sentence relevance. This straightforward algebraic operation offers good model interpretability, with $2d$ variables to be optimized.

Pooling

To aggregate significant information and to reduce the size of the similarity representation, we apply a pooling process to the computed similarity matrix $\mathbf{S}_{ij}$. The pooling layer returns a vector version of the similarity between sentences, given as

$$\mathbf{s}_{ij}^{(l)} = \text{pooling}\left(\mathbf{S}_{ij}^{(l1)}, \mathbf{S}_{ij}^{(l2)}, \cdots, \mathbf{S}_{ij}^{(ln)}\right), \ \forall l \in [1, 2, \ldots, m], \tag{4.2.4}$$

where $\mathrm{s}_{ij}^{(l)}$ denotes the l-th element of the similarity vector s_{ij}, and $\mathrm{S}_{ij}^{(lk)}$ denotes the lk-th element of the full $m \times n$ matrix S_{ij}. The pooling operation compares the semantic similarities between a word in the query sentence and all the words in the answer candidate, and returns an aggregated similarity measure for that word. This results in a length-m similarity vector s_{ij} for each query.

The two motivations for pooling are (1) to extract influential combinations of words for text pair so that the pooled vector can be conveniently fed into the next layer, and (2) to reduce redundant information and the amount of model parameters. Commonly used pooling operations include average, max, min, and stochastic poolling [138, 139]. In this work, max pooling is applied since it can be seen as searching for the best match for each query word. For instance, consider the example in Fig. 4.2, the query is '*where is the cat?*' and its corresponding answer is '*the cat sat on the mat.*' For each word in query, we have multiple choices of words to pair with from the answer. In particular, for the word '*cat*' in the query, its possible word pairs include '*cat cat*', '*cat sat*','*cat mat*', etc., and the the pooling process would select '*cat cat*' as it has the best matching score.

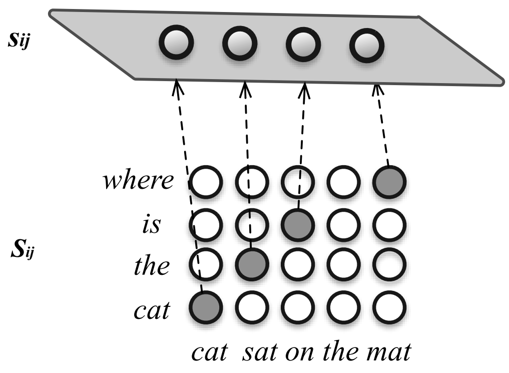


Figure 4.2: An example of the pooling process. The input query is '*where is the cat?*', an answer is '*the cat sat on the mat*'. A semantic word pair of each row in similarity matrix, which is pooled to from a vector representation after pooling. Symbol S_{ij} represents the similarity matrix between query and answer, and s_{ij} is the output from pooling process.

4.2.2 Deep Similarity Enhanced by Word Overlapping

Encoded High-level Similarity Representation

To denoise the similarity vector $\mathbf{s}_{ij}$ and remove redundant information, we incorporate multiple layers of autoencoders (AEs), referred as stacked autoencoders (SAE), to compute a robust and abstracted representation of the similarities between sentences. The network structure is illustrated in Figure 4.3. In the first AE, the input is the similarity vector $\mathbf{s}_{ij}$ returned by the pooling operation, and the encoded representation of the vector is defined by the following mapping function:

$$\mathbf{h}_{ij}^{(1)} = f\left(\mathbf{W}_1 \mathbf{s}_{ij} + \mathbf{b}_1\right). \tag{4.2.5}$$

where the $h_1 \times m$ matrix $\mathbf{W}_1$ and the h_1-length vector $\mathbf{b}_1$ define the encoder weights and bias, respectively, and $f(\cdot)$ denotes the element-wise activation function. Repeatedly, chaining the output of the $(l–1)$-th AE as the input to the l-th AE generates an abstracted representation formulated as

$$\mathbf{h}_{ij}^{(l)} = f\left(\mathbf{W}_l \mathbf{h}_{ij}^{(l-1)} + \mathbf{b}_l\right), \forall l \in [2, \ldots, H]. \tag{4.2.6}$$

Assuming a total of H encoders are employed, the weight parameters to be trained include the $h_1 \times m$ matrix $\mathbf{W}_1$, $h_l \times h_{l-1}$ matrices $\{\mathbf{W}_l\}_{l=2}^{H}$, and the h_l-dimensional bias vectors $\{\mathbf{b}_l\}_{l=1}^{H}$. Let $\boldsymbol{\theta}_l = \{\mathbf{W}_l, \mathbf{b}_l\}$ denote the complete set of parameters of the l-th AE. In general, the high-level representation $\mathbf{h}_{ij}^{(H)}$ output by SAE has the potential of capturing useful "part-whole decomposition" and "hierarchical grouping" properties [71] of the input similarity vector. The use of more encoders may capture a deeper or more robust representation of the information.

Word Overlapping Features

Previous research works have shown that it can be beneficial to utilize shallow word overlapping information between texts as a complementary information resource to further enhance the deep semantic representation [32,33]. Thus, in addition to the high-level representation $\mathbf{h}_{ij}^{(H)}$ output by the length-H SAE, we identify words shared by the query and answer candidate sentences, counting the number of

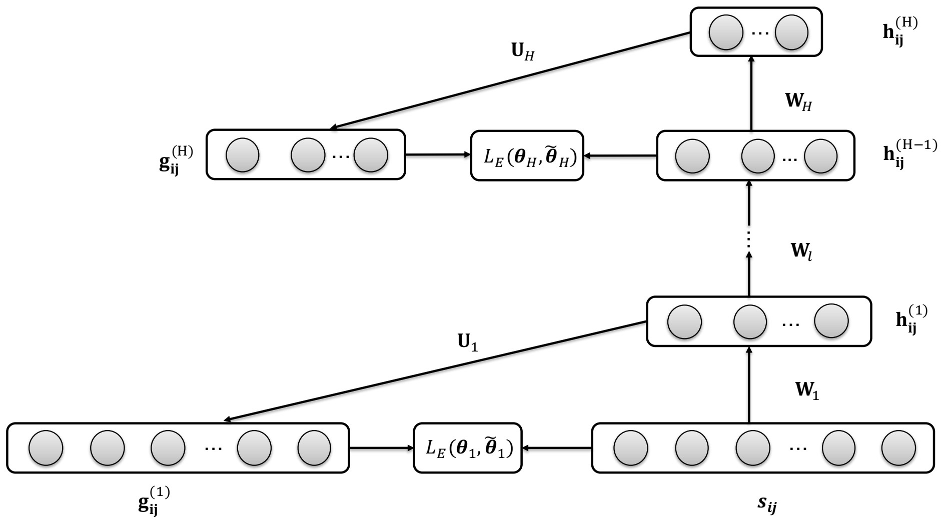


Figure 4.3: The operation structure of SAE. SAE is stacked by multiple layers (l=1,...,H) of traditional AE. Each AE encodes the input to the encoder representation, which is fed to next AE as input. The decoding part of each AE is used to reconstruct input and optimize the variables in training process.

co-occurring words, and also using their corresponding inverse document frequencies (idf) as weights: $\mathbf{h}_{ij}^{(\text{add})} = \boldsymbol{w}_{idf} \times n$. This is motivated by the loss of specific information (such as years and proper nouns) which may not be accounted for in the distributed representation of the words, but is useful when matching a query to an answer. The SAE output $\mathbf{h}_{ij}^{(H)}$ and the additional features $\mathbf{h}_{ij}^{(\text{add})}$ are concatenated into one single vector, $\mathbf{h}_{ij} = \begin{bmatrix} \mathbf{h}_{ij}^{(H)} \\ \mathbf{h}_{ij}^{(\text{add})} \end{bmatrix}$, which is used as the final similarity representation between two sentences.

4.2.3 Matching Prediction

Given a query sentence q_i, the probability that an answer candidate a_j is related to q_i can be modelled using two-way softmax based on the encoding of the similarity representation $\mathbf{h}_{ij}$, given as

$$p(t_{ij} = 1|\mathbf{h}_{ij}) = \frac{\exp\left(\mathbf{h}_{ij}^T \boldsymbol{\alpha}_1\right)}{\exp\left(\mathbf{h}_{ij}^T \boldsymbol{\alpha}_0\right) + \exp\left(\mathbf{h}_{ij}^T \boldsymbol{\alpha}_1\right)}, \tag{4.2.7}$$

where the two vectors $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ are softmax parameters with the same dimensionality as the similarity/feature vector $\mathbf{h}_{ij}$. The matching prediction task is formulated as a binary classification problem, where the label $t_{ij} = 1$ indicates that a_j is related to q_i, while $t_{ij} = 0$ otherwise.

4.2.4 Proposed Greedy Layer-wise Model Training Scheme

The training of the proposed text matching model involves the optimization of the distributed embedding matrix for the vocabulary $\mathbf{W}$, the bilinear similarity weights and biases $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{b}_s\}$, the length-$H$ SAE network encoding weights $\{\boldsymbol{\theta}_l\}_{l=1}^H$ and decoding weights $\{\tilde{\boldsymbol{\theta}}_l\}_{l=1}^H$, as well as the softmax parameters $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$. Given a collection of query and answer candidate sentences with available ground truth knowledge of whether they are related, the traditional training approach optimizes the model variables $\boldsymbol{\theta} = \left\{\mathbf{W}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{b}_s, \{\boldsymbol{\theta}_l\}_{l=1}^H, \{\tilde{\boldsymbol{\theta}}_l\}_{l=1}^H, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1\right\}$ all together by minimizing a cost function, e.g., the regularized cross-entropy cost function as shown below

$$\begin{aligned} L(\boldsymbol{\theta}) = -\sum_{(i,j)\in I} & [t_{ij} \log p(t_{ij} = k|\mathbf{h}_{ij}) \\ & +(1 - t_{ij}) \log (1 - p(t_{ij} = k|\mathbf{h}_{ij}))] + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 , \end{aligned} \tag{4.2.8}$$

where the index set I denotes the used training sentence pairs, and $\lambda > 0$ is the regularization parameter set by the user. However, this model shares many of the same challenges as other deep learning models [140], e.g., the large amount of variables to be optimized, the non-convexity and high complexity of the model, and is very difficult to obtain a good local optimal solution when following the traditional training approach. We thus propose to use a greedy layer-wise training scheme [72], which trains different sets of model variables in different stages based on multiple objective functions relevant to the task, and further fine tunes the trained parameters based on Eq. (4.2.8), with the goal of obtaining a better model solution with improved generalization ability.

Stage 1: Training the Distributed Similarity Metric based on the Ranking Score

In this stage, we aim at obtaining good initial parameters for the bilinear similarity weights $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{b}_s\}$ and the distributed word representation $\mathbf{W}$, while temporarily ignoring the remaining layers of the SAE and the softmax-based prediction. Rather than a random initialization, we initialize $\mathbf{W}$ with the embedding representation returned by a state-of-the-art unsupervised neural language model, e.g. word2Vec [141] or Glove [142], and further optimize it together with the bilinear weights. The parameters $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{b}_s, \mathbf{W}\}$ determine the quality of the computed similarity vector $\mathbf{s}_{ij}$. We evaluate $\mathbf{s}_{ij}$ based on a pairwise ranking scheme [29]. Specifically, an aggregated similarity score is computed as

$$\text{sim}(q_i, a_j) = f(\mathbf{p}_3^T \mathbf{s}_{ij} + b_p), \tag{4.2.9}$$

where $\mathbf{p}_3$ is a column vector with the same dimensionality as the distributed similarity vector $\mathbf{s}_{ij}$ and b_p is a bias parameter. Assuming a total of M query sentences are used for training, for each query, a correct answer a_{i+} and an incorrect answer a_{i-} are randomly chosen. The cost function evaluates the difference between the similarity scores of the correct and incorrect query/answer pairs. It is assumed that ranking of the correct pair is larger than the incorrect one such that sim $(q_i, a_{i+}) \geq$ sim $(q_i, a_{i-}) + \epsilon$, where $\epsilon > 0$ is the margin threshold set by the user. Denoting all the variables to be optimized by $\boldsymbol{\theta}_s = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{b}_s, b_p, \mathbf{W}\}$, the margin error function to minimize is

$$\begin{aligned} L_M(\boldsymbol{\theta}_s) = &\sum_{i=1}^{M} \max\left(0, \epsilon + \text{sim}\left(q_i, a_{i-}\right) - \text{sim}\left(q_i, a_{i+}\right)\right) \\ &+ \frac{\lambda_s}{2} \left\|\boldsymbol{\theta}_s\right\|_2^2, \end{aligned} \tag{4.2.10}$$

where $\lambda_s > 0$ is a regularization parameter set by the user. The minimization drives the differences to be large enough to facilitate the discrimination between the correct and incorrect pairs. We use $\mathbf{p}_1^{(1*)}, \mathbf{p}_2^{(1*)}, \mathbf{b}_s^{(1*)}$ and $\mathbf{W}^{(1*)}$ to denote the optimized bilinear similarity weights and the word distributed embeddings, and $\mathbf{s}_{ij}^{(1*)}$ to denote the resulting similarity vector after the stage-1 training, and $\mathbf{h}_{ij}^{(\text{add},1*)}$ the resulting additional overlapping features determined by $\mathbf{W}^{(1*)}$.

Stage 2: Training Deep Similarity based on Regularized Reconstruction Error

This stage seeks to optimize the encoder weights $\{\boldsymbol{\theta}_l\}_{l=1}^{H}$ in a layer-by-layer manner when $\mathbf{s}_{ij}^{(1*)}$ is presented as the input to the SAE. Instead of the traditional AE training that is commonly used to compute deep semantic similarity between text [19, 135], we apply a sparse and denoising AE training procedure [71], referred as Sparse Denoising Auto-encoder (SDA). It learns from a partially destructed input and attempts to reconstruct a "repaired" input from the destroyed one. Moreover, it contains a sparsity constraint on encoding representation that restricts the output magnitude of the inactive neurons to be zero. The operations of input deconstruction and sparsity enforcement enables a SDA to output more robust features that are less affected by noisy or irrelevant information in the input [72].

To implement dropout mechanism, the input similarity $\mathbf{s}_{ij}^{(1*)}$ to SAE is corrupted by a stochastic mapping approach: $f_R\left(\mathbf{s}_{ij}^{(1*)}\right) \rightarrow \tilde{\mathbf{s}}_{ij}^{(1*)}$, where f_R is the destruction function and $\tilde{\mathbf{s}}_{ij}^{(1*)}$ is the destructed input. The input destruction is implemented by randomly choosing a certain number of input elements and then fixing their value at zero, while retaining values of the other elements [72]. Each encoder is trained in conjunction with a decoder, which reconstructs the input from the encoded representation. The l-th reconstruction layer is formulated as

$$\mathbf{g}_{ij}^{(l)} = f(\mathbf{U}_l\mathbf{h}_{ij}^{(l)} + \mathbf{c}_l), \text{ for } l = 1, 2, \ldots, H, \tag{4.2.11}$$

where the $m \times h_1$ matrix $\mathbf{U}_1$, $h_{l-1} \times h_l$ matrices $\{\mathbf{U}_l\}_{l=2}^{H}$, the m-dimensional and h_{l-1}-dimensional bias vectors $\mathbf{c}_1$ and $\{\mathbf{c}_l\}_{l=2}^{H}$ store the decoder weights. Letting $\tilde{\boldsymbol{\theta}}_l = \{\mathbf{U}_l, \mathbf{c}_l\}$ denote the decoding weights, each encoder is trained by minimizing the following reconstruction error:

$$L_E\left(\boldsymbol{\theta}_1, \tilde{\boldsymbol{\theta}}_1\right) = \frac{1}{M}\sum_{j=1}^{M}\left\|\mathbf{s}_{ij}^{(1*)} - \mathbf{g}_{ij}^{(1)}\right\|^2 + \frac{\lambda_e}{2}\|\boldsymbol{\theta}_1\|_2^2 + \beta R_s(\boldsymbol{\theta}_1), \tag{4.2.12}$$

and for $l = 2, 3, \ldots, H$, we have

$$L_E\left(\boldsymbol{\theta}_l, \tilde{\boldsymbol{\theta}}_l\right) = \frac{1}{M}\sum_{j=1}^{M}\left\|\mathbf{h}_{ij}^{(l-1)} - \mathbf{g}_{ij}^{(l)}\right\|^2 + \frac{\lambda_e}{2}\|\boldsymbol{\theta}_l\|_2^2 + \beta \mathbf{R}_s^{(l)}(\boldsymbol{\theta}_l), \tag{4.2.13}$$

where $\lambda_e, \beta > 0$ are the regularization parameters, and $\mathbf{R}_s$ is the sparsity term. In practice, most of the units in the initial layer of network are inactive because of input text pairs are usually much shorter than the maximum length, resulting in a similarity vector with many trailing zeros. After the stage-2 training, we obtain the SAE weights $\left\{\boldsymbol{\theta}_l^{(2*)}\right\}_{l=1}^{H}$, the high-level similarity representation $\mathbf{h}_{ij}^{(H,2*)}$ computed from the original input $\mathrm{s}_{ij}^{(1*)}$ using the trained weights $\left\{\boldsymbol{\theta}_l^{(2*)}\right\}_{l=1}^{H}$, as well as the enhanced high-level similarity representation $\mathbf{h}_{ij}^{(2*)} = \begin{bmatrix} \mathbf{h}_{ij}^{(H,2*)} \\ \mathbf{h}_{ij}^{(\mathrm{add},1*)} \end{bmatrix}$.

Stage 3,4: Greedy Training based on Cross-Entropy

The final output of the SAE $\mathbf{h}_{ij}^{(2*)}$ is the input to the prediction layer. In the 3rd training stage, the softmax weights are adapted by minimizing the cross-entropy cost function as in Eq. (4.2.8). The adapted weights are denoted $\boldsymbol{\alpha}_0^{(3*)}$ and $\boldsymbol{\alpha}_1^{(3*)}$. Finally, the 4th training stage performs a fine-tuning [143] of the entire model, which optimizes all the model variables $\boldsymbol{\theta} = \left\{\mathbf{W}, \mathbf{p}_1, \mathbf{p}_2, \mathrm{b}_s, \{\boldsymbol{\theta}_l\}_{l=1}^{H}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1\right\}$ by minimizing the cross-entropy cost in Eq. (4.2.8), but very importantly, starting from the initial solution of

$$\boldsymbol{\theta}_0 = \left\{\mathbf{W}^{(1,*)}, \mathbf{p}_1^{(1,*)}, \mathbf{p}_2^{(1,*)}, \mathrm{b}_s^{(1,*)}, \{\boldsymbol{\theta}_l^{(2,*)}\}_{l=1}^{H}, \boldsymbol{\alpha}_0^{(3,*)}, \boldsymbol{\alpha}_1^{(3,*)}\right\}. \tag{4.2.14}$$

To train this model, we apply back-propagation to update model variables of network [144, 145]. With many parameters in the neural network model, there is the danger of overfitting on a small size of training data. To tackle the overfitting issue, we use the dropout technique along with early stopping as surrogate forms of regularization for the matching model [146, 147]. The dropout method prevents feature co-adaption by randomly removing hidden units from the neural network during forward training.

4.3 Experimental Analysis and Results

In this section, we empirically analyze and examine the design of the proposed DSSM model with its greedy layer-wise training, and compare it with multiple state-of-the-art models for solving the answer selection and reranking task using the text retrieval conference (TREC) QA dataset and Yahoo! Answer database.

Table 4.1: Data information for TREC data.

TREC Data	Questions	Correct QA pairs	Incorrect QA pairs
TRAIN-ALL	1,229	53,417	automatic
TRAIN	94	4,718	manual
DEV	82	1,148	manual
TEST	100	1,517	manual

4.3.1 Datasets and Experiment Setup

Datasets

The answer selection dataset TREC[1] is generated from TREC QA tracks 8-13, which each contain a set of factoid queries and candidate answers [148]. The correct answers for each query are manually labelled and ranked in the dataset. Three partitions of TRAIN, DEV and TEST are provided in the data. An additional set named TRAIN-ALL is provided containing 1,229 queries, with its answers determined using automatic keyword match. Thus, we can say that the TRAIN-ALL set is noisier than the other three sets: TRAIN, DEV, and TEST. Following the evaluation scheme used in previous work, two experiments are conducted in which the TRAIN and TRAIN-ALL sets are used to train separate models. In both experiments, DEV is used for model validation and TEST for reporting the answer selection performance. More detailed data information for TREC is shown in Table 4.1.

The Yahoo! answer collection is a large-scale dataset collected through Yahoo Webscope Program[2] based on community service. It includes approximately 4 million questions and answers, and each question is associated with a best answer and a category. The BM25 retrieval algorithm[3] is used to retrieve the top 100 answers for each question. These retrieved answers are also labelled as the correct ones

[1]http://trec.nist.gov/data/qa/t8qa_data.html
[2]http://webscope.sandbox.yahoo.com
[3]https://lucene.apache.org/

for each corresponding question, ranked after its best answer that is provided by the collection [12, 19]. For model training, we randomly select 10,000 questions, and five candidate answers for each question which consist of the best answer and four randomly selected incorrect answers. Two different sets each containing 5000 questions are randomly selected for model validation and evaluation.

Experimental setup

In the proposed DSSM model, the word embedding matrix **W** is initialized by the word embeddings generated by a neural language model. Specifically, for the TREC data, a skip-gram model [54] with a learning rate of 0.025 is trained using Wikipedia dumps[4] containing approximately 3 million words (after removing words that appear less than 5 times in the corpus). For the Yahoo cQA dataset, the Glove model[5] is trained using the 2B Tweets corpus containing approximately 1.2 million words after removing the infrequent words [142]. In both cases, the dimensionality of the learned word embedding vector is set as 50. For words that appear in TREC (or Yahoo) but not in Wikipedia (or Tweet), we assign random values uniformly sampled from the interval $[-0.3, 0.3]$ to each embedding dimension. The same text pre-processing procedure as in [33] is used to process all the studied text. For instance, all digits are replaced by zero, word tokenization is applied, and all the words are converted to lowercase.

To train the proposed model, we set the margin parameter as $\epsilon = 0.1$ and all the regularization parameters as $\lambda = \lambda_s = \lambda_e = 10^{-3}$ in the different cost functions. Three layers of SAE are employed, for which the hidden neuron numbers are set as $h_1 = 500$, $h_2 = 250$ and $h_3 = 100$. The sparsity control parameter is set as $\beta = 0.05$. The sparsity term $\mathbf{R}_s^{(l)} = KL\left(\rho, \|\mathbf{h}_{ij}^{(l)}\|_1/h_{(l-1)}\right)$ computes the Kullback-Libler (KL) divergence between the sparsity control parameter $0 < \rho < 1$ and the averaged activation over the hidden unit. Stochastic gradient descent is used for optimization with a mini-batch containing 50 training examples, a learning rate of 0.1, and a dropout rate of 0.5 [146]. Early stopping [149] is allowed when there is

[4]http://dumps.wikimedia.org/backup-index.html
[5]http://nlp.stanford.edu/projects/glove/

no further improvement on training performance after 30 epochs. Here, we evaluate the training performance by Mean Reciprocal Rank (MRR), see Eq. (4.3.1) below, computed using 10 mini-batches. The parameters adopted above were selected using the validation set based on a coarse manual tuning.

To report the model performance using the test set, MRR, Mean Average Precision (MAP) and Precision@N (p@N) are employed [150][6]. MRR focuses on the order of the correct answers, and is formulated as

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank@1}(i)}, \tag{4.3.1}$$

where rank@1(i) denotes the computed ranking by the model of the best correct answer in the ground truth ranking list for the i-th query, and $|Q|$ denotes the total number of queries tested. MAP accumulates the mean ranking of all the correct answers of each query, given by

$$\text{MAP} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{n_i}\sum_{\text{j}=1}^{n_i}\text{rank@j}(i), \tag{4.3.2}$$

where rank@j(i) denotes computed ranking of the j-th correct answer in the ground truth ranking list for the i-th query, and n_i denotes the number of truly correct answers for the i-th query. Being more flexible than MAP, Precision@N (p@N) calculates the mean ranking of the top N ranking answers for each query, given as

$$\text{p@}N = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{N}\sum_{j=1}^{N}\text{rank@j}(i). \tag{4.3.3}$$

Usually, N can be set as integers such as 3, 5, 10 [150].

4.3.2 Empirical Analysis of DSSM

In this section, we analyze different components of the proposed method and compare them with alternative design options appearing in existing works, to justify the effectiveness of the proposed method. Specifically, the following experiments are conducted:

[6]The tool *trec_eval* provided by Text REtrieval Conference (TREC) is used to compute MAP and MRR.

- **Experiment 1**: A version of DSSM based on the standard training scheme, which jointly optimizes all the network parameters from a random initialization, is implemented and is referred as DSSM-Tr-S. Another version of DSSM based on the proposed greedy-layer training scheme but with the final fine-tuning stage removed is implemented, referred as DSSM-Tr-GL-NoF. The goal is to demonstrate the effect of the layer-wise training based on different but relevant objective functions and the effect of fine-tuning.

- **Experiment 2**: The effect of SAE length is investigated by examining different numbers of the encoding layers, e.g., $H \in \{1, 2, 4\}$, referred as DSSM-SAE-1 when $H = 1$ for instance. These are compared to DSSM that employs $H = 3$ encoding layers.

- **Experiment 3**: We investigate the advantage of learning distributed representation for sentence pairs over that for individual sentences. An alternative text matching network is implemented by using the length-3 SAE to encode the sentence representations ($\mathbf{X}_i$ and $\mathbf{Y}_j$) instead of the similarity representation s_{ij} as DSSM does, which is referred as DeepMatchAE.

- **Experiment 4**: Alternative design choices of the parameter matrix **P** are compared, including the fixed identity matrix and a diagonal matrix as in Eq. (4.2.2), referred as DSSM-P-I and DSSM-P-D, respectively. This **P** matrix is responsible for computing the bilinear similarity in DSSM as shown in Section 4.2.1.

- **Experiment 5**: A version of DSSM based on the proposed greedy-layer training scheme but with the standard entropy loss trains the distributed similarity metric, referred as DSSM-Tr-SIM-E. To implement the training with standard entropy loss, the distributed similarity vector $\boldsymbol{s}_{ij}$ is fed to softmax function Eq.(4.2.7). Consequently, the cross-entropy cost function Eq.(4.2.8) is formulated to optimize the summarized variable $\boldsymbol{\theta}_s$. The goal is to investigate the effect of the distributed similarity metric training based on the margin based loss of the proposed model.

Table 4.2: Comparison of the proposed DSSM method with alternative designs and model settings, evaluated using the Yahoo cQA data. The best performance is highlighted in bold and second best underlined.

Experimental Settings		MRR	MAP	P@3
Exp. 1	DSSM-Tr-S	0.509	0.443	0.371
	DSSM-Tr-GL-NoF	0.536	0.492	0.412
Exp. 2	DSSM-SAE-1	0.557	0.509	0.427
	DSSM-SAE-2	0.580	0.526	<u>0.458</u>
	DSSM-SAE-4	0.572	0.518	0.451
Exp. 3	DeepMatchAE	0.531	0.473	0.402
Exp. 4	DSSM-P-I	0.550	0.493	0.418
	DSSM-P-D	<u>0.579</u>	<u>0.528</u>	0.452
Exp. 5	DSSM-Tr-SIM-E	0.568	0.525	0.450
DSSM ($H = 3$)		**0.584**	**0.532**	**0.467**

Table 4.2 summarizes performance of these compared settings, in terms of MRR, MAP and p@3.

In experiment 1, different training schemes are compared. It can be seen from Table 4.2 that the training procedure that jointly optimizes all the network parameters results in the worst performance. However, training different parts of the network in turn, according to different objective functions, leads to a performance increase of more than 3%, and further fine-tuning offers another increase over 5%. This demonstrates the advantage of greedy-layer wise training enhanced by fine tuning. In experiment 2, the number of hidden layers in the SAE is increased from 1 to 4, and it can be seen from Table 4.2 that the matching performance increases until $H = 3$ and then starts to drop slightly when $H = 4$. Based on this empirical observation, a middle depth number $H = 3$ for SAE is sufficient to achieve robust performance.

As explained in previous sections, one contribution of this work is to suggest not to compress the sentence representation at an early stage. Instead, a full matrix based sentence representation is maintained, from which we directly learn a distributed representation for a sentence pair consisting of individual similarity scores for word

Table 4.3: Demonstration of model interpretability induced by the question word score $\mathbf{X}_i^T\mathbf{p}_1$, answer word score $\mathbf{Y}_j^T\mathbf{p}_2$, and question-answer word pair score $\mathbf{X}_i^T\mathbf{p}_1\mathbf{p}_2^T\mathbf{Y}_j$. The top 5 words (or pairs) possessing the highest scores (T@5), the bottom 5 with the lowest scores (B@5), and the middle 5 that are closet to zero (0@5) are listed for two example questions each with one correct and one incorrect answer.

Example 1:

question (Q): How to create an equivalent representation for a fraction when given the decimal?

Correct answer (A_+): Read the decimal without saying point Like, 0.36 would be read as thirty-six hundredths, so 36/100.

Incorrect answer (A_-): Like one of the admission requirements for a college is 4.4 but my school goes by a 4 point scale and I have a 3.8?

	words (Q)		words (A_+)		words (A_-)		word pairs (Q-A_+)		word pairs (Q-A_-)	
	content	score ($\mathbf{X}_i^T\mathbf{p}_1$)	content	score ($\mathbf{Y}_j^T\mathbf{p}_2$)	content	score ($\mathbf{Y}_j^T\mathbf{p}_2$)	content	score ($\mathbf{X}_i^T\mathbf{p}_1\mathbf{p}_2^T\mathbf{Y}_j$)	content	score ($\mathbf{X}_i^T\mathbf{p}_1\mathbf{p}_2^T\mathbf{Y}_j$)
T@5	decimal	0.925	decimal	0.832	admission	0.239	(decimal, decimal)	0.770	(decimal, admission)	0.221
	create	0.686	hundredths	0.771	college	0.215	(decimal, hundredths)	0.713	(decimal, college)	0.199
	fraction	0.270	thirty-six	0.532	num	0.203	(create, decimal)	0.571	(decimal, num)	0.188
	representation	0.217	num	0.203	?	0.178	(create, hundredths)	0.529	(decimal, ?)	0.165
	equivalent	0.203	point	0.236	school	0.162	(decimal, thirty-six)	0.492	(create, admission)	0.164
B@5	given	-0.352	saying	-0.433	requirements	-1.297	(decimal, saying)	-0.401	(decimal, requirements)	-1.199
	when	-0.067	as	-0.352	goes	-0.968	(decimal, as)	-0.326	(decimal, goes)	-0.895
	how	-0.045	like	-0.232	scale	-0.736	(create, saying)	-0.297	(create, requirements)	-0.890
	the	2.32×10^{-4}	read	-0.081	point	-0.523	(given, decimal)	-0.293	(decimal, scale)	-0.681
	for	2.82×10^{-4}	without	-0.072	have	-0.382	(given, hundredths)	-0.271	(create, goes)	-0.664
0@5	a	1.23×10^{-4}	so	2.04×10^{-4}	and	2.06×10^{-4}	(a, so)	2.51×10^{-8}	(a, and)	2.53×10^{-8}
	an	1.67×10^{-4}	the	2.25×10^{-4}	of	2.18×10^{-4}	(a, the)	2.77×10^{-8}	(a, of)	2.68×10^{-8}
	to	2.07×10^{-4}	be	3.83×10^{-4}	the	2.25×10^{-4}	(an, so)	3.41×10^{-8}	(a, the)	2.77×10^{-8}
	the	2.32×10^{-4}	without	-0.072	for	2.46×10^{-4}	(an, the)	3.76×10^{-8}	(a, for)	3.03×10^{-8}
	for	2.82×10^{-4}	would	-0.034	is	2.55×10^{-4}	(to, so)	4.22×10^{-8}	(a, is)	3.14×10^{-8}

Example 2:

question (Q): What is the formula for magnesium chloride?

Correct answer (A_+): Magnesium has a charge of +2 and Chlorine has a charge of -1. You use the "crisscross" method where you switch the charges and so the chemical formula is MgCl2.

Incorrect answer (A_-): In reality, you don't have KCl. But you have a K+ and a Cl-. Indicating that the Potassium lost 1 electron (which is a negative charge), and the Cl gained an electron.

	words (Q)		words (A_+)		words (A_-)		word pairs (Q-A_+)		word pairs (Q-A_-)	
	content	score ($\mathbf{X}_i^T\mathbf{p}_1$)	content	score ($\mathbf{Y}_j^T\mathbf{p}_2$)	content	score ($\mathbf{Y}_j^T\mathbf{p}_2$)	content	score ($\mathbf{X}_i^T\mathbf{p}_1\mathbf{p}_2^T\mathbf{Y}_j$)	content	score ($\mathbf{X}_i^T\mathbf{p}_1\mathbf{p}_2^T\mathbf{Y}_j$)
T@5	formula	0.883	formula	0.729	electron	0.352	(formula, formula)	0.644	(formula, electron)	0.311
	magnesium	0.497	chemical	0.435	indicating	0.284	(formula, chemical)	0.384	(formula, indicating)	0.251
	?	0.178	magnesium	0.281	reality	0.217	(magnesium, formula)	0.362	(formula, reality)	0.192
	is	2.63×10^{-4}	num	0.203	negative	0.195	(formula, magnesium)	0.248	(magnesium, electron)	0.175
	for	2.82×10^{-4}	method	0.068	kcl	0.083	(magnesium, chemical)	0.216	(formula, negative)	0.172
B@5	chloride	-0.421	has	-0.396	gained	-0.625	(formula, has)	-0.350	(formula, gained)	-0.552
	what	-0.052	chlorine	-0.358	have	-0.382	(formula, chlorine)	-0.316	(formula, have)	-0.337
	the	2.32×10^{-4}	charge	-0.273	charge	-0.273	(chloride, formula)	-0.307	(magnesium, gained)	-0.311
	for	2.82×10^{-4}	switch	-0.197	lost	-0.258	(formula, charge)	-0.241	(formula, charge)	-0.241
	is	2.63×10^{-4}	where	-0.035	but	-0.157	(magnesium, has)	-0.197	(formula, lost)	-0.228
0@5	the	2.32×10^{-4}	a	1.17×10^{-4}	a	1.17×10^{-4}	(the, a)	2.71×10^{-8}	(the, a)	2.71×10^{-8}
	is	2.63×10^{-4}	so	2.04×10^{-4}	an	1.54×10^{-4}	(is, a)	3.08×10^{-8}	(is, a)	3.08×10^{-8}
	for	2.82×10^{-4}	and	2.06×10^{-4}	and	2.06×10^{-4}	(for, a)	3.30×10^{-8}	(for, a)	3.30×10^{-8}
	what	-0.052	of	2.18×10^{-4}	the	2.25×10^{-4}	(the, so)	4.73×10^{-8}	(the, an)	3.57×10^{-8}
	?	0.178	the	2.25×10^{-4}	in	2.32×10^{-4}	(is, and)	4.78×10^{-8}	(is, an)	4.05×10^{-8}

pairs. Such a model is more focused on learning the pair interaction rather than the individual representation, and thus could preserve more information between the sentence. In experiment 3, we compare this design with the traditional sentence representation learning network, which uses SAE to directly compress the sentence representation (referred as DeepMatchSAE). It can be seen from Table 4.2 that using the sentence pair interactions rather than individual sentence representation leads to a performance increase of at least 5%.

Experiment 4 compares three design options of the bilinear similarity parameter matrix $\mathbf{P}$. As seen from Table 4.2 that, by fixing it as an identity matrix, a fairly low performance is obtained due to the lack of model expressive power. Using a general diagonal matrix, for which a d-dimensional vector containing the diagonal elements is optimized, offers improved performance of an increase of around 3%. The proposed design utilizing two d-dimensional vectors to formulate the matrix $\mathbf{P} = \mathbf{p}_1\mathbf{p}_2^T$ offers another 1% performance improvement approximately as compared to the setting of diagonal matrix, but doubles the number of variables.

In experiment 5, two different training strategies under the greedy-layer wise training scheme are compared. It can be seen from Table 4.2 that the cross entropy loss trains the distributed similarity metric results in the worse performance. However, the margin-based loss evaluates the difference between the similarity scores of the correct and incorrect sentence pairs, leads to a performance increase of more than 1.5%. This improvement verifies the advantage of distributed similarity metric training enhanced by margin based loss.

Although the performance improvement is modest, the advantage of $\mathbf{P} = \mathbf{p}_1\mathbf{p}_2^T$ is that it allows more interpretation the model and its parameters (as explained in the end of Section 4.2.1) since it is possible to identify dominating words and word pairs that contribute more significantly to the relationship between two sentence based on the question word score $\mathbf{X}_i^T\mathbf{p}_1$, answer word score $\mathbf{Y}_j^T\mathbf{p}_2$, and score of the question-answer word pair $\mathbf{X}_i^T\mathbf{p}_1\mathbf{p}_2^T\mathbf{Y}_j$. Table 4.3 displays the ranked question words, answer words, and question-answer (Q-A) word pairs for the two examples. For a correct Q-A pair, word pairs possessing high values are more significant indicators to the sentence relevance. On the contrary, for an incorrect Q-A pair, word pairs possessing

Table 4.4: Performance comparison using TREC data for the proposed and seven state-of-the-art methods.

Methods	Word Overlapping	Training	MAP	MRR
[32]	no	TRAIN	0.5476	0.6437
[33]	no	TRAIN	0.6258	0.6591
DSSM	no	TRAIN	<u>0.6512</u>	<u>0.6927</u>
[32]	yes	TRAIN	0.7058	0.7800
[33]	yes	TRAIN	0.7329	0.7962
DSSM	yes	TRAIN	<u>0.7589</u>	<u>0.8051</u>
[32]	no	TRAIN-ALL	0.5693	0.6613
[33]	no	TRAIN-ALL	0.6709	0.7280
DSSM	no	TRAIN-ALL	<u>0.6813</u>	<u>0.7532</u>
[110]	NA	TRAIN-ALL	0.6091	0.6917
[108]	NA	TRAIN-ALL	0.5951	0.6951
[109]	NA	TRAIN-ALL	0.6307	0.7477
[105]	NA	TRAIN-ALL	0.6781	0.7358
[53]	NA	TRAIN-ALL	<u>0.7092</u>	<u>0.7700</u>
[32]	yes	TRAIN-ALL	0.7113	0.7846
[33]	yes	TRAIN-ALL	0.7459	0.8078
DSSM	yes	TRAIN-ALL	**0.7641**	**0.8102**

low values are more significant indicators to their irrelevance. It is interesting to observe that, words (or word pairs) possessing scores close to zero correspond to less meaningful words, e.g., “a”, “an”, “the”, “you”, etc. For correct Q-A pairs, the common function words shared by both question and answer often play an important part in identifying the sentence relevance. On the contrary, this is not necessarily the case for the incorrect Q-A pairs.

Table 4.5: Performance comparison using Yahoo cQA data for different methods.

Methods	MAP	p@3	p@5	p@10
Random Guess Baseline	0.205	0.136	0.093	0.038
WordEmbed [90]	0.345	0.290	0.245	0.181
DeepMatch [29]	0.320	0.286	0.225	0.175
DeepMatchCNN [5]	0.421	0.349	0.283	0.215
DSSM	**0.532**	**0.467**	**0.412**	**0.355**

4.3.3 Comparison with State-of-the-art Methods

Using the TREC dataset, the proposed method is compared with seven state-of-the-art approaches. For evaluation, we use the testing setup used in previous works [32, 33]: using the two different training sets, TRAIN and TRAIN-ALL, and with and without the inclusion of word overlapping features. The other existing works [53, 105, 108–110] do not involve word overlapping processing and their performance was reported by using TRAIN-ALL as the training data. The performance for these works along with the DSSM model are reported in Table 4.4. It can be seen from Table 4.4 that the proposed DSSM model with greedy-layer training provides the best performance under all the evaluation setups. The incorporation of the word overlapping features as explained in Section 4.2.2 further improves the matching performance. This is observed for not only our method but also Yu et al. [32] and Severyn et al. [33]. Among the compared existing works, Yih et al. [53] offers the best performance, however, it employs external information resource, e.g., WordNet, which unfortunately is not available for all the language—a case that could limit its usage. In general, the deep learning methods, e.g., [32, 33], perform better than the shallow ones, e.g., [108, 110]. Overall, the proposed method outperforms all the existing deep learning works.

We conduct another evaluation using Yahoo cQA dataset. The proposed method is compared with three state-of-the-art deep learning methods for this data, including WordEmbed [90], DeepMatch [29] and DeepMatchCNN [5]. The performance for compared methods along with the proposed model are shown in Table 4.5. A baseline model based on random guess is also included to provide a general view of the problem. The same performance measures MAP, p@3, p@5 and p@10 as used in existing works are adopted. It can be seen that the proposed method performs significantly better than the competing methods, demonstrating superiority of the proposed deep model design and its training strategy.

It can be seen from Table 4.5 and Table 4.4, the performance of Yahoo! community dataset is worse than the one of TREC dataset. We analyze an example where the proposed model fails to handle a short sentence pair from the Yahoo! dataset. For the question of *"names for an orange puffle?"*, the correct available answer is *"goofy, bouncy, things to describe it."* However, when tested, the proposed model opted *"well, you really can't make purple without blue and red. purple is a secondary color, created by mixing the primary colors blue and red.* as the best answer, and selecting the ground truth answer as the second best answer. In this example, the ground truth answer contains less interactive information to response the informally formulated question, thus the proposed model selects a longer answer sentence that contains rich information.

4.4 Conclusion

In this work, we have proposed a novel deep learning architecture for representing the semantic similarity between question and answer pairs in cQA. This is done by passing a whole sentence worth of word-level similarities, computed using a distributed model, to the DAE network with an unsupervised training. The standard bilinear function matches the sentence pair with a weight matrix, as a common tool applied in the previous works. To examine the effect of word-level similarity formulation for QA matching, we designed different types of bilinear matching functions and compared them by using ranking measurements e.g. MAP and MRR. The results report that the matching function with parameter constraints offers the best performance among all comparison functions.

Furthermore, the proposed model benefits from its design of being focused on learning directly the distributed representation of sentence pair interaction rather than learning individual sentence representation, according to compare the performance between two learning presentation approaches. Regard to the training approach, the proposed model adopts a pair-wise training method to combine the unsupervised training for the DAE network and supervised training for semantic matching. Finally, the model is supported by an effective greedy layer-wise training strategy to fine-tune the entire model, and this decomposable training approach is experimentally shown to outperform the previous models.

Although we have investigated to improve the distributed similarity representation by the proposed model. Nevertheless, there are still exist a number of drawbacks which could be improved. For instance, only pairwise similarities between words are used, but sequences of words may correspond to phrases that require a compositional representations of words. Therefore, in the next section, we aim to explore architectures that can utilize by multiple word-level and word-sequence similarities.

Chapter 5

Context-aware Neural Network for Interactive Matching

5.1 Introduction

QA is the task of enabling a machine to automatically answer questions posted by humans in a natural language form. The selection of the best answer from an existing pool of candidate answers is referred to as cQA [99], whereas enabling the computer to automatically generate a novel answer, through some natural language model, is known as machine dialogue [7,49]. In this work, we focus on cQA by working on the semantic matching between question and answer texts. In general, semantic matching requires the accurate modeling of the relevance between two portions of text, and, in addition to QA, is widely used for tasks, such as paraphrase identification [31,35], machine translation [61,75,94], and image caption generation [27,98].

In order to compute an accurate measure of relevance between the sentence pair, it is beneficial to take the lexical, syntactic and semantic information of the text pairs into account. Traditional matching seeks effective ways of extracting semantic features that improve a given similarity metric [52]. Recent advances have managed to replace this manual feature engineering process with a model that automatically learns distributed representations of words and sentences via neural networks [31,33,56].

As previously mentioned, the goal of a QA matching task is to select the correct answers from a set of candidate answers based on the content of a given question. One of the key hindrances in this, is that the key lexical components and information might not be shared between question and answer texts. In some cases, ambiguous contents in questions or answers may impede this process. For instance, consider the question-answer scenario given in Fig. 5.1. Regards to the object “cat” of query, A_1 provides more distinct keywords in answer than A_2. When focusing on fixed keywords in the question text, such as “cat” and “where”, both answers contain information that matches these keywords, e.g., “cat” and “in the park” in A_1, and also “cat” and “on the mat” in A_2. This simple keyword-based matching strategy, hence, becomes a limitation on machine-based decision making.

Question Q_1

- Where was the cat?

Candidate Answers: A_1 and A_2

- A_1: I saw the cat before, I think it is in the park now.
- A_2: It is left on the mat, in a room wit a cat.

Figure 5.1: Example scenario 1 for QA based on key-word matching.

However, if the focus of the question can be varied according to the context of the answer, e.g., by paying more attention to “see a cat” instead of “cat” and “where”, the machine can then be directed to avoid the answer A_2. Hence, the question-answer matching process can become more effective when the sentence representations for questions and answers are learned jointly, other than in isolation. This interactive learning has been exploited in the previous work [120] using a hybrid model that includes a bi-directional long short-term memory (LSTM) model and a convolutional neural network (CNN). The model incorporates cross-sentence context based on word-level representations. More specifically, the question text is fed as an input to produce a hidden representation of the question text, which is then used to generate the answer representation. As such, the representation of the answer captures the information contained in the question text.

Question Q_2

- Where was the cat?

Candidate Answers: A_{21} and A_{22}

- A_{21}: The cat was sitting on a mat.
- A_{22}: We had a dog that was friendly to our cat.

Question Q_3

- What is the color of that cat?

Candidate Answers: A_{31} and A_{32}

- A_{31}: The cat was sitting on a mat.
- A_{32}: The cat that was sitting on the red mat.

Figure 5.2: Example scenario 2 with two different QA cases.

Answer	**Key Components**
A_{21}	The ***cat*** was sitting ***on a mat***
A_{22}	We had a dog that was friendly to our ***cat***

Figure 5.3: Key components of potential answers to the Question Q_2.

Past research on cQA [95, 120, 121] has shown that it is important to model the content interaction between the question and answer sentences to improve the performance of a cQA system. Matching performance can be compromised when collecting information by examining the question and answer sentences individually and separately. Although interaction between the question and answer sentences can be formulated as a similarity accumulation over word pairs parameterized by weight variables (e.g., [56]), the resulting model can be inflexible. This is because, when converting the discovery of the content interaction between the question and answer sentences to an optimization of the weight variables, fixed contributing patterns of word positions for discriminating the matching question-answer pairs are assumed.

Consider another QA scenario shown in Fig. 5.2, where has two question examples, each with their own pool of answers. We highlight the key components for each

Answer	**Key Components**
A_{31}	The ***cat*** was sitting on a mat.
A_{32}	The ***cat*** that was sitting on the ***red*** mat.

Figure 5.4: Key components of potential answers to the Question Q_3.

of the answers in Figs. 5.3 and 5.4. In both examples, these salient components in the answers directly reflect or respond to the context of the questions, which contribute more significantly towards the relevance of the given question. Such salient information or the key components in sentences can be captured by an attractive approach called attention mechanism [75]. This mechanism has been mostly used in the tasks related to translations [75, 151]. It utilizes a weight function to quantify the importance of the hidden sentence representation at the corresponding word position. Recent findings, for example in [120, 152], have demonstrated their applicability in assigning degrees of importance, known as attention weights, to different word positions in a sentence. In the QA task, it has been of increasing interest to develop effective ways of building attention mechanisms. Previous works [127, 153] have adopted self-attention mechanism to build the connections inside a sentence. Several works [40, 95] have proposed co-attention mechanism to collect the interactive information that involves a sentence representation to the targeted sentence hidden representation, so that the matching performance can be improved by learning from and paying more attention to salient text.

In this work, we aim at improving the modeling of the question-answer interaction in representation learning through investigating effective ways of modeling the involved input. We know that, given different questions, it is natural for a human to pay attention to different parts of the answer sentence. For instance, when reading “a white cat is sitting on the tree", we pay more attention to “white" knowing the question is “what is the colour of the cat", while more attention to “on the tree" if the question is changed to “where is the cat". In this example, there also exist words that are naturally less informative, e.g., “a" and “the" as compared to “white", “cat" and “sitting". And this is not affected by the question content. Therefore, high attention weights should be selectively assigned to more informative word positions in the answer. To automatically identify non-informative words in a sentence and take this into account in attention weight assignment, we propose a new quantity referred to as the context information jump indicator. It captures the informativeness of a word by representing the across joint representation between adjacent words based on pre-trained language model. By including the proposed quantity as part of input, the importance of a word position in an answer sentence is affected not only by the answer and question content that is relevant to the matching task, but also its own informativeness independent of the matching.

In this section, we address the aspects that have been highlighted above, particularly on the interactive learning of the question and answer representations and attention mechanism design, and propose a novel approach to improve the standard of the response accuracy during the cQA process. In particular, we make the following key contributions:

1. We extend the notion of interactive learning by developing a cross-sentence context-aware bi-directional LSTM model, where we generate the hidden representations for both the question and answer texts, thereby making them aware of each other's context. As such, in the proposed model, the hidden representation for the answer text, and particularly the state values for each word position, is affected not only by its previous or next states, but also by the multi-positional representations of the question text.

2. As the interaction between question and answer texts is bi-directional, the content of the question text should also affect the way that the answer text is encoded or characterized. We propose interaction-based and sentence-based two attention parallel mechanisms for sentence representation learning, and augment our proposed approach to consider the relationship between adjacent words, instead of concatenating the word representations to formulate co-attention weights as in previous works [40, 123].

3. A new quantity in co-attention mechanism, referred to as the context information jump, is proposed to represent the aggregation representation between

forward and backward states based on the bi-directional LSTM. Context jump is able to modify the question for every words in answer, vice versa.

4. We perform an exhaustive evaluation of the proposed approach using four community datasets, namely TREC, Yahoo! and StackEx(L) and WikiQA, and share our findings.

The remaining sections is organized as follows: In Section 5.2, we review the operation of LSTM. Then we discuss the proposed method in Section 5.3, explaining the generative bi-directional-interaction model with the context information jump. This is followed by a detailed discussion on the evaluation process in Section 5.4, and results from evaluation in Section 5.5. We finally conclude the work in Section 5.6.

5.2 Preliminaries

A commonly used strategy for selecting from a candidate answer pool a sentence that matches the given question, is to first compute the representations, e.g., in the form of vectors or matrices, for the question and answer sentences based on their word content. Similarity (or relevance confidence) scores between the question and the candidate answers are then computed using their corresponding representations, and the candidate with the highest score is selected.

We denote a sentence as $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$ where x_t is the t-th word in the sentence. An RNN-based language model learns a vector representation to encode the semantic and order information of the words in the sentence. This is typically expressed as

$$\boldsymbol{h}_t = f(\boldsymbol{w}_t, \boldsymbol{h}_{t-1}), \tag{5.2.1}$$

where the t-th word x_t corresponds to a hidden state at time step t, and $\boldsymbol{w}_t$ denotes a vector representation for encoding the semantics of the word x_t. The hidden representation vector $\boldsymbol{h}_t$ contains word context information accumulated up to the t-th word in the sentence. It is computed from the vector representation $\boldsymbol{w}_t$ of the current word and the previous accumulation $\boldsymbol{h}_{t-1}$. The different realizations of the activation function $f(\cdot)$ result in different types of RNNs. For instance, a classical

RNN employs a standard linear operation with a sigmoid activation $\text{sig}(\cdot)$ to process the input $\boldsymbol{w}_t$ and $\boldsymbol{h}_{t-1}$. Differently, an LSTM uses a set of recurrent functions [154] by following defined as

$$\begin{aligned}
\boldsymbol{i}_t &= \text{sig}\left(\mathbf{W}_{xi}\boldsymbol{w}_t + \mathbf{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right), && (5.2.2)\\
\boldsymbol{f}_t &= \text{sig}\left(\mathbf{W}_{xf}\boldsymbol{w}_t + \mathbf{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right), && (5.2.3)\\
\boldsymbol{o}_t &= \text{sig}\left(\mathbf{W}_{xo}\boldsymbol{w}_t + \mathbf{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right), && (5.2.4)\\
\boldsymbol{g}_t &= \tanh\left(\mathbf{W}_{xc}\boldsymbol{w}_t + \mathbf{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_g\right), && (5.2.5)\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t, && (5.2.6)\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right), && (5.2.7)
\end{aligned}$$

where $\odot$ denotes the Hadamard product. The word vector $\boldsymbol{w}_t$, as well as the weight matrices $\mathbf{W}$ and the bias vectors $\boldsymbol{b}$ with different subscript symbols, are the model variables to be optimized.

To enrich the sentence representation, a bi-directional LSTM architecture can be used [75]. Specifically, one LSTM is used to process the input sentence as a sequence of words in the forward direction, of which the computed hidden representation at the t-th word position is denoted by the vector $\boldsymbol{h}_{t,f}$ (all vectors in this manuscript are considered column ones). A different LSTM processes the input sentence in the reverse direction, and the learned hidden representation is denoted by $\boldsymbol{h}_{t,b}$. Combining both, an extended hidden sentence representation at each word position is given as $\boldsymbol{h}_t = \left[\boldsymbol{h}_{t,f}^{\top}, \boldsymbol{h}_{t,b}^{\top}\right]^{\top}$, and is referred to as the positional sentence representation at the t-th word [56]. Working with the two sets of sentence positional representations $\{\boldsymbol{h}_t^{(q)}\}_{t=1}^{T}$ and $\{\boldsymbol{h}_t^{(a)}\}_{t=1}^{T}$, various strategies [56, 120, 122] are developed to compute their similarity or relevance confidence scores (we use the indicator symbols "q" and "a" to distinguish a question sentence from an answer sentence). The model used in this proposed work is described in sub-section 5.3.4.

5.3 Proposed Method

To summarize, the proposed cQA system contains a cross-sentence context-aware bi-directional LSTM referred to as CABIN model illustrated in Fig.5.5 . The proposed

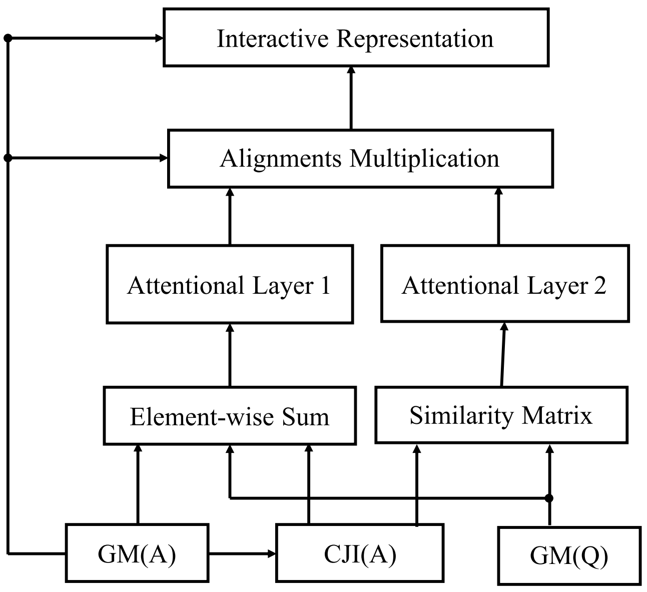


Figure 5.5: Architecture of the proposed CABIN system for computing interactive sentence representations. GM(A) symbol represents the pre-trained answer representation from the generative language model; GM(Q) symbol is the pre-trained question representation from the generative language model; CJI(A) symbol means the context information jump vector of the answer sentence.

model is built upon an improved modeling strategy of the question-answer interaction, containing three key components: (1) the pre-trained language model benefits the proposed method, (2) the attention-driven interactive sentence-aware representation enhanced by context information jump, and (3) the distributed similarity computation. In the following sections, we describe the proposed system in detail.

5.3.1 Co-attention Sentences Mechanism

A common method formulate the self-attention function $A(\boldsymbol{h}_t, \mathbf{X})$ for each positional answer sentence representation is defined as

$$A(\boldsymbol{h}_t, \mathbf{X}) = \tanh\left(\boldsymbol{u}^T \boldsymbol{h}_t + \boldsymbol{v}^T \left(\tfrac{1}{T}\sum_{t=1}^{T} \boldsymbol{h}_t\right)\right), \tag{5.3.1}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the model parameters to be optimized. The sentence content is encoded by its averaged positional representations, given as $\mathbf{X} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{h}_t$. Each positional answer representation and the sentence content jointly control values of the attention weights.

In this work, we propose a new parallel and interactive attention mechanism with its architecture to compute the answer sentence representation as example illustrated in Fig.5.6. Here, we introduce the computation of the attention formulation $A(\cdot,\cdot)$, the attention formulation $E(\cdot,\cdot)$ would be represented in section 5.3.2. Two additional quantities $\tilde{\boldsymbol{h}}_T^{(q)}$ and $\nabla\tilde{\boldsymbol{h}}_t^{(a)}$ are included in $\mathbf{X}$ when formulating the attention function of answer, given as

$$A(\tilde{\boldsymbol{h}}_t^{(a)},\mathbf{X}) = \tanh\left(\boldsymbol{u}_a^T\tilde{\boldsymbol{h}}_t^{(a)} + \boldsymbol{v}_a^T\tilde{\boldsymbol{h}}_T^{(q)} + \boldsymbol{g}_a^T\nabla\tilde{\boldsymbol{h}}_t^{(a)}\right), \tag{5.3.2}$$

the two quantities $\tilde{\boldsymbol{h}}_T^{(a)}$, $\nabla\tilde{\boldsymbol{h}}_t^{(q)}$ are used for formulating the attention of question by

$$A(\tilde{\boldsymbol{h}}_t^{(q)},\mathbf{X}) = \tanh\left(\boldsymbol{u}_q^T\tilde{\boldsymbol{h}}_t^{(q)} + \boldsymbol{v}_q^T\tilde{\boldsymbol{h}}_T^{(a)} + \boldsymbol{g}_q^T\nabla\tilde{\boldsymbol{h}}_t^{(q)}\right), \tag{5.3.3}$$

where the representation vectors $\tilde{\boldsymbol{h}}_T^{(q)}$, $\tilde{\boldsymbol{h}}_T^{(a)}$ encode the content information of the question and answer sentence. Different from existing approaches, they are learned in an unsupervised way by following a sentence generation model. The pre-trained vectors $\tilde{\boldsymbol{h}}_T$ effectively reduce the computational complexity. Moreover, the probabilistic language model is an effective approach to encode semantic information carried by sentences. The vector $\nabla\tilde{\boldsymbol{h}}_t$ is the proposed jump quantity, and the vector $\boldsymbol{g}$ is the model variable associated with this quantity.

Generative Sentence Content Representation

The vector $\tilde{\boldsymbol{h}}_T$ corresponds to the final-state representation of a sentence, which is returned by pre-training a bi-directional LSTM. It is learned in an unsupervised way, by letting this LSTM operate as a generative model to solve a sentence generation task. Here, we use the symbol "~" to distinguish it from the notation $\boldsymbol{h}_T$ of Section 5.2, which also denotes the final-state representation vector of a question returned by a bi-directional LSTM, but trained in a supervised manner tailored to the cQA matching task. We now first describe the unsupervised training of $\tilde{\boldsymbol{h}}_T$ and then explain its advantages.

Taking a corpus containing question sentences only, a bi-directional LSTM is trained by maximizing the log-likelihood of generating these sentences. Following the probabilistic language model [28], we formulate the probability of generating a sentence $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$ as

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} \frac{\exp\left(\mathbf{W}(x_t,:)\tilde{\boldsymbol{h}}_t + \boldsymbol{b}(x_t)\right)}{\sum_{i=1}^{T}\exp\left(\mathbf{W}(x_i,:)\tilde{\boldsymbol{h}}_i + \boldsymbol{b}(x_i)\right)}, \tag{5.3.4}$$

where the weight matrix $\mathbf{W}$ and the bias vector $\boldsymbol{b}$ are the model variables to be optimized. The row number of $\mathbf{W}$ and the length of $\boldsymbol{b}$ are equal to the number of words in the question vocabulary list. The operations $\mathbf{W}(x,:)$ and $\boldsymbol{b}(x)$ extract the row in $\mathbf{W}$ and the element in $\boldsymbol{b}$ that correspond to the input word x. An optimizer stochastic gradient descent method is used to optimize the model by following the same process as the work in [28].

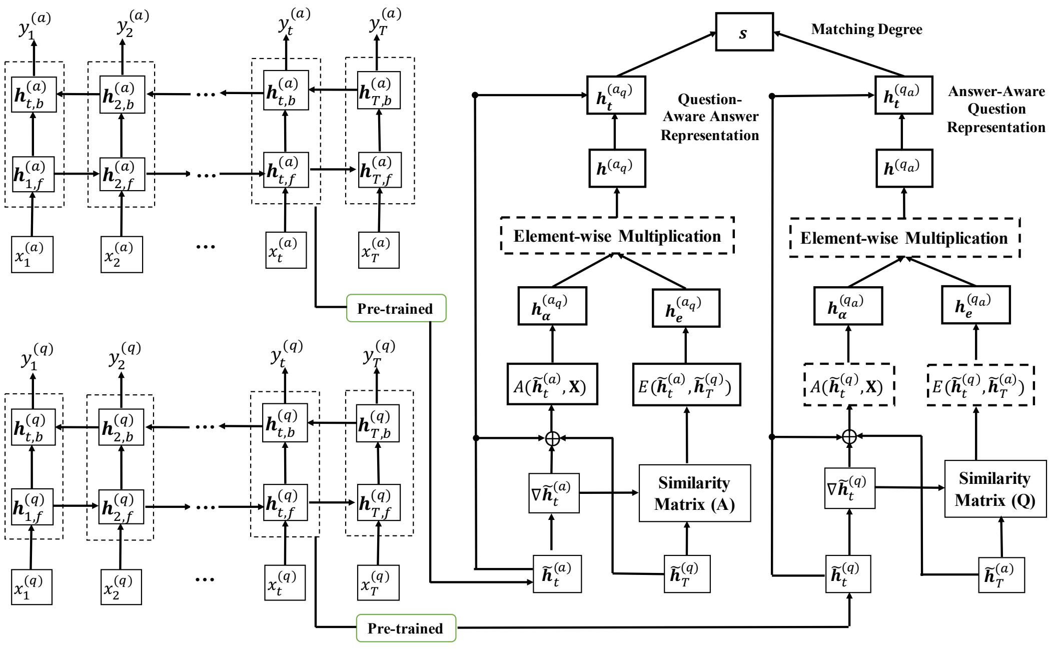
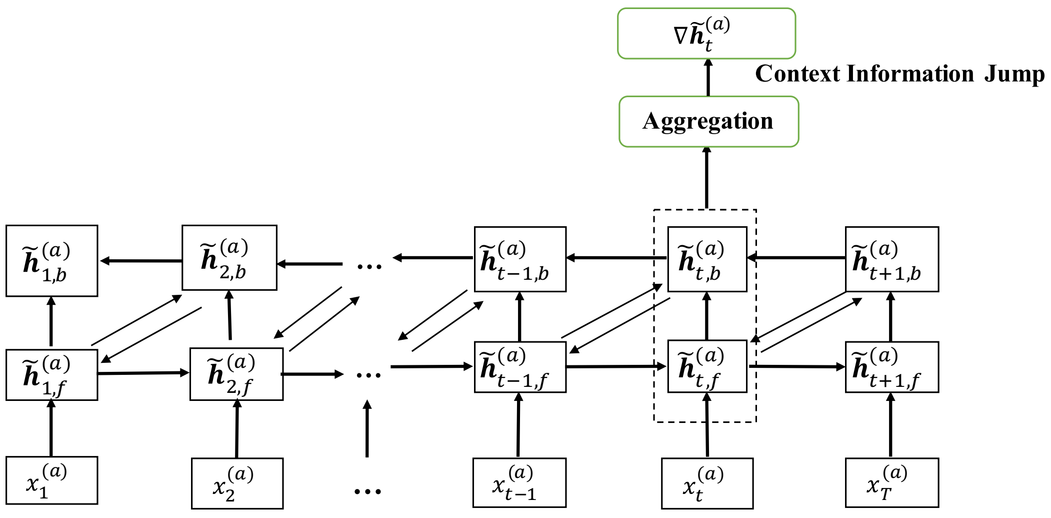

Matching Degree
s
Question-Aware Answer Representation
Answer-Aware Question Representation
Element-wise Multiplication
Similarity Matrix (A)
Similarity Matrix (Q)
Pre-trained
Pre-trained

Figure 5.6: Architecture of the attention mechanisms for computing question-aware answer representations in the proposed CABIN system.

The question representation $\tilde{\boldsymbol{h}}_T^{(q)}$ and the answer representation $\tilde{\boldsymbol{h}}_T^{(a)}$, computed separately from the answer and question representations, acts as a fixed input to the attention function. As compared to Eq.(5.3.1) that requires simultaneous optimization of $\{\boldsymbol{h}_t\}_{t=1}^{T}$ together with $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{h}_t$, the pre-trained $\tilde{\boldsymbol{h}}_T$ effectively reduces the



$\nabla\tilde{\boldsymbol{h}}_t^{(a)}$
Context Information Jump
Aggregation
$\tilde{\boldsymbol{h}}_{1,b}^{(a)}$
$\tilde{\boldsymbol{h}}_{2,b}^{(a)}$
$\tilde{\boldsymbol{h}}_{t-1,b}^{(a)}$
$\tilde{\boldsymbol{h}}_{t,b}^{(a)}$
$\tilde{\boldsymbol{h}}_{t+1,b}^{(a)}$
$\tilde{\boldsymbol{h}}_{1,f}^{(a)}$
$\tilde{\boldsymbol{h}}_{2,f}^{(a)}$
$\tilde{\boldsymbol{h}}_{t-1,f}^{(a)}$
$\tilde{\boldsymbol{h}}_{t,f}^{(a)}$
$\tilde{\boldsymbol{h}}_{t+1,f}^{(a)}$
$x_1^{(a)}$
$x_2^{(a)}$
$x_{t-1}^{(a)}$
$x_t^{(a)}$
$x_T^{(a)}$

Figure 5.7: Architecture of the bi-directional LSTM with context information jump in the proposed CABIN system.

computational complexity. Moreover, the probabilistic language model is an effective approach to encode semantic information carried by sentences. The pre-trained question an answer representations are learned from bi-directional LSTM as the fixed input of the proposed matching model. We will show later in the result Section 5.5 that the proposed model offers competitive performance and the use of pre-trained $\tilde{\boldsymbol{h}}_T$ enhances the matching accuracy.

Context Information Jump

When learning sentence representations by a bi-directional LSTM, each obtained positional representation accumulates context information up to the targeted word position within a sentence in forward and backward directions. It is reasonable to assume that if the previous and next words bring significant change to the sentence semantics and content, it can directly affect the importance degree of the positional representation at the current word. Such a change in sentence semantics could be indicated by the information change contained by the learned hidden representations between the current and adjacent states. Therefore, given a sentence, we aim to formulate a quantity $\nabla\tilde{\boldsymbol{h}}_t$ that can be potentially used as an indicator of its information change between the current (t), the previous ($t-1$) and the next ($t+1$) word positions.

In the common technique of bi-directional LSTM, the positional word representation is affected by the neighboring word in a single direction during the propagation of bi-directional LSTM [75]. Here, we design a positional word state depends on the novel combination of current forward state and backward state. It is reasonable to assume the next state brings the context information to the current forward state. In a similar way, the previous state also enriches the current backward state. Thus, we explore the strategy to compute combined representation at current word position by involving the next state in backward direction and previous state in forward direction

$$\nabla\tilde{\boldsymbol{h}}_t^{(a)} = \begin{bmatrix} \tilde{\boldsymbol{h}}_{t,f}^{(a)} \odot \tilde{\boldsymbol{h}}_{t+1,b}^{(a)} \\ \tilde{\boldsymbol{h}}_{t,b}^{(a)} \odot \tilde{\boldsymbol{h}}_{t-1,f}^{(a)} \end{bmatrix}, \tag{5.3.5}$$

where $\odot$ is the Hadamard product [155], known as element wise product of two vectors. The matrix symbol $[:,:]$ aggregates the hidden states to a dimensional vector. We compute the alignment representation which is a good indicator of similarity between question and answer sentences. Because the quantity $\nabla\tilde{\boldsymbol{h}}_t^{(a)}$ of answer sentence is used as an indicator of the degree that new information is conveyed by the previous and next word between two adjacent states of an answer sentence, we refer to it as context information jump. Fig.5.7 illustrates the working operation of context information jump. Its role is to relate the salience of an sentence word position to the informativeness of this word given its adjacent ancestor word. By computing this quantity using a generative language model independent of the particular cQA task, general language patterns in sentence text can be captured. Using the same process for question sentence, we obtain the quantity $\nabla\tilde{\boldsymbol{h}}_t^{(q)}$.

5.3.2 Positional Word-Sentence Level Similarity

It is known that the semantic relativeness is a key component to determine the similarity between the question and answer sentence. In [1], they computed the similarity matrix between two sentences and applied it to compute the attention alignment representation. Inspired by this work, we design an adaptive similarity matrix to explore the importance of positional word in answer/question sentence for corresponding question/answer sentence. To achieve this, we compute the similarity between the positional word in answer and the question sentence, and vice versa. Specifically, we use pre-trained bi-directional LSTM model to solve the same sentence generation task as in Section 5.3.1. This results in a set of learned positional representations for the sentence, denoted by $\{\tilde{\boldsymbol{h}}_t\}_{t=1}^T$. By treating the final state of question sentence and the current state of answer sentence as the inputs into matching function $E(\tilde{\boldsymbol{h}}_t^{(a)}, \tilde{\boldsymbol{h}}_T^{(q)})$, given as

$$E(\tilde{\boldsymbol{h}}_t^{(a)}, \tilde{\boldsymbol{h}}_T^{(q)}) = \tanh\left(\boldsymbol{q}_1^{(a)}(\nabla \tilde{\boldsymbol{h}}_t^{(a)}(\tilde{\boldsymbol{h}}_T^{(q)})^T)\right), \tag{5.3.6}$$

where the weight $\boldsymbol{q}_1^{(a)}$ is a vector. The vectors $\tilde{\boldsymbol{h}}_{t,f}^{(a)}, \tilde{\boldsymbol{h}}_{t,b}^{(a)}$ indicate the pre-identifying hidden representation in t-th word in the forward and backward direction, separately. The output of matching function is a similarity vector representing the contextual relation between each word in target answer sentence and the question sentence. We employ the matching function to define the similarity-based attention weighted value $\boldsymbol{e}_t^{(a)}$ of question-aware answer as

$$e_t^{(a_q)} = \frac{\exp\left(\boldsymbol{q}_2^{(a)} E(\tilde{\boldsymbol{h}}_t^{(a)}, \tilde{\boldsymbol{h}}_T^{(q)})^T\right)}{\sum_{i=1}^T \exp\left(\boldsymbol{q}_2^{(a)} E(\tilde{\boldsymbol{h}}_i^{(a)}, \tilde{\boldsymbol{h}}_T^{(q)})^T\right)}. \tag{5.3.7}$$

The variable vector $\boldsymbol{q}_2^{(a)}$ transfers the similarity vector to a matching score. The attention weight is computed based on the content from the similarity matrix between the answer word and question sentence. In addition to the attention weight in Eq.(5.3.8), this weight is also used to compute the answer representation in the next section. In a similar way, the similarity-based attention weight of answer-aware question $e_t^{(q_a)}$ could be computed.

5.3.3 Interactive sentence Representation

A method for modeling the interaction between two sentences is through the co-attention mechanism [56]. It utilizes a weight function to quantify the importance of the hidden sentence representation at the word position t. By incorporating the proposed attention formulation of Eq.(5.3.2) into Eq.(5.3.8), an importance weight between 0 and 1 is learned for each positional representation of the answer sentence

$\tilde{\boldsymbol{h}}_t^{(a)}$, given as

$$\alpha_t^{(a_q)} = \frac{\exp\left(A(\tilde{\boldsymbol{h}}_t^{(a)}, \mathbf{X})\right)}{\sum_{i=1}^{T} \exp\left(A(\tilde{\boldsymbol{h}}_i^{(a)}, \mathbf{X})\right)}, \tag{5.3.8}$$

where $\mathbf{X}$ stores the additional information that affects the importance of the targeted word position, and attention function $A(\cdot, \cdot)$ is computed in Eq.(5.3.2). Because the attention weight is affected by the question content and the importance of answer word, we adopt the notations of $\alpha_t^{(a_q)}$ and $e_t^{(a_q)}$ for each weight separately. The following alignment representation vectors are used to compute the two types of answer representation

$$\boldsymbol{h}_\alpha^{(a_q)} = \sum_{t=1}^{T} \alpha_t^{(a_q)} \tilde{\boldsymbol{h}}_t^{(a)}, \tag{5.3.9}$$

with

$$\boldsymbol{h}_e^{(a_q)} = \sum_{t=1}^{T} e_t^{(a_q)} \tilde{\boldsymbol{h}}_t^{(a)}. \tag{5.3.10}$$

This parallel weighted formulations encode information carried by each positional answer representation, and is weighted by an importance score that is affected by the question content, and also the positional representation and the word informativeness at the targeted word position. By combining these two alignment vectors, we compute the fused attention representation of answer sentence by

$$\boldsymbol{h}^{(a_q)} = \boldsymbol{h}_\alpha^{(a_q)} \odot \boldsymbol{h}_e^{(a_q)}. \tag{5.3.11}$$

To compute an adaptive answer sentence representation to the question content, we aggregate the pre-trained positional representation of answer sentence and the weighted representation, is defined as

$$\boldsymbol{h}_t^{(a_q)} = \tanh\left(\mathbf{V}_a(\tilde{\boldsymbol{h}}_t^{(a)} \odot \boldsymbol{h}^{(a_q)}) + \boldsymbol{b}_a\right), \tag{5.3.12}$$

where the weight matrices $\mathbf{V}_a$ and the bias vector $\boldsymbol{b}_a$ are model variables to be optimized. We denote each positional question representation computed with this modified architecture as $\boldsymbol{h}_t^{(a_q)}$, where $t = 1, 2, \ldots, T$. The averaged alignment vector representation is used as the final state representation $\boldsymbol{h}_T^{(a_q)} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{h}_t^{(a_q)}$, which refers to question-aware answer representation vector. The answer-aware question representation $\boldsymbol{h}_T^{(q_a)} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{h}_t^{(q_a)}$, where the combined state $\boldsymbol{h}_t^{(q_a)}$ is computed from the formulation Eq.(5.3.12) for question sentence.

5.3.4 Model Training and Initialization

So far, we have explained the computation of the question-aware answer representation vector $\boldsymbol{h}_T^{(a_q)}$ and the answer-aware question representation vector $\boldsymbol{h}_T^{(q_a)}$. Taking these two vectors as input, we formulate the following similarity vector to encode the distributed matching degree between the question and answer sentences

$$\boldsymbol{s} = \tanh\left(\mathbf{U}_q \boldsymbol{h}_T^{(q_a)} + \mathbf{U}_a \boldsymbol{h}_T^{(a_q)} + \boldsymbol{b}_s\right), \tag{5.3.13}$$

where the weight matrices $\mathbf{U}_q$, $\mathbf{U}_a$ and bias vector $\boldsymbol{b}_s$ are the model variables to be optimized. Subsequently, the sentence matching task can be formulated as a binary classification problem. The label $y = 1$ indicates that the answer is related to the question, while $y = 0$ otherwise. The probability that an answer is related to a question can be modeled using a two-way softmax function, such as

$$p(y = 1|\boldsymbol{s}) = \frac{\exp\left(\boldsymbol{s}^T \boldsymbol{\alpha}_1\right)}{\exp\left(\boldsymbol{s}^T \boldsymbol{\alpha}_0\right) + \exp\left(\boldsymbol{s}^T \boldsymbol{\alpha}_1\right)}, \tag{5.3.14}$$

where the two column vectors $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ are softmax parameters with the same dimensionality as $\boldsymbol{s}$. Based on the above formulation, model variables can be optimized by minimizing a regularized cross-entropy cost by following the logistic regression model [140, 156].

Here, we summarize the training process of the system. First, unsupervised pre-training of two individual bi-directional LSTM models are performed using the question sentences and answer sentences separately. Both models are trained to solve the language generation task via log-likelihood maximization, based on the sentence generation probabilities as formulated by Eq.(5.3.4). Sentence representations learned by these two models, e.g., $\tilde{\boldsymbol{h}}_T^{(q)}$ and $\{\tilde{\boldsymbol{h}}_t^{(a)}\}_{t=1}^T$, are used as the fixed input of the proposed matching model. Then, the matching model is trained to solve a binary classification problem by minimizing the regularized cross-entropy cost, based on the probability of observing a positive sentence pair as formulated in Eq.(5.3.14). Instead of random initialization, we initialize all the distributed word representation vectors with Glove word embeddings [142]. The bi-directional LSTM used for computing the question and answer representations are initialized by the pre-trained bi-directional

Table 5.1: Dataset content statistics in CABIN model.

Parameter	TREC	Yahoo!	StackEx(L)	WikiQA
No. of Questions	1,505	90,000	6,939	3,047
No. of Answers	60,800	4.5M	8,595	29,258
Mean Question Length(words)	11.39	9.73	136.03	7.26
Mean Answer Length(words)	24.63	99.38	217.61	24.94

LSTM model. The remaining variables are initialized randomly.

5.4 Experimental Analysis and Results

In this section, we evaluate the proposed model CABIN against a number of state-of-the-art models using four key cQA datasets. In this section, we present our evaluation methodology.

5.4.1 Datasets

We relied on four key cQA datasets for our evaluation, namely TREC[1] [148], Yahoo![2] [19], Stack-Exchange-Legal[3] (StackEx(L)), and WikiQA[4] [13]. We give a summary of the statistics related to these datasets in Table 5.1.

5.4.2 Performance Metrics

To report model performance using the test set, we use three performance metrics, namely mean reciprocal rank (MRR), mean average precision (MAP) and the mean ranking of the top-N answers, denoted by MRT$_N$ or p_N, as in [150]. The MRR metric

[1]http://`trec.nist.gov/data/qa/t8qa_data.html`

[2]http://`webscope.sandbox.yahoo.com`

[3]https://`law.stackexchange.com/`

[4]https://`aka.ms/WikiQA`

Table 5.2: Benchmark data splits.

Data Set	**Q/A Pairs**	**Development**	**Training**	**Testing**
TREC [32]	8,997	1,148	4,718	1,517
Yahoo! [19]	4M	2,500	50,000	25,000
StackEx(L) [21]	7,760	1,500	4,760	1,500
WikiQA [13]	29,258	2,733	20,360	6,165

focuses on the order of the correct answers, and is formulated as

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{r_i^1}, \tag{5.4.1}$$

where r_i^j denotes the computed ranking of the j-th correct answer in the ground truth ranking list for the i-th query, and $|Q|$ denotes the total number of queries tested. In other words, with $j = 1$, r_i^1 denotes the best possible answer. MAP accumulates the mean ranking of all the correct answers in each query, expressed as

$$\text{MAP} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{n_i}\sum_{j=1}^{n_i}\frac{n_i^j}{r_i^j}, \tag{5.4.2}$$

where r_i^j is the computed ranking of the j-th correct answer from the ground truth ranking list for the i-th query, n_i^j is the number of truly correct answers in the computed ranking list of the j-th correct answer, and n_i denotes the number of truly correct answers for the i-th query.

5.4.3 Experimental Configuration

Experimental platform and recordings: All the training and testing were carried on a system with 36 physical cores, 128GB RAM, three graphical processing units (GPUS) each equipped with 12GB RAM, and running the version of the Tensor Flow Framework (v1.3).

Neural network configurations: The bi-directional LSTM architecture used in our studies contains 100-dimensional hidden sentence representations. The dimensionality of each word embedding vector is set as 300.

Training preparation and initialization: In preparing the dataset for training and testing, we followed the same text pre-processing procedures described in [33]. More specifically, a special end-of-sentence symbol $\langle$_EOS$\rangle$ is added to the end of each sentence, and the out-of-vocabulary words are mapped to a special token symbol $\langle$_UNK$\rangle$. Wherever the sentence lengths fall below the minimum threshold, a special symbol, $\langle$_PAD$\rangle$, is added to the end of the sentence, so as to pad them with extra characters to meet the processing requirements. Furthermore, the basic pre-training model is Glove [142] using a corpus containing 6B words from Wikipedia and Gigaword. For words appearing in each dataset, but not in their training corpus, a random value uniformly sampled from the interval of $[-0.3, 0.3]$ is assigned to each embedding dimension. A normal distribution $\mathcal{N}(0, 0.1)$ is used for model variables initialization.
Training / testing process: For model optimization, a root mean square propagation (RMSProp) algorithm is used. The process includes a mini-batch containing 50 training examples, a learning rate of 0.1, and a dropout rate of 0.5 [146]. The learning rate is halved after 10 epochs. Gradient clipping [157] is used to scale the gradient when the norm of gradient exceeds a threshold of five. The overall datasets have been split for training, testing and development purposes as suggested by the original datasets [13, 19, 21, 32]. These are given in Table 5.2.

5.4.4 Baselines

To compare with the proposed method, the following ten models, stemming from the space of CNN, RNN and conventional/traditional techniques, are considered.
Baseline Models:

1. Random Guess (RandomGuess) [56]: A random ranking list for the test samples without training process.

2. Bag of Words (BoW) [5]: Each sequence of words is represented by the idf-weighted sum of the embeddings of the words it contains, and concatenated before feeding them as input to the network; for instance a multilayer perceptron (MLP).

3. Word Embedding (WordEmbed) [90]: This model uses the Glove tool to obtain the word embedding representation of a sentence. The matching score of two short-texts are calculated with an MLP, taking the embeddings of the two sentences as input.

CNN-based Models:

4. Bigram-CNN [32]: This model produces a sentence representation by feeding the adjacent words to a convolution layer, and then measures the similarity of the generated sentence representations through an MLP.

5. Add-CNN [33]: The model is an enhanced version of the Bigram-CNN model. It uses CNNs to produce the representations individually, and then calculates the matching score with an MLP.

6. AP-CNN [96]: It convolves each word embedding representation of the sentences, the output matrices from the convolution layer use the max pooling function with attention mechanism to learn the sentence representations.

7. Ab-CNN [37]: The model matches the feature maps of phase-level based sentences from the convolution layer to generate an attention matrix. It learns the high-level sentence representations as inputs to the convolutional layer, which is used to calculate the matching similarity.

8. CAM [2]: A recent work proposes the model performs different comparison matching functions to match the sentences based on word-level, where the similarity outputs from the function are aggregated into a vector by a convolution layer. The convolved vector as the input into the final prediction layer to compute the matching score.

RNN-based Models:

9. QA-LSTM [120]: Given two sentences, they are encoded by a bi-directional LSTM with a word-to-word attention mechanism, where the output from the model is fed to a convolution layer for producing the sentence representation.

10. IARNN [40]: The model learns an answer sentence representation using an attention mechanism to involve a question hidden representation from an RNN network, which then generates a high-level answer sentence representation as the input to the RNN network.

11. BiMPM [119]: The model encodes two sentences with a bi-directional LSTM, the encoded outputs of a sentence match each hidden representation of the other sentence in two directions. The sequences of matching vectors are aggregated into a vector as an input to prediction layer.

12. IWAN [1]: It builds an alignment layer based on a word-level similarity martix for computing attention weight of each word, where the similarity martix is computed by the sentence encoded outputs from a bi-directional LSTM.

For the purpose of evaluation, we collect the reported results from the published works of above mentioned models, wherever possible. Wherever this was not feasible, we implemented them to match with the reported specification and experimental evaluation of these models.

5.5 Results and Analysis

5.5.1 Quantitative Evaluation

Comparison with State of the Art Methods

We first compare the performance of our proposed approach against a number of techniques using the metrics mentioned in Section 5.4. Table 5.3 reports the MRR and MAP metrics for different models evaluated using the four datasets mentioned above. Overall, the proposed model CABIN performs best in most cases, and is always amongst the top three performing models. In Table 5.4, we summarize the model ranking, where, for instance, the best performing model possesses the ranking of 1.0, while the worst possesses the ranking of 14.0. For each model, we report its averaged ranking over the two measures for each dataset, and the last column of the table reports the final averaged ranking over all the datasets. It can be seen

Table 5.3: Performance comparison of different models across a range of datasets. The best results are highlighted and the second best results are underlined.

	TREC		WikiQA		Yahoo!		StackEx	
Models	MRR	MAP	MRR	MAP	MRR	MAP	MRR	MAP
RandomGuess [56]	0.5731	0.4920	0.4620	0.4582	0.4173	0.3759	0.4897	0.4350
BoW [5]	0.6810	0.5842	0.5411	0.5330	0.5021	0.4735	0.5842	0.5362
WordEmbed [90]	0.7052	0.6091	0.5630	0.5521	0.5273	0.4911	0.6053	0.5578
Bigram-CNN [32]	0.7846	0.7113	0.6415	0.6311	0.5952	0.5730	0.6821	0.6491
Add-CNN [33]	0.8078	0.7459	0.6652	0.6520	0.6150	0.5722	0.7067	0.6525
QA-LSTM [120]	0.8322	0.7111	0.7045	0.6821	0.6468	0.6157	0.7409	0.7158
AP-CNN [96]	0.8511	0.7530	0.6957	0.6886	0.6489	0.6047	0.7325	0.6830
Ab-CNN [37]	0.8539	0.7741	0.7108	0.6921	0.6530	0.6325	0.7461	0.7205
KV-MemNNs [128]	0.8523	0.7857	0.7265	0.7069	0.6749	0.6431	0.7580	0.7365
IARNN [40]	0.8208	0.7369	0.7418	0.7341	0.6687	0.6275	0.7489	0.7175
BiMPM [119]	0.8750	0.8020	0.7310	0.718	0.6892	0.6353	0.7523	0.7240
IWAN [1]	**0.8890**	0.8220	0.7500	0.7330	0.7010	0.6521	0.7689	0.7341
CAM [2]	0.8659	0.8145	0.7545	0.7433	0.7035	0.6630	0.7852	0.7483
CABIN	0.8845	**0.8375**	**0.7653**	**0.7520**	**0.7250**	**0.6825**	**0.8024**	**0.7656**
CABIN-J	0.8563	0.7925	0.7415	0.7242	0.7026	0.6510	0.7812	0.7341
CABIN-A	0.8450	0.7843	0.7323	0.7135	0.6937	0.6442	0.7684	0.7150
CABIN-P	0.8559	0.7962	0.7320	0.7150	0.6883	0.6320	0.7763	0.7212
CABIN-S	0.8621	0.8207	0.7468	0.7292	0.7094	0.6641	0.7891	0.7469

Table 5.4: Averaged ranking of different models. The best results are highlighted in bold and the second best are underlined.

Models	TREC	Yahoo!	StackEx	WikiQA	Overall
RandomGuess [56]	14.0	14.0	14.0	14.0	14.0
BoW [5]	13.0	13.0	13.0	13.0	13.0
WordEmbed [90]	12.0	12.0	12.0	12.0	12.0
Bigram-CNN [32]	10.5	11.0	10.5	11.0	10.8
Add-CNN [33]	9.0	10.0	10.5	10.0	9.9
QA-LSTM [120]	9.5	8.5	8.5	6.5	8.3
AP-CNN [96]	7.0	8.5	8.5	9.0	8.3
Ab-CNN [37]	5.5	7.0	6.5	5.0	6.0
KV-MemNNs [128]	5.5	6.0	4.5	3.5	4.9
IARNN [40]	9.0	3.5	4.5	6.5	5.9
BiMPM [119]	<u>3.5</u>	5.0	4.5	5.0	4.5
IWAN [1]	**1.5**	3.5	3.0	3.5	2.9
CAM [2]	<u>3.5</u>	<u>2.0</u>	<u>2.0</u>	<u>2.0</u>	<u>2.4</u>
CABIN (Proposed)	**1.5**	**1.0**	**1.0**	**1.0**	**1.1**

from Table 5.4 that the proposed model possesses the highest ranking among all the compared ones.

In the following, we make a number of more specific observations from Table 5.3:

- With respect to the MRR, where a higher value indicates better performance, the proposed approach outperforms all models when evaluated against the WikiQA, Yahoo! and StackEx(L) datasets. In particular, the proposed outperforms the next best performing model, which is CAM model, by 1.33%, 2.15%, and 1.72% respectively, on these datasets.

- When considering the MAP performance, the proposed approach outperforms the CAM model, when compared against the WikiQA, Yahoo! and StackEx(L) datasets, by 0.87%, 1.95%, and 1.73% respectively.

- On the TREC dataset, the proposed approach offers the best MAP performance, followed by the 2nd best IWAN model providing close performance. The proposed model beats the IWAN model by 1.55% in MAP performance.

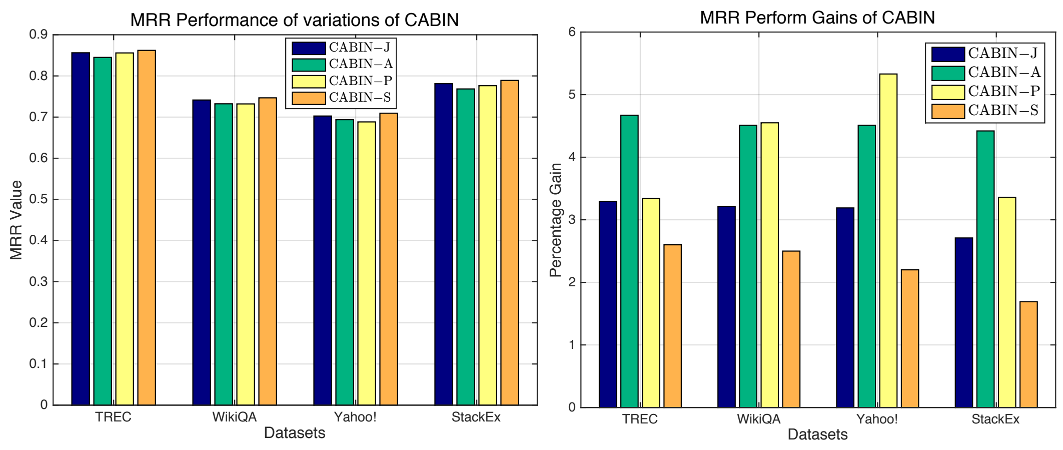
- The MRR performance of the proposed on the TREC dataset, however, are not as good as would be expected. The best IWAN model outperforms the proposed approach by 0.6%.

When comparing both MRR and MAP performance over the four datasets, the proposed model achieves the best results on TREC dataset, conversely, the worst results on Yahoo! dataset. Upon a closer inspection of the different datasets, we observe that there is a noticeable difference in mean lengths for questions and answers between the Yahoo! and other datasets. Also, the Yahoo! dataset contains questions and answers that are more informally formulated or expressed in a colloquial way, and this is particularly the case when compared against the TREC, StackEx(L) and WikiQA datasets. For example, in the Yahoo! dataset, it is common to see a question sentence like *"What Subbed episode does Nel transform???? @ Bobbi: Cause i saw it on youtube and yeah i just wanted to know, Thank you :-)"*, and a matching answer like *"Hmmm...bleach episode 192!!!!!!!!!! heres the list of the episodes lol:... GOOD LUCK!"*. Albeit being trivial, such informal formations of question-answer pairs render the cQA problem more difficult to handle.

In comparison, the three other datasets describe non-trivial, but well formulated question-answer pairs with long sentences. Both the proposed attention mechanism and the context information jump are developed to capture and encode information flow in sentences based on word semantics and order information. As such, the proposed model can better be exploited on the TREC, StackEx(L) and WikiQA datasets containing better formulated and longer question and answer sentences. Thus, it still has the challenge to solve the colloquial sentences matching in cQA datasets such as Yahoo! dataset.

Empirical Analysis of CABIN model

To understand the performance behavior of our proposed CABIN in detail, and to verify the model varieties against our hypothesis, we trained and tested the proposed model under the four different conditions: without attention mechanism (CABIN-A), without context jump (CABIN-J), without pre-training process (CABIN-P) and with standard self-attention function Eq.(5.3.1) in attention mechanism (CABIN-S). These evaluations enable the relative merits of the attention and context jump mechanisms



(a) (b)

Figure 5.8: Left figure (a): Absolute performance and right figure (b): Performance gains of the proposed approach.

and pre-training to be quantified over the proposed version. To assess the absolute advantage over the proposed version, we define the percentage gain on MRR as:

$$G_{MRR}(\mathbf{x}) = \frac{MRR_{(\text{CABIN})} - MRR_{\mathbf{x}}}{MRR_{\mathbf{x}}} \tag{5.5.1}$$

where $\mathbf{x} \in \{(\text{CABIN-A}), (\text{CABIN-J}), (\text{CABIN-P}), (\text{CABIN-S})\}$. Corresponding MRR performance and gains are shown in Figure 5.8. A number of observations can be made:

- When considering the absolute MRR performance (Figure 5.8(a)), the pre-training, the attention, and the context information jump mechanisms always improve a certain value in the MRR performance. Thus proposed model performance against the other varieties of the model.

- When considering the TREC and the StackEx(L) datasets (Figure 5.8(b)), the biggest contribution comes from the attention mechanism. The gain values of the two datasets are 4.67% and 4.42% among the proposed configurations.

- When considering the WikiQA and the Yahoo! datasets, the pre-training process contributes to the highest gain in MRR performance, by 4.55% and 5.33%, respectively.

- The above observation is, for all datasets, where the jump mechanism produces a stable and similar gains around 3% in MRR performance. In particular, the jump mechanism brings the biggest increasement on TREC and WikiQA datasets, by 3.29% and 3.21%, respectively.
- Comparing with the MRR performance between proposed co-attention and self-attention, the proposed attention offers an increased gains around 2% for all datasets. The MRR perform gains provide the biggest improvement on TREC and WikiQA datasets, by 2.6% and 2.5%, respectively.

Overall, the above results show that the attention mechanism leads to a better performance to the highly contentable and structured type of the TREC and StackEx(L) datasets with less training samples. On the other hand, the pre-processing process is an efficient tool on the Yahoo! and WikiQA datasets with more training samples. The context jump mechanism considers the effect of adjacent text information leads to a better performance on highly structured and grammatically correct nature of the TREC and WikiQA datasets. This observation, to a certain extent, verifies the hypothesis that well-phrased English sentences are predictable.

5.5.2 Example Demonstration

To illustrate the efficacy of the proposed approach in a qualitative manner, we present a number of sample question-answer cases, from the best performing TREC and worst performing Yahoo! datasets. In both the cases, we show the top-three possible answers picked up by two different architectures. The two different, yet compatible, architectures are the IWAN model for TREC dataset, and CAM model for Yahoo! dataset, which are the second best performing models on the two datasets individually, and hence chosen as a comparative model.

First consider the Example 1 and Example 2 from the TREC and Yahoo! datasets, presented in Tables 5.5 and 5.6. It can be observed that the true answers are correctly identified by the proposed model. Also, the ranked answers from the proposed model are more accurate than the ones from the IWAN model.

In addition to these two successful examples, we now consider an example where our model fails to handle an informally formulated question-answer pair from the Yahoo! dataset. For the question of *"ahh help, what is a really scary pea my pants scary story? I want it to be soo scary, Thank you :)"*, the correct available answer is *"oh god, man now that is really scary you – your pants from reading a scurry story lol... XD Hope it helps, X"*. However, when tested, the proposed model opted *"if you don't take a test, you'll continue to be scared. you should really just take it. just remember, if you're stressed and scared, your period can be late. it's best to just take a test to know for sure."* as the best answer, and selecting the ground truth answer as the second best answer. In this example, the ground truth answer contains informal language, the proposed model could not encode such information accurately and selects a longer sentence, which is more formally formulated, as the best answer.

To illustrate the efficiency of the proposed attention mechanism, we illustrate the salient word positions highlighted by the question-aware answer attention weights $\alpha_t^{(a_q)}$, for two example question-answer pairs from the TREC and Yahoo! datasets, in Table 5.7. Attention weights learned by the proposed and the existing attention mechanisms are reported for each pair. It can be seen from Table 5.7, that the proposed method is able to capture more accurately the salient word positions, which are important for the matching task.

To examine the efficiency of context jump mechanism in proposed model, we demonstrate corresponding similarities between an question and its context information jump using two example questions from the TREC datasetin, shown in Table 5.8. In the table, the word positions possessing the two largest context information jump and the two smallest context information jump are marked and indicated by T@k and B@k, respectively, for $k = 1, 2$. For each example question, a correct answer and an incorrect one are examined. It is interesting to observe that the T@k words are generally more informative than the B@k words.

For the same two example questions, we also illustrate the difference of the selected salient answer words and the top three retrieved answer sentences, between our two model versions CABIN-J and CABIN in Table 5.9. This is to demonstrate the effect of the proposed quantity of context information jump in attention learning and sentence matching. It can be seen from Table 5.9, that the inclusion of the proposed quantity results in more accurate answer retrieval and salient word identification for both example questions.

5.6 Conclusion

In this section, we have proposed the cQA matching model CABIN, which is based on a cross-sentence context-aware bi-directional LSTM architecture. The goal is to improve the semantic matching between query and answer sentences, and this is achieved by exploring three aspects: contextual information between adjacent words in a sentence, an adaptive attention mechanism and the generative sentence representation by pre-processing based bi-directional LSTM. Thereby, we examine and analyze these specific skills are benefit for cQA matching.

A novel pair-wise attention mechanism is designed to produce the interactive sentence representation, the first co-attention based on the sentence content, and the second interactive attention depends on the similarities between question and answer. In particular, we augment the existing techniques, which mainly use positional question and answer representations, with word frequency and co-occurrence information in order to improve the computation of attention weights. Further contributions of this work, include the context information jump and the use of a generative sentence representation. The former helps improving the computation of attention weights by considering informativeness of different word positions, whereas the latter eases the computation without sacrificing the representation quality.

Furthermore, to take into account adjacent context in sentence representation, the bi-directional LSTM learning representation is not only based on the simple previous or the next states in one direction propagation, but also on the use of the cross states of the sentence in hand. This results in a context-aware inside the sentence representation, which is self-adaptive to the sentence content.

Overall, we evaluated the proposed model with the aid of four datasets, using a number of metrics and against a considerably large number of models from the literature including state-of-the-art ones. Our results indicate that the proposed attention mechanism, the proposed quantity of context information jump and the generated sentence representation can help to improve the question answer matching on a certain extent for different situations of datasets. The empirical results also show that the proposed co-attention function improves matching performance for all datasets by comparing with the self-attention function. Although further evaluations may be needed to differentiate the benefits on well-written text, our results indicate the proposed method is a very useful technique to improve the cQA process.

Table 5.5: Comparison of the top three answers returned by the proposed CABIN and existing IWAN [1] architectures for an example question from the TREC dataset, where the ground truth answer sentences are marked by (∗) in the end.

Example 1 (TREC Dataset)	
Question	Who is the president or chief executive of Amtrak?
Top 3 answers by CABIN	**No.1**: " long-term success here has to do with doing it right, getting it right and increasing market share," said george warrington, amtrak 's president and chief executive. (∗)
	No.2: " amtrak is committed to treating all employees fairly," amtrak president george warrington said in a statement. (∗)
	No.3: amtrak is also upgrading the tracks between washington and boston, said warrington, which should lead to improved service even before the high-speed trains are introduced.
Top 3 answers by IWAN [1]	**No.1**: amtrak will lose money again this year, but will meet the congressional deadline of weaning itself from operating subsidies by the fiscal year ending sept. 30 , 2002, officials said.
	No.2: " amtrak is committed to treating all employees fairly," amtrak president george warrington said in a statement. (∗)
	No.3: amtrak is offering a deal it hopes few travelers can resist: get good service or a free ride.

Table 5.6: Comparison of the top three answers returned by the proposed CABIN and existing CAM [2] architectures for an example question from the Yahoo! dataset, where the ground truth answer sentences are marked by (∗) in the end.

Example 2 (Yahoo! Dataset)	
Question	how to push yourself to the limit during excercising?
Top 3 answers by CABIN	**No.1**: try wearing a bandana and looking really cool and maybe you can "push yourself to the limit" in a top gun kind of way. listen to some bon jovy music. (∗)
	No.2: the answer to your question is no not necessarily. you probably are suffering from a subluxation of the lumabr spine.
	No.3: you have to break in a composite bat, which is what rolling it does. it's just like hitting a few hundred times. it works just fine, but the bats pop will probably die out sooner. but you will hit the ball harder and further.
Top 3 answers by CAM [2]	**No.1**: the red one is a shiny one meaning its rarer if i were you i would go for the red. but blue is good too, that the only difference is its colour.
	No.2: "squidward you like crabby patties don't you!?"
	No.3: try wearing a bandana and looking really cool and maybe you can "push yourself to the limit" in a top gun kind of way. listen to some bon jovy music. (∗)

Table 5.7: Comparison of the top three salient word positions in answer captured by the proposed CABIN and the second best models using two examples from the TREC and Yahoo! datasets. The learned attention weight is reported in parenthesis for each selected salient word.

Example 1 (TREC Dataset)	
Question	Who is the president or chief executive of Amtrak?
CABIN	“long-term success here has to do with doing it right , getting it right and increasing market share , " said ***george*** (0.0751) warrington, ***amtrak***’s (0.0825) president and ***chief*** (0.0613) executive.
IWAN [1]	***amtrak*** (0.0612) will lose money again this year, but will ***meet*** (0.0469) the congressional deadline of weaning itself from operating subsidies by the fiscal year ending sept. 30, 2002, ***officials*** (0.0625) said.
Example 2 (Yahoo! Dataset)	
Question	how to push yourself to the limit during excercising?
CABIN	try wearing a bandana and looking really cool and maybe you can “***push*** (0.0754) yourself to the ***limit*** (0.0627)” in a top gun kind of way. ***listen*** (0.0516) to some bon jovy music.
CAM [2]	the red one is a shiny one meaning its rarer if i were ***you*** (0.0632) i would ***go*** (0.0562) for the red. but blue is ***good*** (0.0415) too, that the only difference is its colour.

Table 5.8: Illustration of answer word positions with either the largest two similarity values of the context information jump $\nabla\tilde{\boldsymbol{h}}_t^{(a)}$ indicated by T@K for K=1,2 (highlighted in bold), or the smallest two similarity values of $\nabla\tilde{\boldsymbol{h}}_t^{(a)}$ indicated by B@K for K=1,2 (underlined). We use **Q**, $\mathbf{A}_+$ and $\mathbf{A}_-$ to distinguish the question, correct answer and incorrect answer sentences.

Example 1	
Q: what is eileen marie collins' occupation ?	
$\mathbf{A}_+$: selected <u>*by*</u> (B@1) nasa <u>*in*</u> (B@2) January 1990, ***collins*** (T@1) became an ***astronaut*** (T@2) in July 1991.	$\mathbf{A}_-$: also, is she by any chance <u>*from*</u> (B@1) the ***daughter*** (T@2) <u>*of*</u> (B@2) michael collins, one of the ***apollo*** (T@1) 11 astronauts?
Example 2	
Q: what is the religious affiliation of the kurds ?	
$\mathbf{A}_+$: most ***kurds*** (T@2) are secular muslims who belong <u>*to*</u> (B@1) <u>*the*</u> (B@2) main ***sunni*** (T@1) sect.	$\mathbf{A}_-$: about 2 million kurds ***live*** (T@1) in northeastern syria near its border with ***turkey*** (T@2) and iraq, but <u>*the*</u> (B@1) kurdish military presence there centered mainly <u>*around*</u> (B@2) kurds from iraq, not turkey.

Table 5.9: Comparison of the top three answers and salient word positions returned by the two versions of CABIN-J and CABIN corresponding to ones with and without using the context information jump. The same two example questions as in Table 5.8 are examined, where the ground truth answer sentences are marked by (∗) in the end.

Example 1	
Question	what is eileen marie collins' occupation ?
Top 3 answers by CABIN	**No.1**: ***selected*** by ***nasa*** in January 1990, ***collins*** became an ***astronaut*** in July 1991. (∗)
	No.2: the five-member ***crew*** of the shuttle ***columbia*** that will launch chandra is led by veteran astronaut eileen ***collins*** , who would become the first woman of any nation to command a ***spaceflight***. (∗)
	No.3: also, is she by any ***chance*** from the ***daughter*** of michael ***collins***, one of the ***apollo*** 11 astronauts?
Top 3 answers by CABIN-J	**No.1**: also, is ***she*** by any chance from the daughter of michael ***collins***, one of the ***apollo*** 11 ***astronauts***?
	No.2: the five-member crew of the shuttle ***columbia*** that will launch chandra is led by veteran ***astronaut*** eileen ***collins***, who would become the first woman of any nation to command a ***spaceflight***. (∗)
	No.3: ***selected*** by nasa in January 1990, ***collins became*** an astronaut in ***July*** 1991. (∗)
Example 2	
Question	what is the religious affiliation of the kurds ?
Top 3 answers by CABIN	**No.1**: most ***kurds*** are secular ***muslims*** who ***belong*** to the main ***sunni*** sect. (∗)
	No.2: now his capture ***gives ocalan*** the ***stature*** among other ***kurds*** he never had before.
	No.3: about 2 million kurds ***live*** in northeastern syria near its ***border*** with ***turkey*** and iraq, but the kurdish military presence there centered mainly around ***kurds*** from iraq, not turkey.
Top 3 answers by CABIN-J	**No.1**: about 2 million kurds live in northeastern syria near its ***border*** with ***turkey*** and iraq, but the kurdish military presence there centered mainly around ***kurds*** from ***iraq***, not turkey.
	No.2: most ***kurds*** are secular ***muslims*** who ***belong*** to the ***main*** sunni sect. (∗)
	No.3: now his ***capture*** gives ***ocalan*** the ***stature*** among other ***kurds*** he never had before.

Chapter 6

Attentive Memory Network For Answer Selection

6.1 Introduction

QA is a challenging matching problem that is required to understand the unstructured context and generate an answer for given inputs, not only selecting the correct answers from candidate pools with a pure QA matching in answer selection task. The text understanding task is refereed to MRC, which also is important part as well as the answer selection in NLP tasks. The semantic matching as an efficient approach accurately measuring the relevance between two portions of text and is widely used in various NLP tasks, such as paraphrase identification [31, 35], machine translation [34, 61], image caption generation [27, 98], sentiment analysis [6, 36], etc.

In QA, the semantic matching is required to find the relationships between question-answer (Q-A) or question-context (Q-C) pairs, whatever for MRC or answer selection task. It is verified by the MRC example is shown in Figure 6.1. Given the question and context, the generated answer consists of a segment text “Khartoum killed Gordon”, which is a span of context. Since there appears a number of word “Gorden” may disturb correct answer generation for given question, it is vital to find the word-level semantic information between the question and context interaction, especially for the multiple-words generation.

Question Q_1

- Which British general was killed at Khartoum in 1885?

Candidate Context:

- In February 1885 Gordon returned to the Sudan to evacuate Egyptian forces. Khartoum came under siege the next month and rebels broke into the city. Khartoum killed Gordon and the other defenders. The British public reacted to his death by acclaiming 'Gordon of Khartoum', a saint. However, historians have suggested that Gordon defied orders and refused to evacuate...

Answer Prediction:

- Khartoum killed Gordon

Figure 6.1: Example scenario for MRC. With respect to given question and context, the spans answer is labeled by underline,

The traditional approach is query-based IR techniques [3] searches a text segment from corpus to support QA matching. It is hard to capture the semantic relevance between given sentence pair, and derive a useful contextual representation. An alternative knowledge-based QA approach [8, 10] can be used to inform the matching about the relationship between words or phrases, however, this requires additional effort to manually create or curate a knowledge resource, and still requires a mechanism to combine the word or phrase-level similarity. Thereby these works are not able to provide sufficient information to improve semantic matching.

Replacing the manual feature engineering with a neural language model, memory network-based model [58, 59], a type of neural networks with addressable memory mechanism, has been recently proposed to measure and develop the progress of MRC architecture. The main usage of the memory network is to store the intermediate representations of the given context and return the meaningful information to support answer prediction. Unlike the generic recurrent networks, since the memory network is required to connect with the external resources from the input processing controller, the number of parameters increase exponentially with the capacity size of memory. In order to capture context information with fewer training parameters, several memory networks [14, 15] have designed the memory as a vector that involves the representation of sentence contexts, this behavior may result in memory recording only a small amount of external resources, e.g. portions of relevant content in inputs.

With regards to the memory storage limitation problem, our proposed model utilized the memory matrix instead of vector to store entire semantic similarities based on word-level for the interaction between question and context. Although previous work NTM also designs the memory matrix for word recognition task, a large size of memory network may lead to a burden on the training. Notably, considering the effect of parameters, we make the memory network stores the information in separately two steps at one iteration, without feeding the whole information to memory matrix once. More precisely, the memory matrix stores information based on individual rows and columns vectors. The row-based memory first stores the contextual information and refines a new memory matrix, which is used in the column-based memory as the second step. Besides, the attention mechanism is used to improve the interactive content in the memory, it is helpful for generating multiple words with weakly unsupervised training. We refer to the proposed approach as multi-dimensional memory network model (MMN).

The context is composed of a set of sentences. In MRC task, it has enough content information to memory. QA matching in answer selection without supported context and the given input sentences in the answer selection dataset normally contain a few useful words to perform semantic word-level similarities. Previous works [8, 44] adopt knowledge-based resources to provide external information to support semantic matching. Whereas it costs an expensive computation by collecting the sentences from a web or large corpus. Therefore, the proposed model collects the relevant Q-A pairs under the same category from the internal dataset corpus instead of external knowledge-based resources to compute the distributed similarity matrix between pair-wise words, where the similarity matrix contains relevant word pairs are used to initialize memory network for answer selection. An attentive memory network is represented by a matrix that stores the similarities and inputs based on internal relevance of word pairs, which is adaptive for answer selection.

Specifically, in the proposed memory network scheme for MMN, we divide our proposed model into the following multiple parts:

- Similarity Construction: a distributed similarity matrix is computed by bilinear function between the Q-A or Q-C pairs. The same similarity matrix generation is used as in Chapter 4. A top number of significant word pairs are selected by max-pooling function for question and answer, respectively.
- Pre-processing Layer: relevant Q-A pairs searching as the internal resources to initialize memory network
- Memory Module: Attentive memory network stores the texts and meaningful word pairs from similarity module to improve the context of memory using proposed multi-dimensional approach. The memory update mechanism refines the memory matrix according to the previous memory and the related inputs information produced by the attention mechanism.
- Matching Layer: the refined memory focuses on combing the distributed similarity matrix between question and answer interaction for answer selection task. We utilize a bi-linear matching function to compute the relevance between the question and memory representations for MRC.
- Prediction Layer: The word-level combined representation is applied to soft-max function and trained based on a cross-entropy loss function for answer selection. A pointer network is used to search the answer spans of context and calculate a negative log-likelihood loss function for MRC.

In the experiments, we show empirical comparisons of the proposed MMN method and various state-of-the-art matching models. The answer selection datasets WikiQA and TREC, The MRC datasets TrivialQA and SQuAD are used to evaluate the performance. In addition, we analyze the performance of variations of the proposed model to empirically justify model design choices. The results show that the proposed MMN model outperforms the state-of-the-art methods and demonstrate that the model provides a robust approach for both answer selection and MRC tasks.

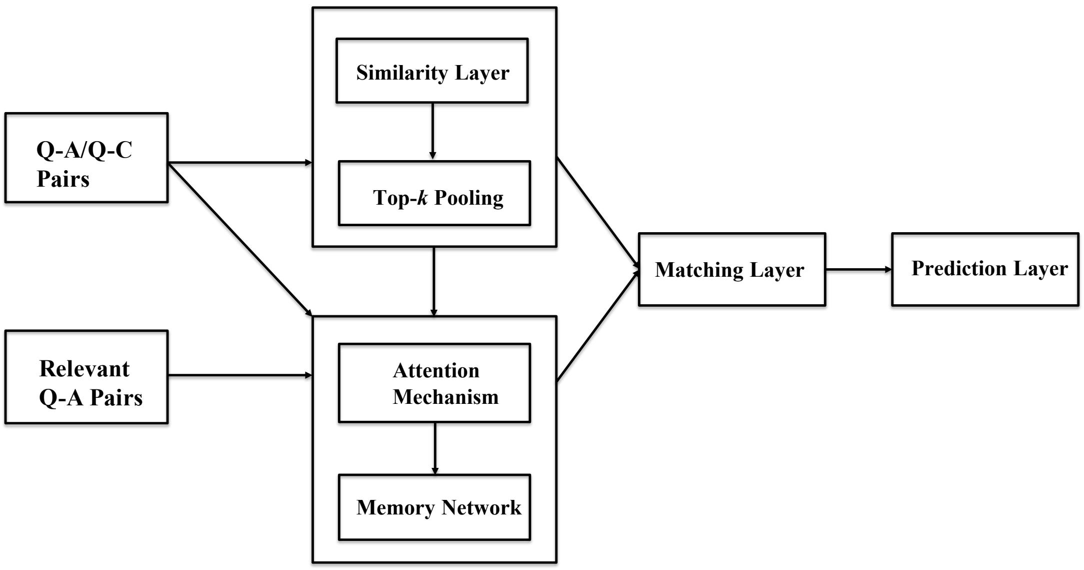


Figure 6.2: An illustration of the proposed model architecture.

6.2 The Proposed Work

Given a question q and answer a sentences, the goal is to rank and generate the relevant answer sentences from a pool of candidates for the answer section and machine comprehension tasks. In this work, we design a dynamic multi-dimension memory network (MMN) that efficiently stores the relevant sentence pairs into a memory matrix and individually refine it in two dimensions of memory according to input sentences. Thus, the memory update mechanism learns the semantic information between question and answer required to improve the matching performance. The multi-dimension memory network architecture is illustrated in Figure 6.2. In this section, we use the answer selection task as an example to describe the design of the proposed model in detail.

6.2.1 Similarity Construction

Bilinear Similarity Computation

Given a question $q = \{w_i^q\}_i^m$, and a candidate answer $a = \{w_i^c\}_i^n$, it is reasonable to assume that the answer's relevance depends on the semantic similarity between the words they contain. A distributed vector representation is employed to model the semantic similarity between words—each word is represented by d-dimensional vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_d]$. Subsequently, if the question has m words it can be represented as a $m \times d$ matrix $\mathbf{X}$, likewise given the answer has n words it is represented by an $n \times d$ matrix $\mathbf{Y}$, where the rows of each matrix correspond to the vector representations of the words appearing in the sentence. Defining the integers m and n as the maximal lengths of a question sentence and an answer or context sentence, variable length sentences can then be characterized by fixed-size matrices by adding zero rows to fill up empty positions for shorter sentences: the $m \times d$ matrix $\mathbf{X}$ denotes a question and the $n \times d$ matrix $\mathbf{Y}$ denotes an answer candidate. We adopt the same similarity matrix formulation Eq. (4.2.1) in Chapter 4, it computes the similarity for each pair of words, one from each sentence, and this results in the $m \times n$ similarity matrix S between the query q and answer candidate a.

Top-k Max Pooling

To aggregate significant information and to reduce the size of the similarity representation, we use a pooling process to select the number of top-ranked word pairs in the similarity matrix S in row and column directions individually, corresponding to word pairs importance between the question and answer. The pooling function focuses on each column of similarity matrix is defined as

$$\mathbf{P}^{(q)} = \text{top-}k \text{ max pooling} \left(\mathbf{S}[:, j]\right); \forall j \in [1, 2, \ldots, n], \tag{6.2.1}$$

with the pooling in each row of similarity matrix:

$$\mathbf{P}^{(a)} = \text{top-}k \text{ max pooling} \left(\mathbf{S}[i, :]\right), \forall i \ \in [1, 2, \ldots, m], \tag{6.2.2}$$

where $\mathbf{P}^{(q)}$ is $m \times k$ matrix and $\mathbf{P}^{(a)}$ is $k \times n$ matrix. The $\mathrm{S}[:, j]$ is a m dimensional vector, and the $\mathrm{S}[i, :]$ is a n dimensional vector. The pooling operation compares the semantic similarities between a word in the sentence and all the words in the other corresponding sentence and returns an aggregated similarity measure for that word. This results in the m-th element and n-th element of the similarity matrix $\mathbf{P}^{(q)}$, $\mathbf{P}^{(a)}$ for the question and answer, respectively. The top-ranked number k can be arbitrarily set to different values for questions and answers. In general, the value k of answer is bigger than the one of question, since the answer contains more words.

The two motivations for pooling are (1) to extract influential combinations of words for text pair so that the pooled vector can be conveniently fed into the next layer, and (2) to reduce redundant information and the number of model parameters. Commonly used pooling operations pooling include average, max, min, and stochastic pooling [138, 139]. In this work, max pooling is applied since it can be seen as searching for the best similarity matching for question and answer word.

6.2.2 Multi-dimensional Memory Network

In this section, we will build the proposed memory network for two parts: (1) pre-processing step extracts the top matching relevant questions and answers. (2) memory initialization. The architecture of initializing the proposed memory network is illustrated in Figure 6.3.

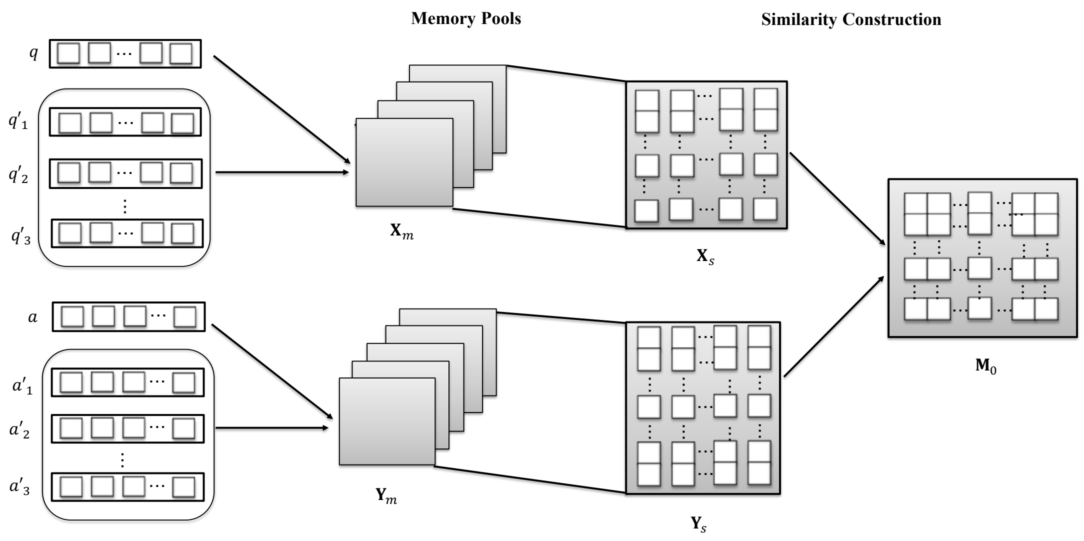


Figure 6.3: An illustration of memory initialization for matching a sentence pair.

Pre-Processing Sentence Representation

The relevant question and answer of corpus have the assistant information for the similarity between the original question and answer. In the processing step, we explore to collect a number of the relevant question and answer pairs that have the top matching between relevant question Q'-Q pair and relevant answer A'-A pair. Assuming the distributed embeddings of the question and answer are represented is $\mathbf{X}$, $\mathbf{Y}$, the embedding representations of relevant question and answer are set as $\mathbf{X}'$, $\mathbf{Y}'$. With respect to dimensional reduction, we convert the input representation from a matrix to a vector by normalizing the columns of matrix corresponding to each word in sentence. The l_2-norm formulation is computed by: $||\mathbf{Z}^{(l)}|| = \left(\sum_i^k \left(\mathbf{Z}_i^{(l)}\right)^2\right)^{\frac{1}{2}}$, where $\mathbf{Z}$ is represented by k-dimensional column vectors. Thus, the normalized question representation $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ is represented by a d-dimensional vector, the element $x_l = ||\mathbf{X}^{(l)}||$ where $\mathbf{X}^{(l)}$ denotes the l-th element of the row vector in question embedding matrix $\mathbf{X}$, and $l \in \forall[1, 2, \ldots, d]$. The same process with normalizing question representation, we obtain the d-dimensional vectors of the relevant question representation $\mathbf{x}'$, the answer representation $\mathbf{y}$ and the relevant answer representation $\mathbf{y}'$, respectively.

In order to search the best Q'-A' pairs, we employ the cosine similarity to compute the similarity score $cos[\mathbf{x}, \mathbf{x}']$ between question and its relevant question, and $cos[\mathbf{y}, \mathbf{y}']$ between answer and its relevant answer. A certain number of relevant Q-A pairs are collected from dataset corpus under the same category with input Q-A pair. We rank a top number l_q, l_a of relevant questions and answers depending on the magnitudes of similarity scores. Then we feed these specific relevant question and answer embeddings into 3-dimensional memory pools $\mathbf{X}_m$, $\mathbf{Y}_m$, where the entire number of sentences in the memory pools is l_q, l_a, separately.

Considering various approaches for dimensional reduction of sentence representation, the l_2-norm combines all elements in column vector of sentence representation. The commonly method used in option is to set weighted averaged function [34] as dimensional reduction approach, the l-th element in sentence embedding vector is computed by: $z_l = \frac{1}{k}\sum_i^k \mathbf{w}_i \mathbf{Z}_i^{(l)}$, where $\mathbf{w}_i$ is the weight for the l-th element. Apart from above approaches, a bi-linear function is used to transform sentence embedding matrix to a vector representation by: $z_l = f(\mathbf{w}^{\mathrm{T}}\mathbf{Z}^{(l)} + b)$, where the parameter $\mathbf{w}$ is a m-dimensional row vector. Subsequently, we will compare and discuss these three methods in the Section 6.4.2.

Memory Network Initialization

A series of relevant question and answer pairs contain enrich information that relates to the original sentence pairs. Assuming a distributed memory representation $\mathbf{M}_0$ set as a $m \times n$ matrix at time $t = 0$. We use the memory pools of relevant question and answer sentences $\mathbf{X}_m$, $\mathbf{Y}_m$ to initialize the memory.

Memory pools store a number of related questions and answers. Prior to initializing memory matrix, we individually integrate the related l_q questions and l_a answers from the memory pools. The 2-dimensional matrices of memory slots are defined by the weighted sum function: $\mathbf{X}_s = \sum_i^{l_q} \mathbf{V}_i \mathbf{X}_m^{(i)}$ for question; $\mathbf{Y}_s = \sum_j^{l_a} \mathbf{U}_j \mathbf{Y}_m^{(j)}$ for answer, where $\mathbf{V}_i$ denotes $m \times m$ matrix, and $\mathbf{U}_j$ is $n \times n$ matrix. $\mathbf{X}_m^{(i)}$ denotes the i-th element $m \times d$ dimensional matrix of the memory pool $\mathbf{X}_m$. $\mathbf{Y}_m^{(j)}$ denotes the j-th element $n \times d$ dimensional matrix of the memory pool $\mathbf{Y}_m$. In the initialization process, we still follow the bi-linear similarity computation in Chapter 4. Two memory slots $\mathbf{X}_s$, $\mathbf{Y}_s$ are used in Eq. (4.2.1) as the inputs. The similarity output is computed as the initialized memory $\mathbf{M}_0$. This memory contains interactive information between the relevant question and answer pairs. In the next step, we use the initialized memory matrix to refine and update the memory matrix.

6.2.3 Memory Refinement

Recent QA models that use the memory [14, 15, 86] as a vector to store the information from an external controller. In this work, we set a $m \times n$ dimensional memory matrix $\mathbf{M}_t$ at the current time t in order to store not only the local inputs information, but also the relevant questions and answers. A running example with three QA sentence pairs in Figure 6.4 shows the effect of refinement operation with three hops for WikiQA dataset. It can be seen from the figure that memory refinement operation demonstrates the correct answer moves closer to the question in ranking list while incorrect one moves further.

Consequently, it is difficult to update the entire memory matrix with less computation, NTM [85] has been designed to solve this issue by updating a vector from each row element in memory matrix. To write the inputs and interactive similarities from

Question Q

- how does interlibrary loan work?

Candidate Answers: A_1 **and** A_2 **(hop 1)**

- A_1: The lending library usually sets the due date and overdue fees of the material borrowed. (A_-)
- A_2: In many cases, nominal fees accompany interlibrary loan services. (A_-)
- A_3: The user makes a request with their local library, which, acting as an intermediary, identifies owners of the desired item, places the request, receives the item, makes it available to the user, and arranges for its return. (A_+)

Candidate Answers: A_{21} **and** A_{22} **(hop 2)**

- A_{21}: The lending library usually sets the due date and overdue fees of the material borrowed. (A_-)
- A_{22}: The user makes a request with their local library, which, acting as an intermediary, identifies owners of the desired item, places the request, receives the item, makes it available to the user, and arranges for its return. (A_+)
- A_{23}: In many cases, nominal fees accompany interlibrary loan services. (A_-)

Candidate Answers: A_{31} **and** A_{32} **(hop 3)**

- A_{31}: The user makes a request with their local library, which, acting as an intermediary, identifies owners of the desired item, places the request, receives the item, makes it available to the user, and arranges for its return. (A_+)
- A_{32}: The lending library usually sets the due date and overdue fees of the material borrowed. (A_-)
- A_{33}: In many cases, nominal fees accompany interlibrary loan services. (A_-)

Figure 6.4: Example scenario with three QA sentence pairs for memory refinement operation with three hops. In the candidate answers ranking list with each hop, a correct answer is labeled as A_+, an incorrect one is marked as A_-.

controller to memory matrix, we propose the rows and columns two writing heads to update the memory matrix through involving the answer and question information, respectively. The memory matrix updates twice with two writing heads at each time t. The memory refinement structure of the proposed model for text matching is illustrated in Figure 6.5. The memory updating process is a continuous mechanism, once the row elements are updated, the new memory matrix would be used as the input for renewing columns.

Assuming the $m \times n$ memory matrix at current episode t denotes $\mathbf{M}_t$. The i-th element row vector of memory matrix represents $\mathbf{M}_t[i,:]$, $\forall i \in [1,\ldots,m]$, where each row of memory matrix denotes n dimensional vector. The j-th element column vector of memory matrix is $\mathbf{M}_t[:,j]$, $\forall j \in [1,\ldots,n]$, where each column of memory matrix denotes m dimensional vector. To simply the notations convenient for computation, we define the row vector $\mathbf{M}_t[i,:] = \mathbf{m}_t^i$ and the column vector $\mathbf{M}_t[:,j] = \mathbf{m}_t^j$ of memory matrix at the t-th time.

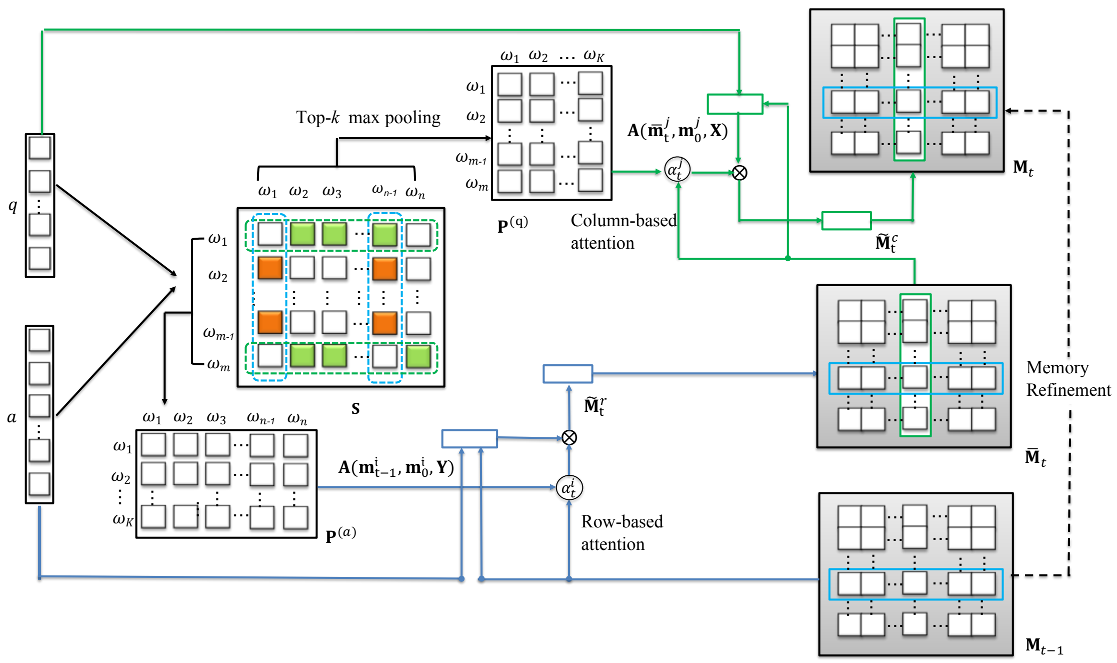


Figure 6.5: An illustration of memory refinement for matching a sentence pair.

Row-based memory

We first update each row vector in the previous episode memory matrix $\mathbf{m}_{t-1}^{i}$, The activation function Rectified Linear Unit ($ReLU$) [65] is applied to convert the element to be sparse, we update the memory by using the following defined functions

$$\bar{\mathbf{M}}_t = ReLU(\mathbf{W}_r \tilde{\mathbf{M}}_t^r + \mathbf{b}_r), \tag{6.2.3}$$

where the $\mathbf{W}_r$ denotes a $m \times 4$ dimensional matrix, $\mathbf{b}_r$ is $m \times n$ dimensional matrix. Soft attention as given by a contextual matrix through a weighted summation of vectors $\mathbf{A}(\mathbf{m}_{t-1}^{i}, \mathbf{m}_0^{i}, \mathbf{Y})$ and attention weight α_t^i. The contextual memory is computed by using the attention function, given as

$$\tilde{\mathbf{M}}_t^r = \sum_{i=1}^{m} \alpha_t^i \mathbf{A}(\mathbf{m}_{t-1}^{i}, \mathbf{m}_0^{i}, \mathbf{Y}), \tag{6.2.4}$$

where the contextual memory matrix $\tilde{\mathbf{M}}_t^r$ represents a $4 \times n$ matrix. The attention mechanism is responsible for generating the contextual memory matrix $\tilde{\mathbf{M}}_t^r$ based on the previous episode memory vector $\mathbf{m}_{t-1}^{i}$, the initialized memory vector $\mathbf{m}_0^{i}$ and the answer sentence representation $\mathbf{Y}$. The attention function $\mathbf{A}(\mathbf{m}_{t-1}^{i}, \mathbf{m}_0^{i}, \mathbf{Y})$ for each row vector of memory is defined as

$$\mathbf{A}(\mathbf{m}_{t-1}^{i}, \mathbf{m}_0^{i}, \mathbf{Y}) = \begin{bmatrix} \mathbf{m}_{t-1}^{i} \odot \mathbf{Y}\mathbf{w}_y \\ |\mathbf{m}_{t-1}^{i} - \mathbf{Y}\mathbf{w}_y| \\ \mathbf{m}_{t-1}^{i} \odot \mathbf{m}_0^{i} \\ |\mathbf{m}_{t-1}^{i} - \mathbf{m}_0^{i}| \end{bmatrix}, \tag{6.2.5}$$

where $\mathbf{w}_y$ is a d-dimensional row vector. The symbol $\odot$ is element-wise product. The symbol $|\cdot|$ is defined as the element-wise absolute value. The feature set $\mathbf{A}(\mathbf{m}_{t-1}^{i}, \mathbf{m}_0^{i}, \mathbf{Y})$ aggregates four vectors of n elements to a $4 \times n$ matrix. The feature vector captures a variety of similarities between input sentence and memory [14]. The feature set $\mathbf{A}(\mathbf{m}_{t-1}^{i}, \mathbf{m}_0^{i}, \mathbf{Y})$ is composed of four different similarity vectors between answer sentence representation, memory and initialized memory. The similarity between related question and answer sentences representations is used as an initialized memory to generate a feature vector instead of a single input sentence.

The attention weight is computed depends on the softmax function

$$\alpha_t^i = \frac{exp(z_t^i)}{\sum_{l=1}^{m} exp(z_t^l)}. \tag{6.2.6}$$

The attention weight is computed by the relational value z_t^i, which depends on the previous memory vector and the pooled similarity matrix of answer. The relational function is defined as

$$z_t^i = f((\mathbf{m}_{t-1}^i + \mathbf{w}_{p_1}\mathbf{P}^{(a)})\mathbf{w}_a + b_a), \tag{6.2.7}$$

where the weight $\mathbf{w}_a$ is a n-dimensional row vector, $\mathbf{w}_{p_1}$ is a k-dimensional column vector, and bias b_a is a scalar.

Column-based memory

After updating the rows of episode memory matrix, we adopt the new $m \times n$ dimensional memory matrix $\bar{\mathbf{M}}_t$ as the input to update the columns of memory matrix. The j-th elements column vector of $\bar{\mathbf{M}}_t$ is set to $\bar{\mathbf{m}}_t^j$. Subsequently, the memory matrix at current time t is computed using the columns of memory matrix, given as

$$\mathbf{M}_t = ReLU(\tilde{\mathbf{M}}_t^c \mathbf{W}_c + \mathbf{b}_c), \tag{6.2.8}$$

where the weight $\mathbf{W}_c$ denotes a $4 \times n$ dimensional matrix, $\mathbf{b}_c$ is $m \times n$ matrix. Next, we follow the same steps as updating the row-based memory, but with different inputs. The previous memory $\bar{\mathbf{m}}_t^j$, initial memory vector $\mathbf{m}_0^j$ and question sentence representation are used to compute the column-based contextual memory $\tilde{\mathbf{M}}_t^c$ as the inputs in Eq.(6.2.4).

6.2.4 Mnemonic Deep Similarity Matching

Above memory update mechanism produces the episodic memory matrix $\mathbf{M}_t$ at current time t based on the contextual representations and memory matrix at previous time $t-1$. The memory network starts from row-based updating of the previous memory $\mathbf{M}_{t-1}$, and ends at completing the update of columns-based as one iteration. Assuming that the number of iterations ('hops') of the memory network is T times

($0 \leq t \leq T$), the final episodic memory $\mathbf{M}_T$ is able to include significant information required to semantic matching between question and answer. We combine the memory matrix $\mathbf{M}_T$ and the similarity matrix $\mathbf{S}$ by using the element-wise multiplication function, given as

$$\mathbf{F} = \mathbf{A}(\mathbf{M}_T, \mathbf{S}) = \mathbf{M}_T \odot \mathbf{S}. \tag{6.2.9}$$

To aggregate significant information and to reduce the size of the similarity representation, we apply a pooling process to the computed similarity matrix $\mathbf{S}$. The pooling function returns a similarity vector that contains the most important pairs between question and answer, is defined as

$$\mathrm{s}^{(l)} = \text{max-pooling}\left(\mathbf{F}^{(l1)}, \mathbf{F}^{(l2)}, \cdots, \mathbf{F}^{(ln)}\right), \ \forall l \ \in \ [1, 2, \ldots, m]. \tag{6.2.10}$$

where $\mathrm{s}^{(l)}$ denotes the l-th element of the similarity vector s, and $\mathbf{F}^{(lk)}$ denotes the lk-th element of the full $m \times n$ matrix $\mathbf{F}$. The max-pooling operation compares the semantic similarities between a word in the question sentence and all the words in the answer candidate, and returns an aggregated similarity measure for that word. This results in a length-m similarity vector s for each question.

6.2.5 Matching Prediction

Given the Q-A pair in answer selection task, the probability that an answer candidate a is related to q can be modelled using two-way softmax based on the encoding of the similarity representation $\mathbf{h}$, given as

$$p(t = 1|\mathbf{s}) = \frac{\exp\left(\mathbf{s}^{\mathrm{T}}\boldsymbol{\alpha}_1\right)}{\exp\left(\mathbf{s}^{\mathrm{T}}\boldsymbol{\alpha}_0\right) + \exp\left(\mathbf{s}^{\mathrm{T}}\boldsymbol{\alpha}_1\right)}, \tag{6.2.11}$$

where the two column vectors $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ are softmax parameters with the same dimensionality as the similarity vector $\mathbf{s}$. The matching prediction task is formulated as a binary classification problem, where the label $t = 1$ indicates that a is related to q, while $t = 0$ otherwise.

In answer selection, given a collection of question and answer candidate sentences with available ground truth knowledge of whether they are related, the traditional training approach optimizes the model variables by minimizing the regularized cross-entropy cost function as shown below

$$L(\boldsymbol{\theta}) = -\sum_{(i,j)\in I} [t_{ij} \log p(t_{ij} = k|\mathbf{s}_{ij}) \\ +(1 - t_{ij}) \log (1 - p(t_{ij} = k|\mathbf{s}_{ij}))] + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 , \tag{6.2.12}$$

where the index set I denotes the used training sentence pairs, and $\lambda > 0$ is the regularization parameter set by the user. The training of the proposed text matching model involves the bilinear similarity weights and biases $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{b}_s\}$, a set of memory network parameters $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{W}_r, \mathbf{W}_c, \mathbf{w}_{p_1}, \mathbf{w}_{p_2}\mathbf{w}_a, \mathbf{w}_q\}$, as well as the softmax parameters $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$. The proposed memory network is "end-to-end" structure, thus the model training is direct and simpler than using the pre-prepared external resource as memory network.

To train this model, we apply back-propagation to update model variables of network [144, 145]. With many parameters in the neural network model, there is the danger of overfitting on a small size of training data. To tackle the overfitting issue, we use the dropout technique along with early stopping as surrogate forms of regularization for the matching model [146, 147]. The dropout method prevents feature co-adaption by randomly removing hidden units from the neural network training process.

6.2.6 Model Specification for Machine Comprehension

Apart from answer selection, the proposed model is capable of completing machine comprehension task. Given a question $q = \{w_i^q\}_i^m$ and context $c = \{w_j^c\}_j^n$ pair, where the m, n is the number of words in question, and context. In general, $n \geq m$. The goal of this task is to predict an answer a that is constrained as a segmented text of context. Subsequently, we set the question and context pair (q, c) as inputs into the proposed model instead of question and candidate answer pair (q, a).

In particular, to initialize memory network $\mathbf{M}_0$ in sub-section 6.2.2, we set the memory slots $\mathbf{X}_s$ and $\mathbf{Y}_s$ to question embedding $\mathbf{X}$, and context embedding $\mathbf{Y}$. Here, we replace the pre-processing module for machine comprehension due to the context contains a number of sequences related to question, and the length of context is much longer than the one of a candidate answer. Subsequently, the question, context inputs embeddings $\mathbf{X}$, $\mathbf{Y}$ and the initial memory matrix $\mathbf{M}_0$ as the inputs into the memory update mechanism. By the final hop T, the updated memory network $\mathbf{M}_T$ captures the semantic information from context required to answer the question.

In matching layer, different from the memory focuses interactive similarity pairs between question and answer in answer selection, machine comprehension task requires the model find a segment of sequence spans of the context to answer the question. For machine comprehension task, a pointer networks [158] is a popular decoding approach to predict the start and end position of the answer. We follow the approach in [126] to compute the start and end distribution of words in context using a bilinear matching between the question and current memory, given as

$$\mathbf{p}_{\mathrm{s}} = softmax\left((\mathbf{X}\mathbf{w}_s)^{\mathrm{T}}\mathbf{M}_T\right), \tag{6.2.13}$$

with the probability distribution of end index as

$$\mathbf{p}_{\mathrm{e}} = softmax\left((\mathbf{X}\mathbf{w}_e)^{\mathrm{T}}\mathbf{M}_T\right), \tag{6.2.14}$$

where the trainable weights $\mathbf{w}_s$, $\mathbf{w}_e$ both are d-dimensional row vector. The notation $softmax(\cdot)$ represents softmax function shown in Eq.(6.2.6). According to the bilinear function, we obtain the n-dimensional distributed outputs $\mathbf{p}_{\mathrm{s}}$, $\mathbf{p}_{\mathrm{e}}$.

In training process, the boundary detecting method [159] is adopted for machine comprehension task, it minimizes the sum of the negative log probabilities of the true start and end position by the predicted distributions, the loss for start and end position is defined as

$$L(\boldsymbol{\theta}) = -\sum_{i \in D} \log \mathbf{p}_{\mathrm{s}}(y_i^{\mathrm{s}}) + \log \mathbf{p}_{\mathrm{e}}(y_i^{\mathrm{e}}), \tag{6.2.15}$$

where the index set D denotes the training question and context pairs. y_i^{s} and y_i^{e} are the ground-truth start and end position indices of i-th pair, respectively. The model overall variable $\boldsymbol{\theta}$ is the set of entire trainable weights and bias, including bilinear similarity and multi-dimensional memory network parameters.

6.3 Experimental Analysis and Results

In this section, we empirically analyze and evaluate the design of the proposed model MMN on two tasks: answer selection and machine comprehension. We first introduce the experimental setting of the MMN model in sub-section 6.3.1, then we compare the proposed model with multiple state-of-the-art models on various benchmark datasets. Finally, we demonstrate the properties of the proposed model through empirical analysis.

6.3.1 Datasets

In answer selection, we focus two benchmark datasets: TREC and WikiQA datasets. TREC[1] [148], is generated from TREC QA tracks 8-13, which each contain a set of factoid questions and candidate answers [148]. The correct answers for each question are manually labeled and ranked in the dataset. WikiQA[2] [13], is the public released QA dataset in which all answers are collected from Wikipedia.

In machine comprehension, we also adopt two authoritative datasets to evaluate the proposed model: Stanford Question Answering Dataset (SQuAD) and TriviaQA datasets. The SQuAD dataset [18] totally consists of more than 100k questions manually annotated by crowd sourcing workers on 536 Wikipedia articles. Each question corresponds to contexts is a paragraph collected from an article. The best answer responses to question is a segment of text to be a span of contexts. The dataset contains 87k question context tuples for training, 10k tuples for validation. We follow the same experimental setting as in [160] by dividing 10% of training samples as the test set, and computing performance when training on subsets of the remaining samples of the entire dataset.

TriviaQA[3] [161] is a recent popular machine comprehension dataset consisting of over 650K question-answer-context triples, which consist of 95K Trivia QA pairs with the average six contexts as supporting evidence for each question. A dataset of questions in trivia question databases paired with contexts is collected from either

[1]http://`trec.nist.gov/data/qa/t8qa_data.html`
[2]https://`aka.ms/WikiQA`
[3]http://`http://nlp.cs.washington.edu/triviaqa/`

Table 6.1: Dataset content statistics.

Parameter	TREC	WikiQA	SQuAD	TriviaQA Wiki
No. of Questions	1,505	3,047	97,000	77,400
No. of Answers	60,800	29,258	-	-
No. of Contexts	-	-	20,800	138,538
Avg. Q-Length(words)	11.39	7.26	11.0	15.0
Avg. A-Length(words)	24.63	24.94	-	-
Avg. C-Length(words)	-	-	122	495

Table 6.2: Benchmark data splits.

Data Set	Q/A Pairs	Development	Training	Testing
TREC [32]	8,997	1,148	4,718	1,517
WikiQA [13]	29,258	2,733	20,360	6,165
SQuAD [18]	100,000	10,000	78,300	8,700
TriviaQA Wiki [161]	78,582	7,900	61,800	7,700

Wikipedia (Wiki) or Web search results. TriviaQA web dataset is derived from TriviaQA database with unfiltered strings in each question context pair. Different with TriviaQA web, TriviaQA Wiki ensures each document has a segment of text as the answer label. The mean length of original contexts contains 2,895 words is much larger than the one in SQuAD dataset. Thus, we truncate the contexts followed with [125], and reduce the average length of contexts to 495 words. We will test the full and verified subsets of TriviaQA Wiki dataset in experimental evaluation, where the verified subsets contains the part of full dataset correctly answered question context pairs in [161]. More detailed data information for datasets is shown in Table 6.1.

6.3.2 Performance Measures

To report model performance using the test set, we use three performance metrics to measure the performance in answer selection, namely mean reciprocal rank (MRR) and mean average precision (MAP), as in [150]. The MRR metric focuses on the order of the correct answers, and is formulated as

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{r_i^1}, \tag{6.3.1}$$

where r_i^j denotes the computed ranking of the j-th correct answer in the ground truth ranking list for the i-th question, and $|Q|$ denotes the total number of questions tested. In other words, with $j = 1$, r_i^1 denotes the best possible answer. MAP accumulates the mean ranking of all the correct answers in each question, expressed as

$$\text{MAP} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{n_i}\sum_{j=1}^{n_i}\frac{n_i^j}{r_i^j}, \tag{6.3.2}$$

where r_i^j is the computed ranking of the j-th correct answer from the ground truth ranking list for the i-th question, n_i^j is the number of truly correct answers in the computed ranking list of the j-th correct answer, and n_i denotes the number of truly correct answers for the i-th question.

In machine comprehension, we use two different performance metrics to evaluate the model accuracy. Exact match (EM) [159] measures the precision of predicted answer that match any one of the groundtruth answers exactly over all test question context pairs. The exact match score is equal to 1 when the prediction is exactly the same as groundtruth or 0 otherwise. Alternative metric is F1 score [162] that indicates the average of word overlap between the prediction and ground truth answer, where the predicted answer and groundtruth are treated as bags of words. In evaluation, we adopt the maximal F1 score over all of the possible ground truth answers for a given question, and then average it over all of test questions.

6.3.3 Experimental Configuration

Experimental platform and recordings: All the training and testing were carried on a Barkla HPC cluster with two 40-cores, 1 TB shared memory system for large memory jobs along with one GPU node equipped with four Nvidia NVLink P100 GPUs, and running the new version of the Tensor Flow Framework (v1.6) supported by Nvidia CUDA library (v.8.0).

Neural network configurations: In answer selection task, considering the top-k max pooling process, the number of top ranked word pairs k is set to 10 for question, and 30 for answer. In pre-processing step, the number of relevant questions l_q=10, and the number of relevant answers l_a=50. The total number of iterations or hops is defined as T=3 for memory update mechanism. In machine comprehension, the number k is set to 15 for question, and 150 for context. The total number of iterations is T=4 for memory update mechanism.
Training preparation and initialization: In preparing the dataset for training and testing, we followed the same text pre-processing procedures described in [33]. More specifically, the out-of-vocabulary words are mapped to a special token symbol $\langle$_UNK$\rangle$. Wherever the sentence lengths fall below the minimum threshold, a special symbol, $\langle$_PAD$\rangle$, is added to the end of the sentence, so as to pad them with extra characters (e.g. zeros) to meet the processing requirements. Furthermore, the basic pre-training model is Glove [142] using a a corpus containing 27B words from 2B Tweets. The Tweets have been filtered by removing infrequent words, resulting in 1.2M words from the English vocabulary. The dimensionality of each word embedding vector is set as 100. For words appearing in each dataset, but out of vocabulary, a random value uniformly sampled from the interval of $[-0.3, 0.3]$ is assigned to each embedding dimension. For model variables to be initialized, a normal distribution $\mathcal{N}(0, 0.1)$ is used.
Training / testing process: For model optimization, a root mean square propagation (RMSProp) optimizer is used. The process includes a mini-batch containing 50 training examples, a learning rate of 0.1, and a dropout rate of 0.5 [146]. The learning rate is halved after 10 epochs. Gradient clipping [157] is used to scale the gradient when the norm of gradient exceeds a threshold of five. The overall datasets have been split for training, testing and development purposes as suggested by the original datasets [13, 19, 21, 32]. These are given in Table 6.2. The parameters adopted above were selected using the development set based on a coarse manual tuning.

6.3.4 Baselines

To compare with the proposed method, the following ten models, stemming from the space of CNN, RNN and conventional/traditional techniques, are considered.

Baseline Models:

1. Random Guess (RandomGuess) [56]: A random ranking list for the test samples without training process.

2. Word Embedding (WordEmbed) [90]: This model uses the Glove tool to obtain the word embedding representation of a sentence. The matching score of two short-texts are calculated with MLP, taking two sentence embeddings as inputs.

3. Classifier [161]: A linear classifier to classify the sentence pairs for answer.

Comparison Models:

4. IARNN [40]: The model learns an answer sentence representation using an attention mechanism to involve a question hidden representation from an RNN network, which then generates a high-level answer sentence representation as the input to the RNN network.

5. BiMPM [119]: The model first encodes two sentences with a bi-directional LSTM, the encoded output of a sentence match each hidden representation of the other sentence in two directions. The sequences of matching vectors are aggregated into a vector as an input to prediction layer.

6. IWAN [1]: The model builds an alignment layer depends on a word-level similarity matrix to compute attention weight of each word, where the similarity matrix is computed by the sentence encoded outputs from bi-directional LSTM.

7. CAM [2]: The model proposes the model performs different comparison matching functions to match the sentences based on word-level, where the similarity outputs from the function are aggregated into a vector by a convolution layer. The convolved vector as the input into the final prediction layer to compute the matching score.

8. MEMEN [59]: The model designs a hierarchical attentive memory to learn an alignment memory matrix, which contains the syntactic and semantic information of the words returned by skip-gram model.

9. SLQA [126]: The model provides the fusion functions to combine self-attention and similarity matrix to complete the machine comprehension.

10. M-Reader [125]: The model uses re-attention mechanism to refine current attentions for machine comprehension.

For the purpose of evaluation, we collect the reported results from the published works of these models, wherever possible. Wherever this was not feasible, we implemented them to match with the reported specification and experimental evaluation of these models.

6.4 Results and Analysis

6.4.1 Comparison with State-of-the-art Methods

Answer Selection

We first evaluate the performance of proposed MMN model in QA task, the proposed method is compared with seven state-of-the-art approaches using the benchmark TREC and WikiQA datasets. For evaluation, we compare the performance of the proposed model across a number of techniques using the MAP and MRR metrics in previous Section 6.3.2. Table 6.3 reports the MRR and MAP metrics for different models evaluated using the two datasets mentioned above. It can be seen from Table that the proposed MMN model with designed memory network provides the best performance under all the evaluation setups. In the following, we report a number of more specific observations from Table 6.3:

- With respect to the MRR result, where a higher value indicates better performance, the proposed approach outperforms all models when evaluated against the TREC and WikiQA datasets. In particular, the proposed outperforms

Table 6.3: Performance comparison using TREC and WikiQA data for different models. The best results are highlighted and the second best results are underlined.

	TREC		WikiQA	
Models	MRR	MAP	MRR	MAP
Random Guess [96]	0.8511	0.7530	0.6957	0.6886
WordEmbed [96]	0.8511	0.7530	0.6957	0.6886
AP-CNN [96]	0.8511	0.7530	0.6957	0.6886
Ab-CNN [37]	0.8539	0.7741	0.7108	0.6921
KV-MemNNs [128]	0.8523	0.7857	0.7265	0.7069
IARNN [40]	0.8208	0.7369	0.7418	0.7341
BiMPM [119]	0.8750	0.8020	0.7310	0.7180
IWAN [1]	<u>0.8890</u>	<u>0.8220</u>	0.7500	0.7330
CAM [2]	0.8659	0.8145	<u>0.7545</u>	<u>0.7433</u>
MMN	**0.8865**	**0.8390**	**0.7752**	**0.7545**

the second best performing CNN-based model CAM, by 2.06%, and 2.07% respectively.

- When considering the MAP performance, the proposed approach outperforms the CAM model, when compared against the TREC and WikiQA datasets, by 2.45%, and 1.12% respectively.

Although most recent works are likely to use structured CNN or RNN-based models into answer selection, there is a memory network called KV-MemNNs in the Table offers a good performance on both two datasets. With respect to KV-MemNNs model using the external knowledge database, our proposed memory network focuses on utilizing the related internal sentences to input Q-A pair, it aims to reduce the larger computation and manual data creation caused by the external information resource. Additionally, different with KV-MemNNs model discards the original input information to help memory network updating, we prefer to explore the combination approach to involve the input sentences in order to prevent the information loss after several iterations. Besides, it is significant to examine the combination method between similarity and the current memory matrix. In proposed MMN model, we adopt an element-wise multiplication function in sub-section 6.2.4, which provides the best performance than other combination functions, such as, add, subtraction and bilinear. Regard to this point, CAM model verifies the multiplication function provides a satisfied result to improve the model performance, especially on TREC and Wiki datasets. Overall, the proposed model outperforms all the existing deep learning works in answer selection.

In fact, we observed that not all relevant question-answer pairs can provide useful information to the input question-answer pair. We analyze an example where the proposed model fails to handle the relevant question-answer pairs with a small amount of information from TREC dataset. For the question of *"during what war did Nimitz serve ?"*, the ground truth answer is *"Conant had been a photographer for Adm. Chester Nimitz during World War II."*. However, when tested, the proposed model chose *" Yuengling served as a staff sergeant with the U.S. Army Air Corps during World War II."* as the correct answer. In this example, the relevant question is *"what town was Nimitz native of ?"* and its correct available answer is *"Fredericksburg native Admiral Chester Nimitz, under whom bush served."* The relevant question-answer pair has less information that relates to the input sentence pair, the proposed model can not encode the information to select the ground truth answer.

Machine comprehension

To analyze the proposed model effectiveness in machine comprehension, we test the model performance using TriviaQA Wikipedia dataset. Table 6.4 illustrates that the EM and F1 scores for different models evaluated on two types datasets of TriviaQA Wikipedia dataset: Full, Verified. A baseline based on Classifier model is included to provide a general view of the problem. With respect to SQuAD dataset, a longer length of context increases the complexity to memory the long-term sentences information, and search the answer spans of context for the TriviaQA Wikipedia dataset. From the results in Table, the proposed MMN model shows the state-of-art performance among the comparison models on more complex dataset. In the

Table 6.4: Performance comparison on the TriviaQA Wikipedia data for different models. The best results are highlighted and the second best results are underlined.

	Full		Verified	
Models	EM(%)	F1(%)	EM(%)	F1(%)
Classifier [161]	23.40	27.70	23.6	27.90
BiDAF [43]	40.26	45.74	47.47	53.70
MEMEN [59]	43.16	46.90	49.28	55.83
M-Reader [125]	46.94	52.85	54.45	59.46
QANet [153]	51.10	56.60	53.30	59.20
document-qa [163]	63.99	68.93	67.98	72.88
BiDAF+SA [160]	-	-	69.03	74.61
SLQA [126]	<u>66.56</u>	<u>71.39</u>	74.83	78.74
MMN	**68.69**	**73.57**	**75.95**	**79.67**

following, we report a number of specific points from Table 6.4:

- With respect to both EM and F1 scores, the proposed approach outperforms all models when evaluated against the Full and Verified datasets. When considering the EM performance, the proposed outperforms the second best performing hierarchical attentive model SLQA, by 2.13%, and 1.12% respectively, on Full and Verified datasets.
- When considering the F1 performance, the proposed approach outperforms the SLQA model, by 2.18%, and 0.93% respectively, on Full and Verified datasets.
- With respect to the primary baseline, the proposed approach performs a much better than the Classifier model, where EM performance is improved by 45.29%, and 52.35% respectively, and F1 performance is improve by 45.87%, and 51.77% respectively, on Full and Verified datasets.
- From the observations in Table, EM and F1 scores perform a better improvement on Full dataset, when comparing with the above scores in Verified dataset

Table 6.5: Performance comparison on the SQuAD data for various competitive models. The best results are highlighted and the second best results are underlined.

	Dev Set		Test Set	
Models	EM(%)	F1(%)	EM(%)	F1(%)
LR Baseline [18]	40.0	51.0	40.4	51.0
Match-LSTM [159]	64.1	73.9	64.7	73.7
DCN+ [164]	74.5	83.1	75.1	83.1
Interactive AoA Reader [165]	-	-	73.6	81.9
FusionNet [166]	-	-	76.0	83.9
SAN [167]	76.2	84.0	76.8	84.4
BiDAF + SE [168]	-	-	78.6	85.8
MEMEN [59]	-	-	75.4	82.7
R-Net+ [169]	-	-	79.9	86.5
QANet [153]	-	-	76.2	84.6
M-Reader [125]	<u>78.9</u>	<u>86.3</u>	<u>79.5</u>	<u>86.6</u>
MMN	**79.8**	**87.1**	**80.2**	**87.3**

for the proposed MMN model.

Above results verify the proposed model not only offers a good performance on a small subset of dataset, e.g. Verified datset. It also shows the proposed is capable of performing a robust performance with a large scale dataset, e.g. the Full TriviaQA Wikipedia dataset. The second best model SLQA stacks the intermediate representations of question and context pair using multiple attention functions, without considering the order of words in a long context situation. In summary, the result shows that the proposed model offers the best performance among another published results on TriviaQA Wikipedia dataset.

To further examine the robustness of the proposed model, we conduct the evaluation using the popular SQuAD dataset. The proposed model is compared with ten state-of-art neural network models for this dataset, including recent model

QANet [153], BiDAF+SE [168], and Match-LSTM [159]. A LR baseline model based on linear regression is given to provided a standard view among all results. For evaluation, we use two different dataset to test model: Dev and Test sets. The same performance matrics EM and F1 as used in existing works. The performance for different models along with the proposed MMN model are reported in Table 6.5. It can be seen from Table that the proposed model preforms better than the competing models on both Dev and Test sets, demonstrating the superiority of the proposed model and its memory network strategy.

6.4.2 Empirical Analysis of Memory Network

In this section, we investigate different components of the proposed MMN model and compare them with alternative design options, to analyze the effectiveness of the proposed model. Specifically, a number of following experiments are explored.

- **Experiment 1**: An attentive memory update mechanism in proposed model based on the multi-dimension strategy, which refines the memory matrix from two sides individually. We analyze the memory network under the four compared conditions: memory only updates the column side but without updating the row side, referred to as MMN-Mr. Oppositely, memory only updates the row side but without updating the column side, referred to as MMN-Mc. An alternative choice is to update the two sides in a meanwhile by refining a memory matrix directly, referred to as MMN-Md. The last compared option is the proposed model without the memory network, which is referred to as MMN-M. The goal is to demonstrate the effect of the designed memory update mechanism of the proposed model.
- **Experiment 2**: The attention mechanism contributes to the memory update in both row and column sides. Here, we examine the usage of attention mechanism under the three different conditions: the proposed model only uses the attention mechanism in column-based memory update, but not in row-based memory update, referred to as MMN-Ar. On the contrast, the proposed model applies attention mechanism in column-based memory update, referred to as

MMN-Ac. Another approach of the proposed model is to remove the attention mechanism in both two sides, referred to as MMN-A.

- **Experiment 3**: In the pre-processing stage of memory network for answer selection, we investigate the advantage of learning distributed sentence embedding vector based on l_2-norm algorithm. An alternative bilinear function is used to transform the sentence embedding matrix to a vector instead of l_2-norm algorithm referred as MMN-bilinear. Another general method of reducing a sentence representation dimension based on averaged sum weighted function by stacking the elements in sentence embedding matrix, which is referred to as MMN-avg.

- **Experiment 4**: The effect of memory update is investigated by examining different numbers of iterations T, e.g., $T \in \{1, 2, 3\}$, referred as DSSM-HOP-1 when $T = 1$ for instance. These are compared to MMN that employs $T = 4$ memory update mechanism.

Table 6.6 summarizes performance of these compared settings over WikiQA and SQuAD datasets for answer selection and machine comprehension, respectively.

In experiment 1, we compare different memory update approaches. It can be seen from Table 6.6 that the proposed memory network update in both row and column dimensions results in the best performance, followed by the memory without row update MMN-Mc. Such a result represents that refining the memory with two dimensions is benefit for the proposed model, where the performance of memory update in row side MMN-Mc is better than the one in column side MMN-Mr. Whereas the proposed model without memory network MMN-M offers the worst performance, leads to both MRR and MAP results decrease by approximate 2.5$\%$. As mentioned in previous sections, one contribution of this work is to update the memory network in multi-dimension. Instead, a memory matrix MMN-Md is directly updated based on the previous memory, initialized memory, and answer representation, which provides a worse performance than the proposed MMN model. The computational time of MMN approximately costs 15 hours for WikiQA dataset, it is less than 10 hours of MMN-Md due to the specific memory refining structure of MMN. From the empirical

Table 6.6: Comparison of the proposed method with alternative designs and model settings, evaluated using the WikiQA and SQuAD datasets. The best performance is highlighted in bold and second best underlined.

	WikiQA		SQuAD	
Experimental settings	MRR	MAP	EM(%)	F1(%)
MMN-Mc	0.7596	0.7380	78.6	85.8
MMN-Mr	0.7572	0.7378	78.1	85.3
MMN-M	0.7300	0.7105	77.5	84.1
MNN-Md	0.7563	0.7340	78.4	85.6
MMN-Ac	0.7650	0.7442	79.1	86.2
MMN-Ar	0.7601	0.7395	78.7	85.9
MNN-A	0.7548	0.7343	78.0	85.5
MMN–avg	0.7690	0.7485	-	-
MMN-bilinear	0.7672	0.7467	-	-
DSSM-hop-1 (T=1)	0.7486	0.7212	77.5	84.3
DSSM-hop-2 (T=2)	0.7635	0.7389	78.8	85.7
DSSM-hop-3 (T=3)	**0.7752**	**0.7545**	<u>79.7</u>	<u>86.8</u>
MMN (T=4)	<u>0.7748</u>	<u>0.7541</u>	**80.2**	**87.3**

observation, a similar performance situation happens on the SQuAD dataset.

In experiment 2, we compare various designs of attention mechanism in memory network. As can be seen from Table 6.6, the proposed model uses attention mechanisms on the row and column sides of the memory matrix to cause the performance of the WikiQA dataset to increase by at least 1% compared to the MMN-Ac method. Such a result represents that the attention mechanism on each dimension is important for improving the matching performance, where the contextual information as the input into attention function. The proposed model without the attention mechanism MMN-A offers the worst performance, it leads to a performance decrease of at least 2% by comparing with the proposed model.

Experiment 3 compares three design options of learning d-dimensional sentence representation vector for answer selection, where non-parametric l_2-norm function of our proposed model aggregates the word elements in sentence embedding matrix. As seen from Table 6.6 that, by using neither bi-linear or average weighted sum as an transformation function, a fairly low performance is obtained since the lack of regularization of distributed sentence presentation. In details, the performance of the model with average weighted sum function MMN-avg and bilinear function MMN-bilinear are less than our proposed deign by 0.6% and 0.8%, respectively.

Experiment 4 compares the number of iteration in the memory update mechanism is increased from 1 to 4. It can be seen from Table 6.6 that the matching performance MAP and MRR increases until $T = 3$ over the WikiQA dataset, and then starts to be flat when $T = 4$. Different from WikiQA dataset, the predictive accuracy performance EM and F1 score continue to grow until $T = 4$ over the SQuAD dataset. Based on this empirical observation, a middle iteration number $T = 3$ for answer selection, and a larger number $T = 4$ for machine comprehension task are sufficient to achieve robust performance of the proposed model.

6.5 Conclusion

In this section, we have proposed a novel memory network architecture MMN is capable of completing multi-tasks in QA field: answer selection for semantic matching between the question and answer pairs; machine comprehension for generating an answer in a span of context by given question. The key idea of our proposed memory approach is to efficiently store and extract the semantic information related to question and answer or context pairs as an internal knowledge resource for the memory network, rather than construct the memory according to the external knowledge database. To achieve the goal, we search the certain number of relevant queries and answers, and compute the interactive similarities of word pair in order to initialize a memory matrix.

In memory update mechanism, since the computation of entire memory matrix update is expensive, we prefer to refine the memory matrix based on its row and column sides, respectively. Typically, the row and column of the memory matrix separately correspond to the importance of word pairs for the question and answer sentences. Therefore, we utilize the pair-wise attentive mechanisms to explicitly involve the question and answer or context into the corresponding side of the memory matrix for supporting the similarity matching between word pairs.

Additionally, the refined memory matrix with rich information as a flexible tool uses to improve the performance of different tasks. For instance, in answer selection, the memory matrix combines similarity representation focuses the word-level based interactive matching between question and answer. Differently, in machine comprehension, the memory matrix provides the contextual information required to question for predicting the answer.

In the experiment of four datasets for two different tasks, the experimental results indicate that our proposed method improves the matching performance over methods using the benchmark QA datasets. Besides, it also offers good predictive performance when generating an answer for given question in machine comprehension. Multiple performance comparisons with the state-of-art approaches also demonstrate the superiority of the proposed model.

The important challenge for semantic matching is to improve the task performance with relevant information that is easy to access. Overall, our proposed model finds a certain relevant quires and answers from data corpus, and adopts a multi-dimensional memory matrix to collect these relevant information. Thus, in future work, we aim to explore architectures that can utilize a simpler and efficient network to store a large number of input relevant information.

Chapter 7

Conclusion

In this thesis, we have proposed three different deep neural networks for handling semantic text matching problem for three types of QA tasks. Here we want to recap our contribution in this thesis and discuss some promising research directions in field.

7.1 Summary

In this thesis, we have researched the QA sentence matching with the proposed neural networks. Generally, the semantic sentence matching corresponds to the relationship between question and answer can be defined as the similarity representation learning for neural networks. The research work aims to design and employ the proposed neural networks to learn the high-level semantic similarity for QA tasks. Specifically, we have focused our model on three different learning approaches of semantic sentence matching in this thesis: **a)** distributed similarity based on word-level **b)** interactive sentence representation **c)** semantic similarity representation enhanced by memory network. According to these learning approaches, the proposed neural networks overcome the specific problems with different matching algorithms, including the semantic information loss of word pairs, the expensive computation cost, and low storage capacity of neural networks. Extensive experiments on multiple datasets and comparative analysis with the state-of-art existing methods in QA task demonstrate the superiority of our proposed neural networks.

The proposed models offers good performance and verifies the hypotheses of thesis come from varied aspects: **a)** In the first deep semantic matching model, we investigate to learn the distributed similarity representation by passing a entire sentence worth of word-level similarities. Designed bilinear matching function with parameter constraints controls the matching impact of each word in sentence for improving the distributed similarity learning. The proposed model benefits from its design of a pair-wise training method that combines an unsupervised training for the DAE network and an supervised training for semantic matching. The ranking measurements, e.g. MAP and MRR, experimentally report the proposed model outperforms the previous models. The proposed model with above quantities offers the best performance among all comparison variants. **b)** The second matching model learns the context-aware sentence representation for overcoming the similarity information loss between sentences, where the interactive similarity information may loss or be damaged during learning sentence representation at early step in deep neural networks. Thereby the specific attention mechanisms based on bi-directional LSTM are designed to continuously join the contextual and interactive information to this sentence representation. The proposed model is experimentally verified that the interactive sentence representation is helpful for semantic QA matching, especially for cQA dataset. **c)** The third model is extended to accomplish a more complex QA task – understanding context and generating the answer sentences in MRC. The proposed memory network model enables a large capacity of memory to store all interactive similarities of all word pairs. Subsequently, the memory returns these information to improve the initial similarity between a pair of sentence. The memory refinement individually stores sentences in row and column sides of the memory matrix, avoiding the proposed model trains the large dimensional parameters in the meantime. The proposed model is experimentally shows that the memory refinement is able to provide important word pairs to improve the semantic matching for both answer selection and MRC tasks.

It can be seen that the powerful performance of the proposed semantic matching models in different types of QA tasks and datasets, e.g., answer selection task in structured QA and cQA datasets, MRC task in QA with inferential context dataset. It is meaningful to learn a sentence representation using the deep neural network, since the deep neural network is efficient to learn the semantic meaning of word according to its deep layered architecture and flexible designed algorithms. The proposed methods in this thesis are able to improve and extend such semantic information.

7.2 Future Work

There are several possible extended directions for the future research. We briefly outline two interesting directions as follows.

Answer Generation In Dialogue System

Conversational modeling is an important task in natural language understanding and machine intelligence. Interactive conversational agents, virtual agents and sometimes chatter-bots, are used in a wide set of applications ranging from technical support services to language learning tools and entertainment [170]. In this work, we investigate the task of building open domain, conversational dialogue systems based on large dialogue corpora using generative models. Generative models produce system responses that are autonomously generated word-by-word, opening up the possibility for realistic, flexible interactions. The proposed model is built in the direction of end-to-end trainable, non-goal-driven systems based on generative probabilistic models. Here, the hierarchical recurrent neural networks are used as the generative model. The recurrent neural network has two main functions: encoding the question and decoding or generating the answer. The network would store information after encoding question, so that the information would be passed to the next conversation by building hierarchical structure. Furthermore, the attentional mechanism is considered to improve this generative model.

Image description for Visual Question Answering

As for Visual Question Answering (VQA) task, the goal is to generate answer according to the given image and question, hence, it is significant to investigate the joined representation of image and sentence. A new deep learning model is proposed to VQA that incorporates explicit spatial attention, the proposed model is based on memory network, which has been proposed in [86]. The memory network combines learned text embeddings with an attention mechanism and multi-step inference. For the sentences matching, the memory network stores textual knowledge to the memory in the form of sentences, and returns the relevant sentences to infer the answer. Nevertheless, once the matching object is changed to image, the knowledge is in the form of an image type, where the memory and question consists of different representation types. In this work, our proposed attentive memory network in Chapter 6 is employed to build the interaction between partial regions of image and question, which provides the specific information for answering the given question. An attention mechanism attentions to the partial region of image that joins with each word in question, then this combined inputs is fed to LSTM for generating encoded the representation of image and question. After encoding the joined representation, Generative Adversarial Nets (GAN) recently offers a well performance on image reconstruction [171]. The proposed model adopts it to generates the answer instead of traditional probability cross-entropy approach at the decoding stage.

.

References

[1] G. Shen, Y. Yang, and Z.-H. Deng, “Inter-weighted alignment network for sentence pair modeling,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1179–1189.

[2] S. Wang and J. Jiang, “A compare-aggregate model for matching text sequences,” in *Proceedings of the 5th ICLR International Conference on Learning Representations*, 2017.

[3] M. Paşca, “Open-domain question answering from large text collections,” *Studies in computational linguistics. CSLI Publications*, 2003.

[4] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *EMNLP*, 2015, pp. 1412–1421.

[5] B. Hu, Z. Lu, H. Li, and Q. Chen, “Convolutional neural network architectures for matching natural language sentences,” in *Proceedings of NIPS Conference on Advances in Neural Information Processing Systems*, 2014, pp. 2042–2050.

[6] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, and Y. Bengio, “A hierarchical latent variable encoder-decoder model for generating dialogues.” in *AAAI*, 2017, pp. 3295–3301.

[7] X. Shen, H. Su, Y. Li, and W. Li, “A conditional variational framework for dialog generation,” in *Proceedings of the 55th ACL Conference on the Association for Computational Linguistics*, 2017, pp. 504–509.

[8] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *Proceedings of the 20th AAAI Conference on Artificial Intelligence (AAAI-06)*, 2006, pp. 775–780.

[9] A. Islam and D. Inkpen, "Semantic similarity of short texts," *Recent Advances in Natural Language Processing V*, vol. 309, pp. 227–236, 2009.

[10] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1533–1544.

[11] X. Qiu and X. Huang, "Convolutional neural tensor network architecture for community-based question answering," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015, pp. 1305–1311.

[12] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," *Nist Special Publication SP*, vol. 109, p. 109, 1995.

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

[14] A. Kumar, O. Irsoy, and P. Ondruska, "Ask me anything: Dynamic memory networks for natural language processing," in *Proceedings of the 33rd ICML International Conference on Machine Learning*, 2016, pp. 1378–1387.

[15] F. Ma, R. Chitta, S. Kataria, J. Zhou, P. Ramesh, T. Sun, and J. Gao, "Long-term memory networks for question answering," *arXiv preprint arXiv:1707.01961*, 2017.

[16] H. Hu, B. Liu, B. Wang, M. Liu, and X. Wang, "Multimodal DBN for predicting High-quality answers in cQA portals," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2013, pp. 843–847.

[17] M. J. Cafarella and O. Etzioni, "A search engine for natural language applications," in *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 442–452.

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

[19] G. Zhou, Y. Zhou, T. He, and W. Wu, "Learning semantic representation with neural networks for community question answering retrieval," *Knowledge-Based Systems*, vol. 93, pp. 75–83, 2016.

[20] G. Wang, K. Gill, M. Mohanlal, H. Zheng, and B. Y. Zhao, "Wisdom in the social crowd: an analysis of quora," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 1341–1352.

[21] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Discovering value from community activity on focused question answering sites: a case study of stack overflow," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 850–858.

[22] Y. Bengio, Y. LeCun *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[23] T. S. Lee and D. Mumford, "Hierarchical bayesian inference in the visual cortex," *JOSA A. Optical Society of America*, vol. 20, no. 7, pp. 1434–1448, 2003.

[24] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[25] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8599–8603.

[26] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8595–8598.

[27] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[28] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, Feb 2003.

[29] Z. Lu and H. Li, "A deep architecture for matching short texts," in *Advances in Neural Information Processing Systems*, 2013, pp. 1367–1375.

[30] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008, pp. 160–167.

[31] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Advances in Neural Information Processing Systems*, 2011, pp. 801–809.

[32] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," *arXiv preprint arXiv:1412.1632*, 2014.

[33] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2015, pp. 373–382.

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

[35] J. Cheng and D. Kartsaklis, “Syntax-aware multi-sense word embeddings for deep compositional models of meaning,” *arXiv preprint arXiv:1508.02354*, 2015.

[36] M. Yang, Q. Qu, Y. Shen, Q. Liu, W. Zhao, and J. Zhu, “Aspect and sentiment aware abstractive review summarization,” in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1110–1120.

[37] W. Yin, H. Schütze, B. Xiang, and B. Zhou, “Abcnn: Attention-based convolutional neural network for modeling sentence pairs,” *TACL Transactions of the Association for Computational Linguistics*, vol. 4, pp. 259–272, 2016.

[38] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, “Simple question answering by attentive convolutional neural network,” in *Proceedings of the 26th ACL-COLING International Conference on Computational Linguistics*, 2016, pp. 1746–1756.

[39] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, “Recurrent neural network based language model,” in *Proceedings of Interspeech Conference on International Speech Communication Association*, 2010, pp. 1045–1047.

[40] B. Wang, K. Liu, and J. 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)*, vol. 1, 2016, pp. 1288–1297.

[41] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” in *Proceedings of ACL-EMNLP Conference on Empirical Methods on Natural Language Processing*, 2014, pp. 1724–1734.

[43] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," in *Proceedings of the 5th ICLR International Conference on Learning Representations*, 2017.

[44] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proceedings of the 55th ACL Conference on Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 221–231.

[45] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[46] P. C. Nielson and W. Kaufman, "Machine visual inspection device and method," 1988, uS Patent 4,760,444.

[47] G. G. Chowdhury, "Natural language processing," *Annual review of information science and technology*, vol. 37, no. 1, pp. 51–89, 2003.

[48] B. Pang, L. Lee *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[49] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016, pp. 3776–3783.

[50] A. Farzindar and D. Inkpen, "Natural language processing for social media," *Synthesis Lectures on Human Language Technologies*, vol. 8, no. 2, pp. 1–166, 2015.

[51] J. Ko, L. Hiyakumoto, and E. Nyberg, "Exploiting multiple semantic resources for answer selection," in *Proceedings of LREC*, 2006, pp. 1139–1142.

[52] L. Meng, R. Huang, and J. Gu, "A review of semantic similarity measures in wordnet," *International Journal of Hybrid Information Technology*, vol. 6, no. 1, pp. 1–12, 2013.

[53] W.-t. Yih, M.-W. Chang, C. Meek, and A. Pastusiak, "Question answering using enhanced lexical semantic models," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2013, pp. 1744–1753.

[54] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[55] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in Neural Information Processing Systems*, 2013, pp. 2265–2273.

[56] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng, "A deep architecture for semantic matching with multiple positional sentence representations," *arXiv preprint arXiv:1511.08277*, 2015.

[57] Y. Zhang, S. He, K. Liu, and J. Zhao, "A joint model for question answering over multiple knowledge bases." in *AAAI*, 2016, pp. 3094–3100.

[58] M. Tan, B. Xiang, and B. Zhou, "Memory networks," in *Proceedings of the 3th ICLR International Conference on Learning Representations*, 2015.

[59] B. Pan, H. Li, Z. Zhao, B. Cao, D. Cai, and X. He, "Memen: multi-layer embedding with memory networks for machine comprehension," *arXiv preprint arXiv:1707.09098*, 2017.

[60] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 24, no. 4, pp. 694–707, 2016.

[61] J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," *arXiv preprint arXiv:1506.01057*, 2015.

[62] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in

Proceedings of the Conference on Empirical Methods in Natural Language Processing EMNLP. Association for Computational Linguistics, 2011, pp. 151–161.

[63] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International Workshop on Artificial Neural Networks*. Springer, 1995, pp. 195–201.

[64] T. M. Mitchell, *Machine Learning*. WCB/McGraw-Hill Boston, MA:, 1997.

[65] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, p. 947, 2000.

[66] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[67] Y.-l. Boureau, Y. L. Cun *et al.*, "Sparse feature learning for deep belief networks," in *Advances in neural information processing systems*, 2008, pp. 1185–1192.

[68] C. Poultney, S. Chopra, Y. L. Cun *et al.*, "Efficient learning of sparse representations with an energy-based model," in *Advances in neural information processing systems*, 2007, pp. 1137–1144.

[69] S.Kullback and R.A.Leibler, *On Information and Sufficiency*. The Annals of Math. Statistics, 1951, vol. 22.

[70] B. Olshausen and D. Field, *Sparse Coding with an Over-complete Basis Set: A Strategy Employed by VI?* Vision Research, 1997, vol. 37.

[71] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[72] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008, pp. 1096–1103.

[73] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[74] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[75] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proceedings the 6th ICLR International Conference on Learning Representations*, 2015.

[76] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[77] M. Hu, Y. Peng, Z. Huang, X. Qiu, F. Wei, and M. Zhou, “Reinforced mnemonic reader for machine reading comprehension,” *arXiv preprint arXiv:1705.02798*, 2017.

[78] D. Wang and E. Nyberg, “A long short-term memory model for answer sentence selection in question answering,” in *Proceedings of the 53th ACL Annual Meeting of the Association for Computational Linguistics*, 2015, pp. 707–712.

[79] T.Mikolov, S.Kombrink, and L.Burget, “Extensions of recurrent neural network language model,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE*, 2011.

[80] H. Zimmermann, R. Grothmann, A. Schaefer, and C. Tietz, “Identification and forecasting of large dynamical systems by dynamical consistent neural

networks," *New Directions in Statistical Signal Processing: From Systems to Brain*, pp. 203–242, 2006.

[81] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," *Backpropagation: Theory, architectures, and applications*, vol. 1, pp. 433–486, 1995.

[82] A. Robinson and F. Fallside, *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering, 1987.

[83] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[84] J. Chen and N. S. Chaudhari, "Capturing long-term dependencies for protein secondary structure prediction," in *International Symposium on Neural Networks*. Springer, 2004, pp. 494–500.

[85] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," in *arXiv preprint arXiv:1410.5401*, 2014.

[86] S. Sukhbaatar, J. Weston, and R. Fergus, "End-to-end memory networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2440–2448.

[87] L. Qiu, M.-Y. Kan, and T.-S. Chua, "Paraphrase recognition via dissimilarity significance classification," in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2006, pp. 18–26.

[88] Y. Zhang and J. Patrick, "Paraphrase identification by text canonicalization," in *Proceedings of the Australasian language technology workshop*, vol. 2005, 2005, pp. 160–166.

[89] Z. Kozareva and A. Montoyo, "Paraphrase identification on the basis of supervised machine learning techniques," in *Advances in Natural Language Processing*. Springer, 2006, pp. 524–533.

[90] L. Kang, B. Hu, X. Wu, Q. Chen, and Y. He, “A short texts matching method using shallow features and deep features,” in *Natural Language Processing and Chinese Computing*. Springer, 2014, pp. 150–159.

[91] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, “Indexing by latent semantic analysis,” *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[92] S. Fernando and M. 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*, 2008, pp. 45–52.

[93] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A convolutional neural network for modelling sentences,” *arXiv preprint arXiv:1404.2188*, 2014.

[94] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[95] H. Li, M. R. Min, Y. Ge, and A. Kadav, “A context-aware attention network for interactive question answering,” in *Proceeding of the 23nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[96] C. d. Santos, M. Tan, B. Xiang, and B. Zhou, “Attentive pooling networks,” in *CoRR*, vol. abs/1602.03609, 2016.

[97] C. Xiong, S. Merity, and R. Socher, “Dynamic memory networks for visual and textual question answering,” in *Proceedings of the 33rd ICML Conference on International Conference on Machine Learning*, 2016, pp. 2397–2406.

[98] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3128–3137.

[99] C. Shah and J. Pomerantz, “Evaluating and predicting answer quality in community QA,” in *Proceedings of the 33rd International ACM SIGIR Conference*

on Research and Development in Information Retrieval. ACM, 2010, pp. 411–418.

[100] C. Shah, S. Oh, and J. S. Oh, "Research agenda for social Q&A," *Library & Information Science Research*, vol. 31, no. 4, pp. 205–209, 2009.

[101] W. Wu, S. Xu, and W. Houfeng, "Question condensing networks for answer selection in community question answering," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2018, pp. 1746–1755.

[102] X. Zhang, S. Li, L. Sha, and H. Wang, "Attentive interactive neural networks for answer selection in community question answering." in *Proceeding of the 31th AAAI Conference on Artificial Intelligence*, 2017, pp. 3525–3531.

[103] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," in *Proceedings of 54th ACL Annual Meeting of the Association for Computational Linguistics*, 2016.

[104] L. Vilnis and A. McCallum, "Word representations via gaussian embedding," in *Proceedings of the 4th ICLR Conference on International Conference on Learning Representations*, 2015.

[105] A. Severyn and A. Moschitti, "Automatic feature engineering for answer selection and extraction," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2013, pp. 458–467.

[106] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, and J. Prager, "Building watson: An overview of the DeepQA project," *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010.

[107] P. N. Klein, "Computing the edit-distance between unrooted ordered trees," in *European Symposium on Algorithms*. Springer, 1998, pp. 91–102.

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

[109] X. Yao, B. Van Durme, C. Callison-Burch, and P. Clark, "Answer extraction as sequence tagging with tree edit distance," in *Proceedings of NAACL-HLT*. Association for Computational Linguistics, 2013, pp. 858–867.

[110] M. Heilman and N. 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*. Association for Computational Linguistics, 2010, pp. 1011–1019.

[111] W. Yin and H. Schütze, "Convolutional neural network for paraphrase identification," in *Proceedings of the NAACL HLT Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 901–911.

[112] Z. Wang, H. Mi, and A. Ittycheriah, "Sentence similarity learning by lexical decomposition and composition," in *Proceeding of the 25th ACL-COLING International Conference on Computational Linguistics*, 2016, pp. 1340–1349.

[113] R. Socher, C. D. Manning, and A. Y. Ng, "Learning continuous phrase representations and syntactic parsing with recursive neural networks," in *Proceedings of the NIPS Deep Learning and Unsupervised Feature Learning Workshop*, 2010, pp. 1–9.

[114] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," *In Proceeding of the 5th ICLR International Conference on Learning Representations*, 2017.

[115] X. Zhou, B. Hu, Q. Chen, and X. Wang, "Recurrent convolutional neural

network for answer selection in community question answering," *Neurocomputing, Elsevier*, vol. 274, pp. 8–18, 2017.

[116] B. Wang, X. Wang, C. Sun, B. Liu, and L. Sun, "Modeling semantic relevance for question-answer pairs in web social communities," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 1230–1238.

[117] M. L. Littman, S. T. Dumais, and T. K. Landauer, *Automatic Cross-Language Information Retrieval Using Latent Semantic Indexing*. Boston, MA: Springer US, 1998, pp. 51–62.

[118] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[119] Z. Wang, W. Hamza, and R. Florian, "Bilateral multi-perspective matching for natural language sentences," in *Proceedings of the 26th IJCAI International Joint Conference on Artificial Intelligence*, 2017, pp. 4144–4150.

[120] M. Tan, B. Xiang, and B. Zhou, "Lstm-based deep learning models for non-factoid answer selection," in *Proceedings of the 4th ICLR International Conference on Learning Representations (Workshop track)*, 2016.

[121] W.-N. Hsu, Y. Zhang, and J. Glass, "Recurrent neural network encoder with attention for community question answering," *arXiv preprint arXiv:1603.07044*, 2016.

[122] S. Romeo *et al.*, "Neural attention for learning to rank questions in community question answering," in *Proceeding of the 26th ICLR Conference on International Conference on Computational Linguistics*, 2016, pp. 1734–1745.

[123] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," in *Proceeding of the 5th ICLR International Conference on Learning Representations*, 2017.

[124] M. Zhang and Y. Wu, “An unsupervised model with attention autoencoders for question retrieval,” in *arXiv preprint arXiv:1803.03476*, 2018.

[125] M. Hu, Y. Peng, Z. Huang, X. Qiu, F. Wei, and M. Zhou, “Reinforced mnemonic reader for machine reading comprehension,” in *Proceedings of the 27th IJCAI International Joint Conference on Artificial Intelligence*, 2018, pp. 4099–4106.

[126] W. Wang, M. Yan, and C. Wu, “Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2018, pp. 1705–1714.

[127] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, “Disan: Directional self-attention network for rnn/cnn-free language understanding,” in *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, 2018, pp. 5446–5455.

[128] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, “Key-value memory networks for directly reading documents,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1400–1409.

[129] D. Li and A. Kadav, “Adaptive memory networks,” *arXiv preprint arXiv:1802.00510*, 2018.

[130] T. Kenter and M. de Rijke, “Attentive memory networks: Efficient machine reading for conversational search,” *arXiv preprint arXiv:1712.07229*, 2017.

[131] A. Neelakantan, Q. V. Le, and I. Sutskever, “Neural programmer: Inducing latent programs with gradient descent,” *arXiv preprint arXiv:1511.04834*, 2015.

[132] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, “Learning deep structured semantic models for web search using clickthrough data,” in

Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management. ACM, 2013, pp. 2333–2338.

[133] X. Xue, J. Jeon, and W. B. Croft, "Retrieval models for question and answer archives," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 2008, pp. 475–482.

[134] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," *arXiv preprint arXiv:1503.02364*, 2015.

[135] H. Amiri, P. Resnik, J. Boyd-Graber, and H. Daumé III, "Learning text pair similarity with context-sensitive autoencoders," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, August 2016, pp. 1882–1892.

[136] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[137] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[138] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th International Conferenceon machine learning (ICML-10)*, 2010, pp. 111–118.

[139] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557*, 2013.

[140] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[141] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[142] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, vol. 14. Association for Computational Linguistics, 2014, pp. 1532–43.

[143] P.Lamblin and Y. Bengio, "Important gains from supervised fine-tuning of deep architectures on large labeled sets," in *Proceedings of NIPS*2010 Deep Learning and Unsupervised Feature Learning Workshop*, 2010, pp. 1–8.

[144] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[145] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[146] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[147] R. C. S. L. L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, vol. 13. MIT Press, 2001, p. 402.

[148] M. Wang, N. A. Smith, and T. Mitamura, "What is the jeopardy model? a quasi-synchronous grammar for QA," in *Conference on Empirical Methods on Natural Language Processing*, vol. 7. Association for Computational Linguistics, 2007, pp. 22–32.

[149] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[150] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval.* New York: ACM Press; Harlow, England: Addison-Wesley, 2011.

[151] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *Proceedings of the International conference on machine learning*, 2015, pp. 2048–2057.

[152] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, “Teaching machines to read and comprehend,” in *Proceedings of the 28th NIPS Conference on Advances in Neural Information Processing Systems*, 2015, pp. 1693–1701.

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

[154] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in *Proceedings of the 15th INTERSPEECH Annual Conference of the International Speech Communication Association*, 2014, pp. 338–342.

[155] G. P. Styan, “Hadamard products and multivariate statistical analysis,” *Linear algebra and its applications*, vol. 6, pp. 217–240, 1973.

[156] S. Dreiseitl and L. Ohno-Machado, “Logistic regression and artificial neural network classification models: a methodology review,” *Journal of biomedical informatics*, vol. 35, no. 5, pp. 352–359, 2002.

[157] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks.” in *Proceedings of the 30th ICML International Conference on Machine Learning*, vol. 28, 2013, pp. 1310–1318.

[158] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.

[159] S. Wang and J. Jiang, “Machine comprehension using match-lstm and answer pointer,” *arXiv preprint arXiv:1608.07905*, 2017.

[160] B. Dhingra, D. Pruthi, and D. Rajagopal, “Simple and effective semi-supervised question answering,” in *Proceedings of the 16th NAACL-HLT Annual Conference*, 2018, pp. 582–587.

[161] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, “Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension,” in *Proceedings of the 55th ACL Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 1601–1611.

[162] Y. Sasaki *et al.*, “The truth of the f-measure,” *Teach Tutor mater*, vol. 1, no. 5, pp. 1–5, 2007.

[163] C. Clark and M. Gardner, “Simple and effective multi-paragraph reading comprehension,” *arXiv preprint arXiv:1710.10723*, 2017.

[164] C. Xiong, V. Zhong, and R. Socher, “Dcn+: Mixed objective and deep residual coattention for question answering,” *arXiv preprint arXiv:1711.00106*, 2017.

[165] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, “Attention-over-attention neural networks for reading comprehension,” in *Proceedings of the 55th ACL Conference on the Association for Computational Linguistics*, 2017, pp. 593–602.

[166] Y. Shen, P.-S. Huang, J. Gao, and W. Chen, “Reasonet: Learning to stop reading in machine comprehension,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1047–1055.

[167] X. Liu, Y. Shen, K. Duh, and J. Gao, “Stochastic answer networks for machine reading comprehension,” in *Proceedings of the 56th ACL Conference on the Association for Computational Linguistics*, 2018, pp. 1694–1704.

[168] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of the 56th ACL Conference on the Association for Computational Linguistics*, 2018, pp. 2227–2237.

[169] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, “Gated self-matching networks for reading comprehension and question answering,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 189–198.

[170] M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young, “On-line policy optimisation of bayesian spoken dialogue systems via human interaction,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8367–8371.

[171] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.