Query Question Similarity for Community Question Answering System Based on Recurrent Encoder Decoder

by

Borui Ye

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

Waterloo, Ontario, Canada, 2017

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

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

Abstract

The measurement of sentence similarity is a fundamental task in natural language processing. Traditionally, it is measured either from word-level or sentence-level (such as paraphrasing), which requires many lexical and syntactic resources. In order to solve the problem of lacking labelled data and Chinese language resources, we propose a novel sentence similarity framework based on a recurrent neural network (RNN) Encoder-Decoder architecture. This RNN is pre-trained with a large set of question-question pairs, which is weakly labelled automatically and heuristically. Though less accurate, the pre-training greatly improve the performance of the model, also better than other traditional methods. Our proposed model is capable of both classification and candidate ranking. In addition, we release our evaluation dataset – a finely annotated question similarity dataset, which will be the first public dataset under this purpose in Chinese to the best of our knowledge.

Acknowledgements

I would like to express the deepest appreciation to my supervisor, Professor Ming Li, for his patient and valuable guidance throughout these two years and a half time. Without him I couldn't have completed the work and thesis during my Master's period.

I would like to thank the readers of my thesis, Professor Bin Ma and Professor Jesse Hoey, for being willing to spend time reviewing my work, and giving useful suggestions.

A thank you to Anqi Cui, Guangyu Feng, Kun Xiong and Haocheng Qin, who gave me a lot of help and advice in my work.

Also, thank you to my idol Zemin Jiang, who stimulates me to be an exceptional researcher, possibly a professor in the future.

Last but not least, I would like to give my thanks to my family and friends. Thanks to my parents, for kindly support my decision to go abroad to study; thanks to my boyfriend Yuzhi Wang, for giving me countless academic help and mental support; thanks to my friend Yuefei Liu, for staying by my side in my hardest time.

Dedication

This is dedicated to my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

As a Web 2.0 community service, the large user and text base of community Question Answering (cQA) becomes an advantage for us to solve natural language tasks. Nowadays, Chinese language has 900 million first language users as compared to 400 million in English, and textual resources of 5,000 years history as compared to 1,500 for English. The Chinese social messaging app WeChat has 1.1 billion users, as of 2016, with 570 million daily active users. As the nearest report shows, the cQA website Yahoo Answers[1] claimed they hit 300 million (English) questions on July 10, 2012. In contrast, the biggest Chinese cQA website Baidu Knows[2] claimed they have more than 330 million Chinese questions solved as of September 10, 2014.

1.1 Query-Question Similarity

In the field of natural language processing (NLP), a core problem is to find if two sentences have approximately the same meaning. That is, we need to know "Thou art mine" and "You are mine" express the same feeling; and "How old are you?" and "What is your age" are the same question. One of the most typical applications of the problem of question similarity is the cQA system. For example, as depicted in Figure 1.1, when a user queries the cQA system, it first retrieves a list of possible candidate questions from a large database (typically via an indexing service), resulting in only a few hundred questions.

[1] http://answers.yahoo.com/
[2] http://zhidao.baidu.com/

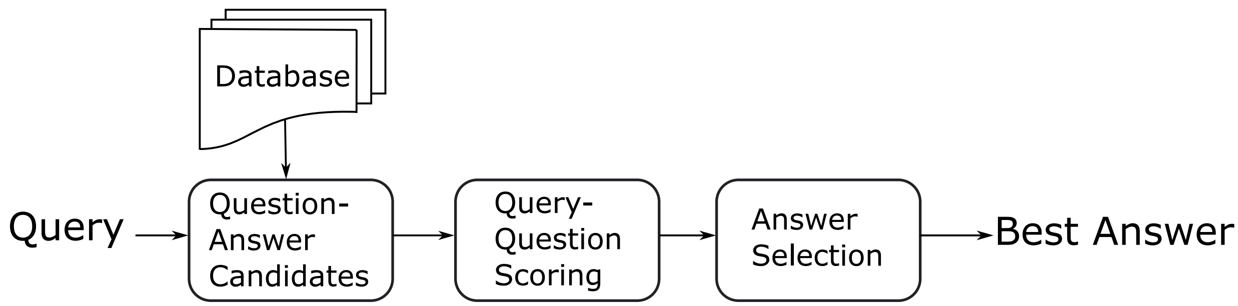

Figure 1.1: Example of a community question answering system.

Then, it selects the most similar question set from the candidate list with a sentence similarity algorithm. The answers to these existing questions are most likely the correct according to the original query. In such a system, the key issue is to determine query-question similarities, i.e., find a question that is most similar to the user's query. More formally, the problem is defined as follows:

> Given a query Q and a set of relevant question candidates $\{C_1, C_2, \ldots, C_n\}$ retrieved from an indexing service, determine whether or not each candidate C_i is similar to Q, and rank them by their similarities to the original query Q.

1.2 Motivation

Determining question similarity is difficult because of the complication of semantics in human languages. For instance, for the query "What must not I feed a dog?", similar questions would be "What can't dogs eat?", and "What food may make dogs sick?". On the other hand, the seemingly similar question "What can I feed a dog?" has just the opposite meanings; simply taking its answer may lead to tragic consequences.

But the research into Chinese sentence similarity is rather limited. First of all, there is no Chinese open data set available for Chinese oriented algorithms. When it comes to English, TREC9 [57] releases 54 groups of similar sentences which are cited by 187 articles. Also, fields closely relate to sentence similarity like paraphrase identification (PI) also have authentic open data sets. Microsoft Research Paraphrase Corpus (MSRP) [13] consists of 4,766 training pairs and 1,725 testing pairs retrieved from news clusters. In 2015, Xu et al.[59] extracted paraphrases from twitter (13,063 training, 4,727 development, 972 testing) for SemEval 2015 that attracts 18 teams taking part. Secondly, Chinese lexical resources and natural language processing is more difficult than English. Besides the two hard-to-solve problems [19]: (1) different words may have the same meaning, and (2) different word orders may cause different meanings, the Chinese word segmentation problem reduces the accuracy of all succeeding processes. Worse still, the Chinese Thesaurus resources, such as LTP-Cloud [8] and HowNet [14], are far from complete because of the versatility of Chinese synonyms and fast emergence of Internet out-of-vocabulary words. Also, as findings of [9] reveal, even the best Chinese sentence parser can achieve an accuracy of only 83.1% (as compared to significantly above 90% for English), and such error can propagate in sentence similarity task.

1.3 Contribution

In this paper, we design an Encoder-Decoder recurrent neural network (RNN) training framework to solve this key problem of query-question similarity, which does not rely on any external linguistic resources (except for segmentation). The goal is maximising the probability of first sentence given the second similar one, while minimising the probability of the first sentence given the second dissimilar one. The model solves the similarity problems in three major ways: (1) We use a pre-trained word vector set that captures semantic relatedness among words. (2) The sequential structure of RNN enables word order recognition and it adaptively learns to update information from the previous word using *reset* and *update* gates. (3) In order to deal with the lacking labelled data problem, We have developed a distant supervision learning scheme: We use some hand-crafted features to automatically label a large amount of query-question pairs, which are then used for training. Experiments show that the pre-training boosts the performance of our model.

Thus, to summarise, the main contributions of this paper are: (1) To the best of our knowledge, it is the first time that an RNN Encoder-Decoder is applied to sentence similarity task. (2) We design a semi-supervised framework to capture both textual similarity and semantic similarity. (3) Finally, we extract a Chinese sentence similarity corpus, and make it publicly available for other researchers.

1.4 Thesis Organisation

The rest of this thesis is organised as follows. In Chapter 2, we discuss work related to our sentence similarity task to the best of our knowledge. In Chapter 3, we give the introduction of our Chinese sentence similarity corpus. In Chapter 4, we illustrate the model structure and training method, including parameter tuning and training pipeline. In Chapter 5, we did some experiments, comparing our model to our baselines. In Chapter 6, we discuss the outcome of our experiments. In Chapter 7, we make suggestions to our future improvement and other models that may be efficient to solve the problem.

Chapter 2

Related Work and Background

In our task, we mainly focus on the query-question similarity problem, which can be abstracted into a sentence similarity problem. However, we use neural network models to solve the problem. Thus, our work mainly connects to four main areas: sentence similarity, paraphrase identification, word vectors, and sequence-to-sequence learning. Also, query-question similarity, sentence similarity and paraphrase share some difference and commonalities:

- Area: query-question similarity aims at estimating the similarity among sentence pairs in community question answer area. But for general sentence similarity and paraphrase identification, the training corpus needn't be in the same fields. For example, the Microsoft Paraphrase Corpus (MSRP) was extracted from news clusters. TREC-9 and SemEval 2016 used cQA data. SemEval 2015 sentence similarity task used Twitter posts.

- Sentence length: for query-question similarity and sentence similarity, the sentences are relatively shorter. For example, for our query-question similarity corpus, the average length for each sentence is 8. But for MSRP, the average length is 20. This can make the approaches toward these problems different. For paraphrase identification, using large scale model is better, as the model needs higher level of representation. But for sentence similarity, we always use light weight neural network models, or simpler hand crafted features. Note that approaches for paraphrase identification may not be suitable for sentence similarity, as for some model, short sentences will not be able to feed into the model. An example will be used to illustrate this in the experiment chapter.

- Similarity criteria: last but not the least, the similarity criteria are different. For query-question similarity, we divide our similarity level into three categories (similar, relevant, dissimilar). Whereas Agirre et al.[2] defined six-level similarity levels: (completely equivalent, mostly equivalent, roughly equivalent, not equivalent, not equivalent but same topic, different topics). The details of our similarity level is discussed in the corpus chapter.

2.1 Sentence Similarity

Many of the existing studies for sentence similarity are mainly based on feature engineering, where derived forms of string comparison algorithms are used as to calculate "similarity score" of the two sentences. These algorithms' results are either linearly combined or concatenated as a feature vector for classification. As Achananuparp et al. [1] has mentioned, sentence similarity algorithms are basically classified into three categories: word overlap measures, corpus based measures, and linguistic measures.

Word overlap measures mainly calculates similarity by the common words of the two sentences. Metzler et al.[38] and Allan et al.[3] count the common words and normalise the result using sentence length. The measure is easy to implement, but it fails to capture other information. For example, "I love pets" and "I don't love pets" actually have different meanings, but they will be considered similar using word overlap method. Also, this method does not take word meaning into account. There are different words with the same meanings, but will not be considered.

Corpus based measures mainly use bag-of-words model, mapping sentences into vectors. Then cosine distance is used. Lund et al.[37] promoted the concept of hyperspace analogue to language (HAL), using corpus to calculate co-occurrence word representation, and measure the similarity of sentences using cosine distance. The method is more effective than the simple overlap method, but as Blacoe et al.[5]'s experiment points out, the co-occurrence based word vectors is far less effective than neural network based ones.

Linguistic measures aims at the semantic similarity among words. Li et al.[36] use hierarchical semantic knowledge base to capture word similarities, linearly combined with word order similarity. But thesaurus based methods heavily rely on the quality of our dictionary. As the fast emerging of Internet new words, out-dated lexical resources will cause the algorithm fail.

All of the methods are based merely on word-level similarity, with no regard to sentence structures.

2.2 Paraphrase Identification

The problem of sentence similarity is similar to paraphrase identification (PI), however, they differ from each other, for two questions that are non-paraphrase can still be considered similar, as long as they are on the same topic and share the same answer. Nevertheless, the methods share commonalities.

With the introduction of word vectors, many works have successfully solved the problem in distributional semantic space. Also, by designing various kinds of neural network structures, models are able to solve the problem in sentence level. Some previous work use pre-trained word vector set from neural networks and specific sentence composition methods to detect similarity information. E.g., Socher et al. [49] use recursive auto-encoder (RAE) to train their word vectors, where sentences are parsed into syntactic trees, and construct similarity matrices for classification. Kiros et al. [28] train their contextual skip-gram word vectors over RNN, and use simple compositional methods to compute the sentence representation. They tested their word vector quality on the MSRP task.

Other work embed sentence vectors into self-defined deep learning structures. E.g., Wang et al. [58] use a two-channel convolutional neural network (CNN) to discriminate similar and dissimilar components, then different components are filtered and dimensionality reducted to feed into a softmax classifier. Mueller et al.[42] use two long short-term memory (LSTM) structures to compose two sentences, they take the last output of each LSTM layer, and calculate the Manhattan distance between the two vectors. He et al. [24] use CNNs to extract multiple similarity granularities and pooling ways for similarity comparison. All these models are considered "Siamese" [7], where two sentences are processed in parallel. Different from these models, we apply the sentence similarity to a sequence to sequence (seq2seq) model, and prove that it can also achieve impressive results besides those Siamese models. Due to the different background and languages (English vs. Chinese), we do not directly compare with the above mentioned models. Although the SemEval 2016 Task 3[1] is quite similar to our task, they require taking question description into account, which does not fit our task.

2.3 Word Embedding

In the similarity tasks, a very important point is to map words into semantic space. That is, we need to get vector representations of a words, where semantically similar words tend

[1] http://alt.qcri.org/semeval2016/index.php?id=tasks

to have smaller cosine distance. By doing so, we can avoid the limitations of thesaurus lexicons. An picture example is shown in Figure 2.1[2] to illustrate the attribute of word vectors. In this picture, word vectors are cast into two-dimension space for virtualisation. Words with the same meaning or context appear to be in the same cluster. This is only a simple example to show how word vectors work, in practice, word representations appear to be more powerful when trained with higher dimension.

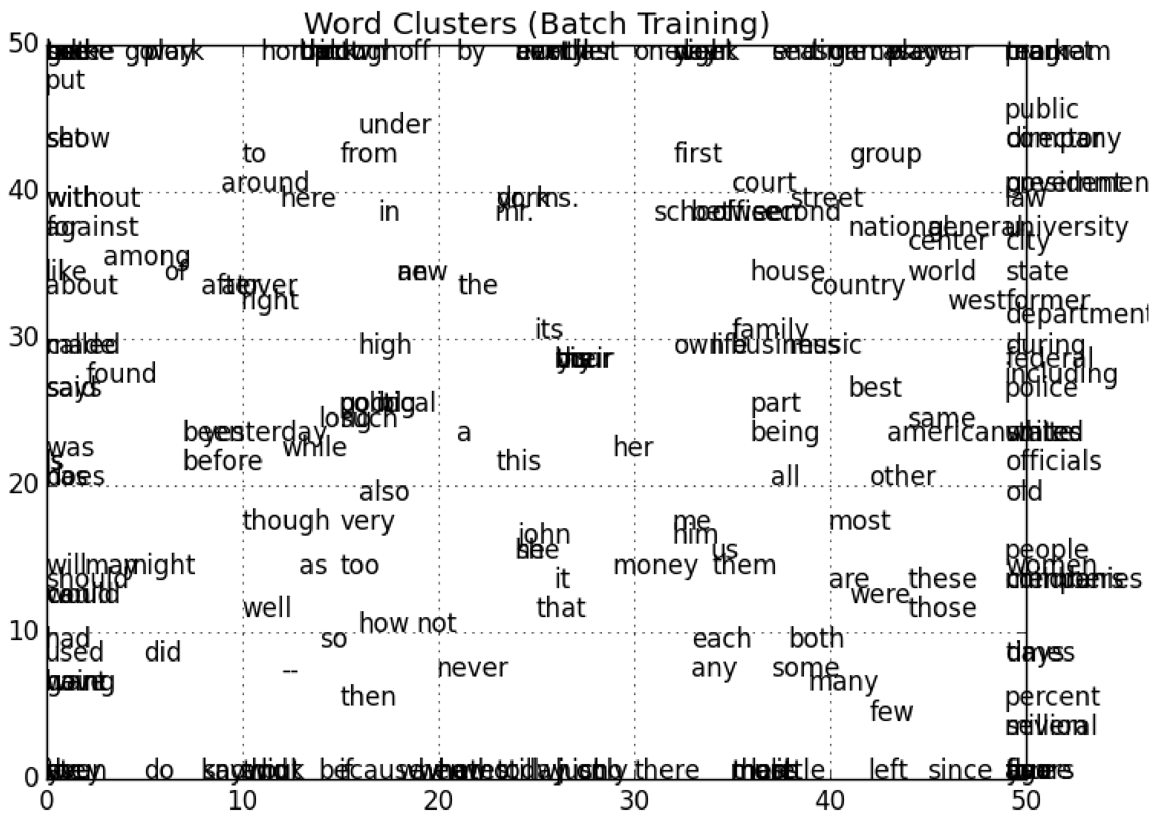

Figure 2.1: Example of word vector clusters.

The most basic way to construct word vectors is using one-hot-encoding. That is, given a word dictionary, for each word, we give an unique index. Then every word's vector size

[2]Image source: http://sujitpal.blogspot.ca/2014/10/clustering-word-vectors-using-self.html

equals to the size of the dictionary. The vector is almost all-zero except for the word's index position, which is one. The method is easy to implement, but it has a main drawback. That is, one-hot representation cannot capture the relationship among words. Words with similar meanings will be treated as different ones. Among all these years, researchers have endeavored to train word vectors that can better capture semantic information, so as to better relate words. Mitchell et al.[41] assume that word meaning can be learned from the linguistic environment. They build word vectors based on word co-occurrence. They first obtained a set of representative words as dimension of word vector, and let each element of word vectors stands for the weighted co-occurrence value of the relevant word. Landauer et al. [33] used Latent Semantic Analysis, they construct a word document co-occurrence matrix using their document collections, then they use singular value decomposition (SVD) to extract eigenvalues. The words relation is also calculated in this process. There are also probabilistic models. Blei et al. [6] and Griffiths et al. [23] represented words as probability distribution over different topics. The method is called Latent Dirichlet Allocation (LDA).

Another way to obtain word vectors is through neural network. Bengio et al. [4] trained their language model via a three-layered neural network. They used a word embedding matrix as one of the parameters, and updated it during the unsupervised training process. In the end, the word embedding matrix consists of sentiment carrying word vectors. Different from Mitchell et al's [41] work, Bengio et al.'s [4] word vectors capture both semantic and syntactic feature. Mikolov et al. [39] further Bengio et al.'s [4] work by taking out non-linear hidden layer, which is very time consuming. They proposed two new models: Continuous Bag of Words, Skip-gram. The former model take context as input layer and corresponding word as output layer, the latter one does the other way round. In order to take more sentence information into account, Socher et al. [49] introduce syntactic tree structure into their word vector training. They used recursive auto-encoder over tree structure and use minimising the reconstruction error. Apart from unsupervised ways, Socher et al. [51] annotated 200, 000 phrases generated by Stanford Parser, and feed them into their neural tensor network for sentiment polarity analysis. Their word vectors are trained in this process.

2.4 Sentence Compositionality Methods

In many natural language processing tasks, a crucial problem is how to map sentences to vector space, while losing as less information as possible. Because of different sentences' lengths, simply concatenate the word vectors will not be possible. The most commonly used way is to design approaches that can merge word vectors into sentence vectors. There are many linear methods to compose sentences or phrases. Mitchell et al. [41] pointed out 9 phrasal composition methods (addictive, multiplicative, dilation, etc.), and did some experiments on subject similarity ratings. The parameters in the multiplicative and dilation model is trained using their dataset. They found that multiplicative and dilation model out-perform others. But these methods do not take sentence structure information into account, which means sentence will be merged in the same way regardless of its inner content and structure.

Non-linear methods uses neural network. Socher et al. [49] used recursive auto-encoders, aiming at finding parameter to merge two vectors to one without losing information. To minimise reconstruction errors, they used greedy algorithm to construct the document tree, and added a softmax layer to each tree layer to predict sentiment distribution. Le et al. [35]'s approach is inspired by Mikolov et al. [39]'s continuous bag-of-words model, where they add document representation as one of the input. Similar as the original language model based structure, they also used a probabilistic output over the dictionary. Apart from recursive structure, recurrent neural network (RNN) and convolutional neural network (CNN) can also be used to compose sentence vectors. Tang et al. [54] use long short term memory (LSTM) or CNN for their first step to compose word vectors into sentence vectors, in the following step, they use bidirectional RNN for document modelling.

2.5 Pairwise Learning

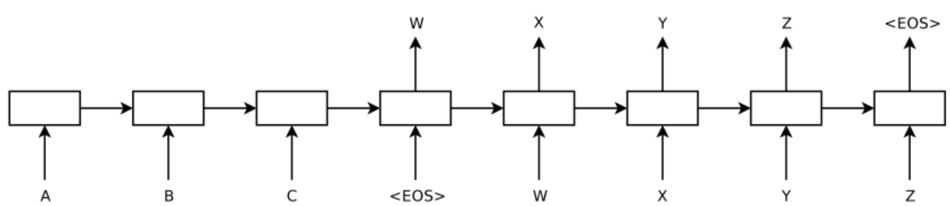

Figure 2.2: Example of RNN encoder decoder for machine translation.

In classical machine learning classification tasks, for each sample, there will be only one input, which is usually a feature vector, fed into the classifier to get the classification results. But in tasks such as machine translation, dialogue generation, and sentence similarity, we need to think of other approaches to fit our task. Our core model, the RNN Encoder-Decoder is used in a wide range of seq2seq applications. E.g., Kiros et al. [28] encode context into thought vectors and decode the succeeding sentence, where they trained context-aware word vectors. Cho et al. [10] and Sutskever et al. [53] apply this to machine translation tasks, while both the encoder and the decoder learn language models during pre-training, and the model is able to decode French given English encoding. See Figure 2.2[3] for example, sequence "ABC" represents source language sentence, while sequence "WXYZ" represents target language sentence. Shang et al. [48] use RNN Encoder-Decoder for their dialogue machine, responding with the output of decoder. Apart from these applications, we use this model for the sentence similarity task, where we also have two sentences, fed into the encoder and the decoder respectively. There are also many other pairwise model other than RNN encoder decoder. For example, Feng et al. [17] use two CNNs to embed querys and answers, connected by a cosine similarity layer to calculate the relevance. The model is used for answer selection. Similarity, Mueller et al. [42] use two LSTM layers to embed two queries, connected by a Manhattan distance layer to calculate sentence similarity.

2.6 Data sets

There have been a few English sentence similarity datasets. TREC-9 released a dataset[4] of variants of questions. It included 54 groups of a total of 260 questions. Within each group, the questions are considered paraphrases. Microsoft Research Paraphrase Corpus (MSRP) [13] consists of 4,766 training pairs and 1,725 testing pairs retrieved from news clusters. SemEval [2, 59, 43] have had semantic similarity tasks since 2012. Each year's dataset has a different focus.

Despite the sufficient English corpus, Chinese sentence similarity corpus is far from enough. As far as we know, there is no accurately annotated and neatly pre-processed corpus, that is ready for Chinese natural language processing researchers to use. In 2015, Sogou Inc.[5] hosted a query-title matching contest, where they have 20, 000 annotated sentence pairs available for use. But the corpus is not finely-annotated, what is worse, the corpus contains much junk information, which makes researchers hard to use for Chinese sentence similarity tasks. For example, for query like "电脑怎么会关机 (why my computer shuts down)", many candidates will look like "电脑自动关机与重启是什么原因-太平洋IT百科 - 产品报价 - 太平洋电脑网 (why my computer restarts automatically - price listings - Ocean Computers Inc.)". The results are automatically retrieved from google

[3]Image source: papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[4]http://trec.nist.gov/data/qa/T9_QAdata/variants.key

[5]www.sogou.com

search results, which will cause the search title will also include website information. This can be distinguished by human, but not computer programmes. Also, there are many wrongly annotated pairs. In order to solve the lacking of data problem, we create a Chinese query-question matching corpus consists of 4,322 records, with 1,346 similar pairs, 2,147 dissimilar pairs and 829 relevant pairs. We also made the data public. We will discuss our corpus in detail in Chapter 3.

Chapter 3

The Annotated Query-Question Corpus

Our annotated corpus[1] is extracted from question-answer (QA) pairs, crawled from some cQA websites in China. The query and question pairs are formed by the following steps:

(1) We crawl 500,094,738 QA pairs from two of the biggest Chinese cQA websites, Baidu Zhidao and Sogou Wenwen[2]. Then we build a word-based inverted index on the questions using Apache Solr[3].

(2) We randomly select 400 different questions (topics) from our database, and for each question, we take it as user's query and search in the indexing service, retrieving at most 100 results. The retrieved questions are considered as candidates to match with the query (examples shown in Table 3.1).

As a keyword search method, the sentences within the same topic group share some same words, but not necessarily the same meaning. After eliminating the duplicates we get 10,000 sentence pairs. Due to the popularity of questions in the cQA websites, some topics have mostly the same or very different questions, which bring little help for training. Hence we remove topics with $> 90\%$ or $< 10\%$ similar pairs. As a result, the remaining topic groups are of mostly small sizes: Half of them contain fewer than 30 pairs, while the second half contain $30 - 80$ pairs. Also, half of the groups contain fewer similar pairs ($< 30\%$), showing that not all questions are popular.

[1]Available at: https://cs.uwaterloo.ca/~b7ye/corpus.html

[2]Baidu Zhidao: http://zhidao.baidu.com/, Sogou Wenwen: http://wenwen.sogou.com/

[3]http://lucene.apache.org/solr/

Table 3.1: Examples of queries and candidates.

	Example 1	**Example 2**	**Example 3**
Query	1 平方公里等于多少平方米 How many square meters equal to 1 square kilometer	非洲包括哪些国家 Which contries does Africa include	flash制作要下载那些软件 Which software should I download to make flash (*videos*)
Candidates	一平方公里等于多少平方米 How many square meters equal to one square kilometer	非洲有什么国家 Which countries does Africa have	下载 FLASH 制作软件 (*How to*) Download FLASH-producing software
	0.8 平方公里等于多少平方米 How many square meters equal to 0.8 square kilometer	非洲最大的国家是哪个 Which is the largest country in Africa	哪能下载制作 Flash 的软件? Where can I download Flash-producing software?
	20 公顷 300 平方米等于多少平方米 How many square meters equal to 20 hectares and 300 square meters	西方国家包括哪些 Which countries does the Western world include	谁会制作 Flash Who knows how to produce Flash (*videos*)

(3) For the remaining pairs, we annotate them into one of the three similarity labels (examples shown in Table 3.2):

- **Similar.** Pairs that share the same meaning, or the user is actually asking for the same thing.
- **Relevant.** Pairs that are mostly similar and answer to these two questions are mostly the same, but the two sentences differ in some details.
- **Dissimilar.** Pairs that have different meanings, or the user expect different answers although the sentences are mostly similar.

We ask three human judges to label the sentence pairs, with a Fleiss' Kappa [20] of 82.84%. We did majority votes for each pair, with 21 undecided pairs annotated by a

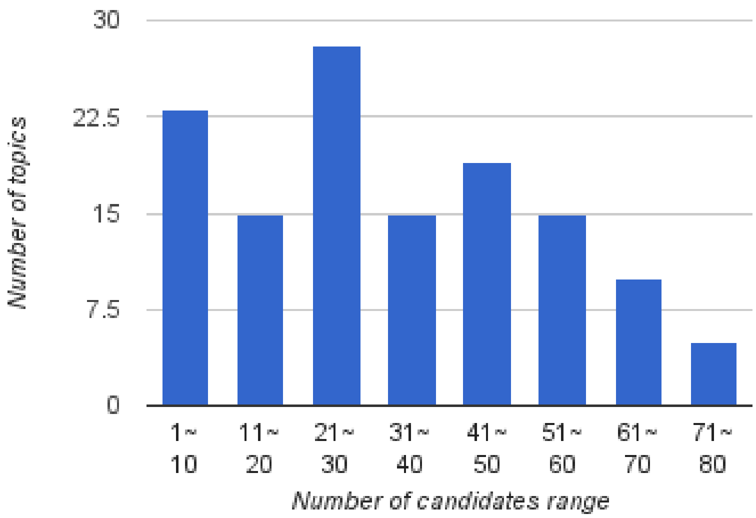

Figure 3.1: Number of candidates' count of our corpus. For example, the first column means we have 23 topics which have candidates number ranging from one to ten. We can see most of the topics have number of candidates ranging from 1 to 60, the minority have candidates more than 60.

fourth annotator. Finally we get 4,322 records, with 1,346 similar pairs, 2,147 dissimilar pairs and 829 relevant pairs. We believe this set of pairs cover a wide range of queries and questions on the Internet: The sentences have a length of $[3, 123]$ characters with an average of 16, a median of 13, and of $[1, 76]$ words with an average of 8, a median of 7. Figure 3.1 and Figure 3.2 show some statistics of our corpus.

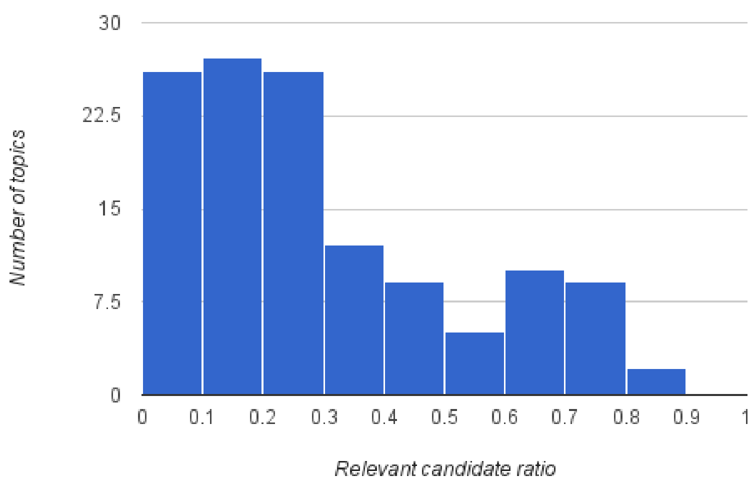

Figure 3.2: Relevant candidate ratio count of our corpus. This shows for each topic we have relevant candidates, divided by total number of candidates, the result is called relevant candidate ratio. For example, the first column means we have 25 topics which have relevant candidate ratio ranging from 0 to 0.1.

Table 3.2: Examples of question pairs and their labels, with explanations.

Question 1	**Question 2**	**Explanation**
	Similar	
宝来和高尔夫那款好 Which one is better, Bora or Golf	买高尔夫好还是新宝来好? Should I buy Golf or the new Bora?	Same meaning.
耳机有回音怎么去除 How to erase the echoes of my headset	电脑耳机有回音啊，求救 There are echoes in my computer headset. Help!	Same intention.
芦荟胶囊能治痘痘么 Can aloe capsules heal pimples	脸上长痘吃芦荟胶囊有效吗 Will it work if I take aloe capsules when I have pimples on my face	Same meaning (with word variations).
JavaScript和java什么关系 Relation between JavaScript and Java	JavaScript和java什么联系 Connection between JavaScript and Java	"Relation" and "Connection" are synonyms.
	Dissimilar	
win7怎么设置自动关机 How to set automatic shutdown for win7	win7关机卡在正在关机不动了。 win7 is stuck at "shtting down" screen	Different meanings (although sharing common words).
在那遥远的地方原唱 Singer of (*the song*) "The place faraway"	什么地方最遥远? What place is the most faraway?	Different topics.
女属虎的和男属兔配吗 Does a woman born in the year of Tiger match a man born in the year of Rabbit	男属虎女属兔相配吗 A man born in the year of Tiger, a woman born in the year of Rabbit, do they match	Different meanings (caused by word ordering).
	Relevant	
猫这个单词怎么写 How to spell the word cat	"野猫"的英语单词怎么写 How to spell the English word for "wild cat"	Minor details differ.
儿童吃什么有助长高 What food will help children grow taller	小孩子助长用什么好? How to make children grow taller	The answer of the latter one includes the former one's.

Chapter 4

The RNN Similarity Model

As mentioned before, our model is based on an RNN Encoder-Decoder which outputs a probabilistic distribution over different similarity classes. It is also pre-trained with some weakly labelled data. In this section, we illustrate the architecture of our RNN Encoder-Decoder, and also list the heuristic similarities for automatic labelling.

4.1 Preliminaries

In order to let reader fully understand our model, we briefly introduce RNN in this section. The content will include RNN's applications in natural language processing, how it updates its parameters, and some of its variations.

4.1.1 Recurrent Neural Networks

Recurrent neural networks have achieved great success in many natural language processing tasks including text generation [52], machine translation [10], speech recognition [22], and image description [27]. Different from the traditional feed-forward neural networks which previous layers' outputs will feed only into the next layer (example is shown in Figure 4.1), recurrent neural networks will feed the output back into the previous layer's input. This is called feedback neural networks (example is shown in Figure 4.2). Feed-forward neural networks work well for all kinds of classification and regression tasks, but they fail to solve the sequential data. For example, given a word sequence, we want to predict the word that comes after, we need not only know the set of words, but also the words' order,

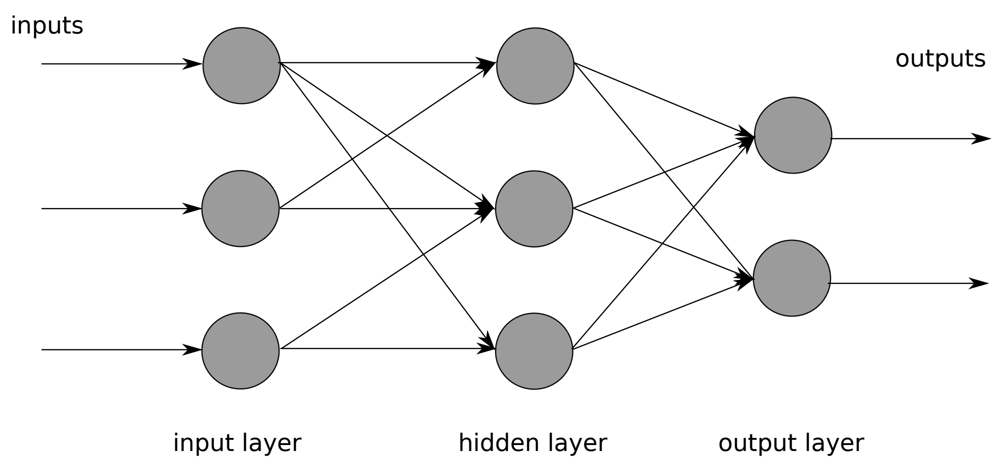

Figure 4.1: Example of a feed-forward neural network.

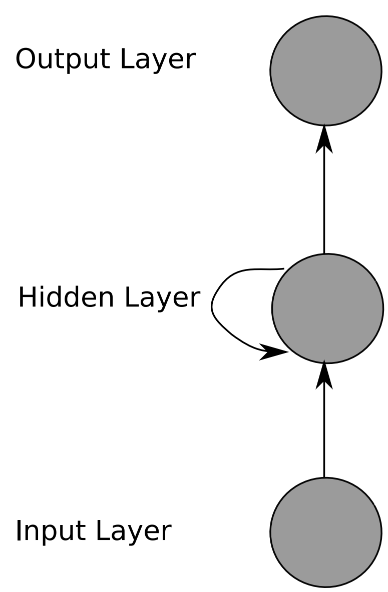

Figure 4.2: Example of a recurrent neural network.

because the probability of a word showing up in a sentence depends on the words that come before it. RNN's recurrent nature can beautifully solve the problem, as it memorises the previous information and applies that to the current computation. This way, the nodes in the hidden layer are connected, as RNN need to take both input layer's input and hidden layer's output from the previous time unit. If we unfold the structure in Figure 4.2, we can see how RNN behaves in each time step.

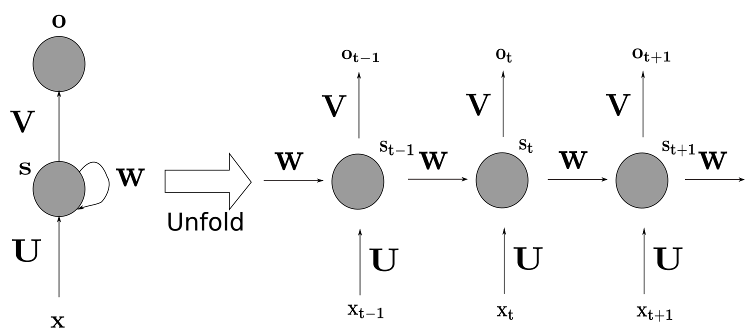

Figure 4.3: Example of a recurrent neural network after unfolding. o is the output, U, V, and W are RNN's weights, s is hidden layer, x is input, t is time unit counter.

4.1.2 Back-propagation Through Time

According to Figure 4.3, we can define RNN's formula as follows:

$$\mathbf{s_t} = tanh(\mathbf{W s_{t-1}} + \mathbf{U x_t} + \mathbf{B}) \tag{4.1}$$

$$\mathbf{o_t} = \mathbf{V s_t} \tag{4.2}$$

where $\mathbf{s}$ is hidden layer, $tanh$ is activation function, $\mathbf{W}, \mathbf{U}, \mathbf{V}$ are parameters, $\mathbf{B}$ is bias, $\mathbf{x}$ is input, $\mathbf{o}$ is output, t is time unit. If the input's dimension is 200, output dimension is 8000, hidden layer dimension is 500, we have: $\mathbf{s} \in \mathbb{R}^{500}$, $\mathbf{W} \in \mathbb{R}^{500\times500}$, $\mathbf{U} \in \mathbb{R}^{500\times200}$, $\mathbf{V} \in \mathbb{R}^{8000\times500}$, $\mathbf{B} \in \mathbb{R}^{500}$, $\mathbf{x} \in \mathbb{R}^{200}$, $\mathbf{o} \in \mathbb{R}^{8000}$.

For example, given a sentence with word sequence $\{w_1, w_2, ..., w_{n-1}\}$, we want to predict the next word w_n. We mapped the word sequence into word vector sequences: $\{x_1, x_2, ..., x_{n-1}\}$, we also map the objective word into one-hot vector $\mathbf{y}$, where $\mathbf{y} \in \mathbb{N}^{8000}$. For all the sentence in our training set, the learning objective is to minimise the cross-entropy:

$$L(\theta) = -\sum_{i=1}^{N}\sum_{j=1}^{8000} y_i^j \log o_i^j \tag{4.3}$$

where y_i^j is jth dimension for ith sentence's one-hot label, o_i^j is jth dimension for ith sentence's output.

Back-propagation Through Time The parameters are shared within different time steps, so we only need to calculate the derivatives from output to input. We use chain rule and take $\mathbf{W}$ as example:

$$\begin{aligned}\frac{\partial L(\theta)}{\partial \mathbf{W}} &= \frac{\partial L(\theta)}{\partial \mathbf{o_n}} \frac{\partial \mathbf{o_n}}{\partial \mathbf{s_t}} \frac{\partial \mathbf{s_t}}{\partial \mathbf{W}} \\ &= \frac{\partial L(\theta)}{\partial \mathbf{o_n}} \frac{\partial \mathbf{o_n}}{\partial \mathbf{s_t}} \frac{\partial \mathbf{s_t}}{\partial \mathbf{s_{t-1}}} \frac{\partial \mathbf{s_{t-1}}}{\partial \mathbf{s_{t-2}}} \cdots \frac{\partial \mathbf{s_3}}{\partial \mathbf{s_2}} \frac{\partial \mathbf{s_2}}{\partial \mathbf{s_1}} \frac{\partial \mathbf{s_1}}{\partial \mathbf{W}}\end{aligned} \tag{4.4}$$

where

$$\frac{\partial L(\theta)}{\partial \mathbf{o_n}} = -\frac{\mathbf{y}}{\mathbf{o}ln2} \tag{4.5}$$

$$\frac{\partial \mathbf{o_n}}{\partial \mathbf{s_t}} = \mathbf{V}^T \tag{4.6}$$

$$\frac{\partial \mathbf{s_t}}{\partial \mathbf{s_{t-1}}} = (1 - tanh^2(\mathbf{W}\mathbf{s_{t-1}} + \mathbf{U}\mathbf{x_t} + \mathbf{B}))\mathbf{W} \tag{4.7}$$

$$\frac{\partial \mathbf{s_1}}{\partial \mathbf{W}} = (1 - tanh^2(\mathbf{W}\mathbf{s_0} + \mathbf{U}\mathbf{x_1} + \mathbf{B}))\mathbf{s_0} \tag{4.8}$$

then

$$\frac{\partial L(\theta)}{\partial \mathbf{W}} = -\frac{\mathbf{y}}{\mathbf{o}ln2}\mathbf{V}\prod_{i=1}^{t}((1 - tanh^2(\mathbf{W}\mathbf{s_{i-1}} + \mathbf{U}\mathbf{x_i} + \mathbf{B}))\mathbf{W}^{t-1}\mathbf{s_0} \tag{4.9}$$

finally, we apply changes to W:

$$\mathbf{W} < -\mathbf{W} - \eta\frac{\partial L(\theta)}{\partial \mathbf{W}} \tag{4.10}$$

Vanishing Gradient Problem Even though the back-propagation through time is commonly used in the RNN training, it suffers from the "Vanishing Gradient Problem". The derivatives for each parameters will approximate zero as the length of sentences increase. Consider the multiple multiplication part in Equation 4.9:

$$\prod_{i=1}^{t}(1 - tanh^2(\mathbf{W}\mathbf{s_{i-1}} + \mathbf{U}\mathbf{x_i} + \mathbf{B})) \tag{4.11}$$

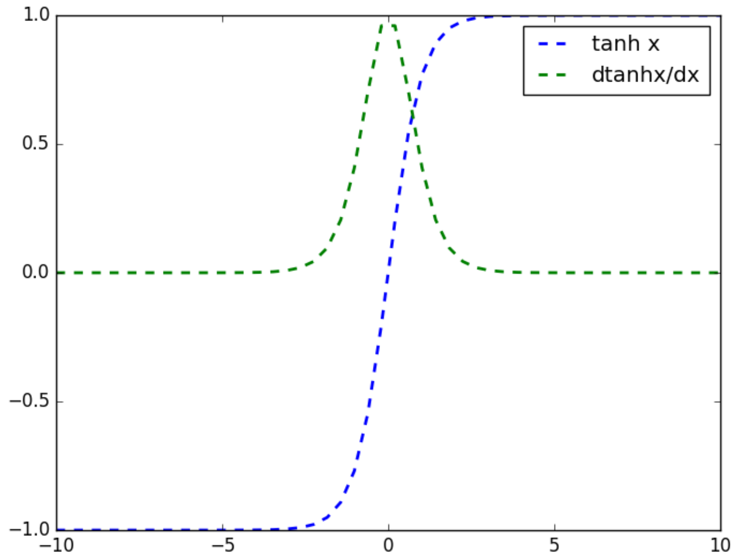

Figure 4.4: tanh and derivative

The *tanh* function and its derivative function are shown in Figure 4.4. We can see that *tanh*'s derivative function ranges from zero to one. When the sentence's length is very long (i.e. t in Equation 4.11 is very large), multiple multiplications' value will shrink dramatically. Thus, the parameter will be updated very slow.

4.1.3 Applications

Language Model and Sentence Generation

Definition 4.1.1. Language Model A statistical language model is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability $P(w_1, w_2, ..., w_n)$ to the whole sequence.

If we connect RNN's last time step's output to a softmax layer, which output the last word w_n's probability distribution over the dictionary. Thus, after using big data to train RNN's parameters, we can predict a sentence's probability. Also, given a seed word, we can even generate an article. An example is shown below [52]:

The meaning of life is the tradition of the ancient human reproduction: it is less favourable to the good boy for when to remove her bigger. In the show's agreement unanimously resurfaced. The wild pasteured with consistent street forests were incorporated by the 15th century BE. In 1996 the primary rapford undergoes an effort that the reserve conditioning , written into Jewish cities, sleepers to incorporate the .St Eurasia that activates the population. Mar??a Nationale, Kelli, Zedlat-Dukastoe, Flrendon, Ptu's thought is. To adapt in most parts of North America, the daynamic fairy Dan please belives the free speech are much related to the

Machine Translation Machine translation aims at translating a source sentence to target sentence. For example, translating English into Chinese. Different from language model, machine translation needs to take all the input from source language and then output then target language. That is, we need to get all of the input sequence in order to output even the first word of the target language. An example of RNN translation model is shown in Figure 4.5.

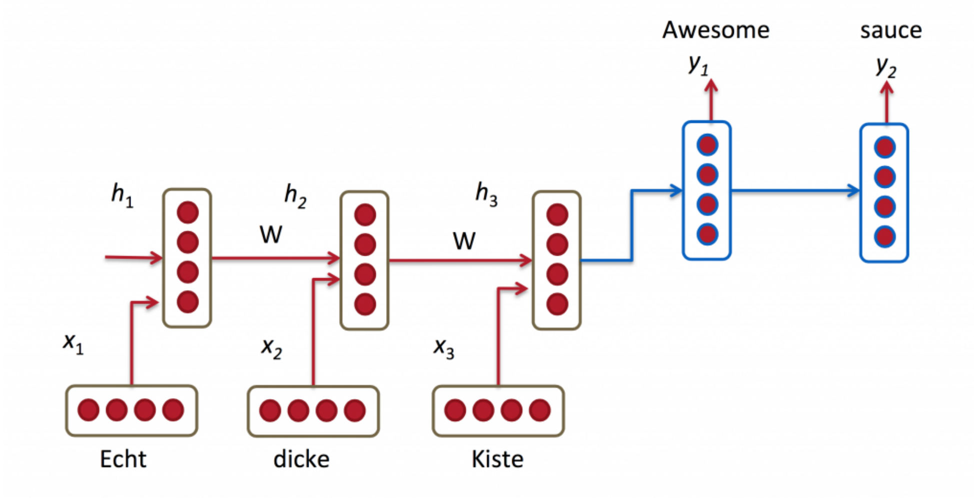

Figure 4.5: Machine translation model. Image source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Image Caption Generation Image caption generation is given a image, you need to descriptive caption about the image. Karpathy et al. [27] designed a model combining

CNN and RNN, where CNN is used for image feature extraction, and RNN is for sentence generation. An example of RNN's results is shown in Figure 4.6.

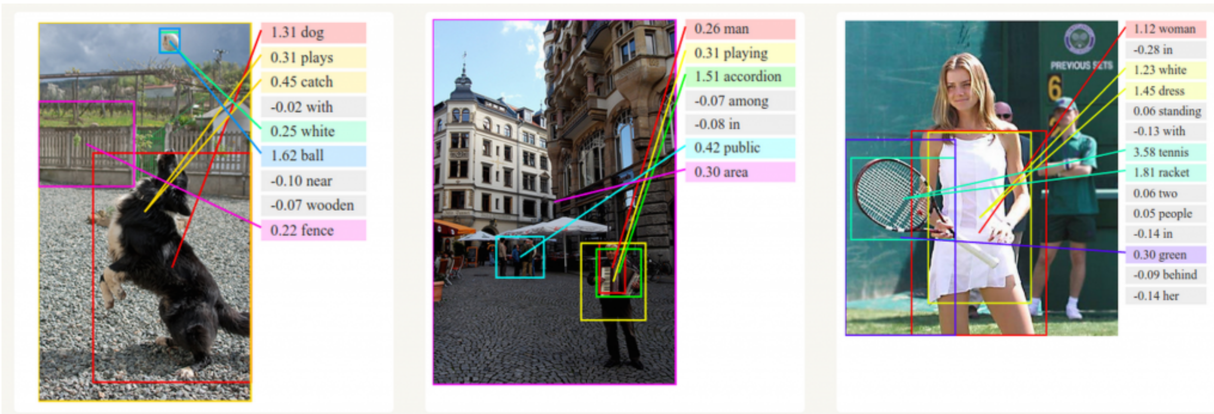

Figure 4.6: Example of image description. Image source: http://cs.stanford.edu/people/karpathy/deepimagesent/

4.1.4 Gated Recurrent Unit and Long Short Term Memory

Gated recurrent unit (GRU) and long short term memory (LSTM) are two variations of RNN, the difference is the formula to calculate hidden state is more complex.

Gated recurrent unit An illustration of GRU's structure is shown in Figure 4.7 [10]. It has two gates: the update gate (z) and the reset gate (r). The update gate controls how much information should be carried from the previous hidden state to the current one, while the reset gate controls if the previous information should be remembered. If the reset gate is close to zero, the network is forced to forget all the previous state. The structure works as follows: GRU firstly compute the hidden state according to the current input vector, and then uses this information to compute update gate and reset gate. Then, it uses current reset gate, word vector and previous hidden state to calculate new memory content. The final hidden state is the linear combination of previous hidden state and the new memory content. In this way, GRU improves RNN in two aspects: 1. Within a word sequence, different words are in the different positions, they have different effect on the current hidden layer. The farther the word is from the current word, the less effect it has on the current hidden layer. That is, GRU weighted the previous hidden states. The

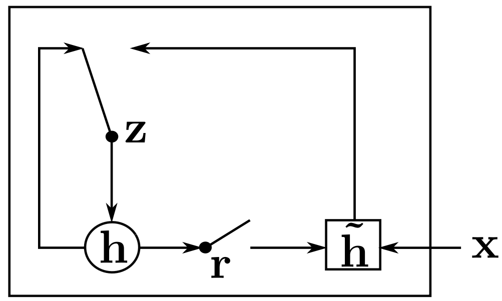

Figure 4.7: Illustration of GRU's structure. Where $\mathbf{x}$ is the input vector, $\mathbf{h}$ is hidden state, $\tilde{\mathbf{h}}$ is the new hidden state, z and r are update and reset gates.

farther the previous word is, the smaller the weight is. 2. If there is an error which may be caused by some previous words, we should ignore the word(s), and only consider the current word.

Long short term memory LSTM is very prevalent in all kinds of natural language processing models. Essentially, it does not have structural difference from RNN, but it uses different kinds of formula to calculate hidden layer status. We can regard LSTM cell as black box, where current input and previous hidden status are stored. It has been proved that the LSTM inner structure is very effective at solving long term dependencies. The inner structure of LSTM looks very similar to GRU (4.8[11]), while there are several differences:

- GRU have the reset gate to control the quantity of information that flows from previous state to current state, but LSTM does not have this gate.
- The approach to generate new state is different. LSTM have two different gates: forget gate and input gate, but GRU has only update gate.
- LSTM has an output gate to adjust the output, but GRU has no such gates.

4.1.5 Softmax function

In practice, we often use Softmax function to output the probability distribution given the input vector. That is, given the input vector that represents the probability distribution,

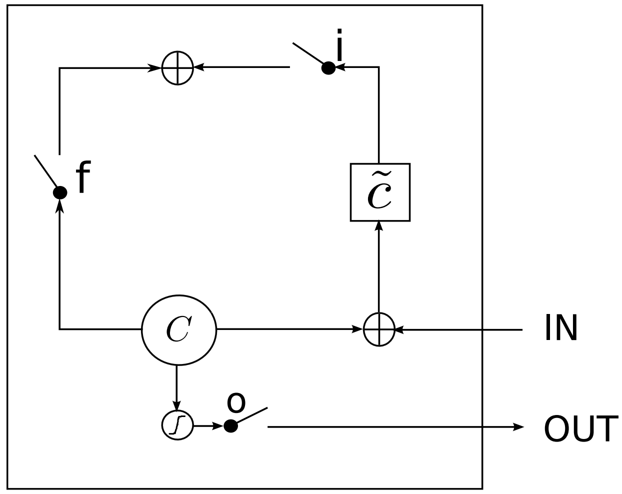

Figure 4.8: Illustration of LSTM's structure. Where i is the input gate, f is the forget gate, o is the output gate, C is the hidden layer.

Softmax can output the vector which contains real values range from zero to one. Also, these values add up to one. The Softmax function is written as follows:

$$P(Y = a|\mathbf{x}) = \frac{e^{\mathbf{x^t w_a} + b_a}}{\sum_{i=0}^{n} e^{\mathbf{x^t w_i} + b_i}} \tag{4.12}$$

4.2 Training Framework Overview

Our training framework is divided into five steps:

1. Training word vectors using large scale of unsupervised data.
2. Crawling huge amount of unsupervised question-question pairs from indexing service.
3. Calculating the similarity scores of the unsupervised pairs using simple similarity algorithms.
4. Train the RNN without pre-training (i.e. using only supervised data), tune the parameters until we find the best set of parameter.
5. Train the new RNN with unsupervised data, then with supervised data.

The work flow is shown in Figure 4.9.

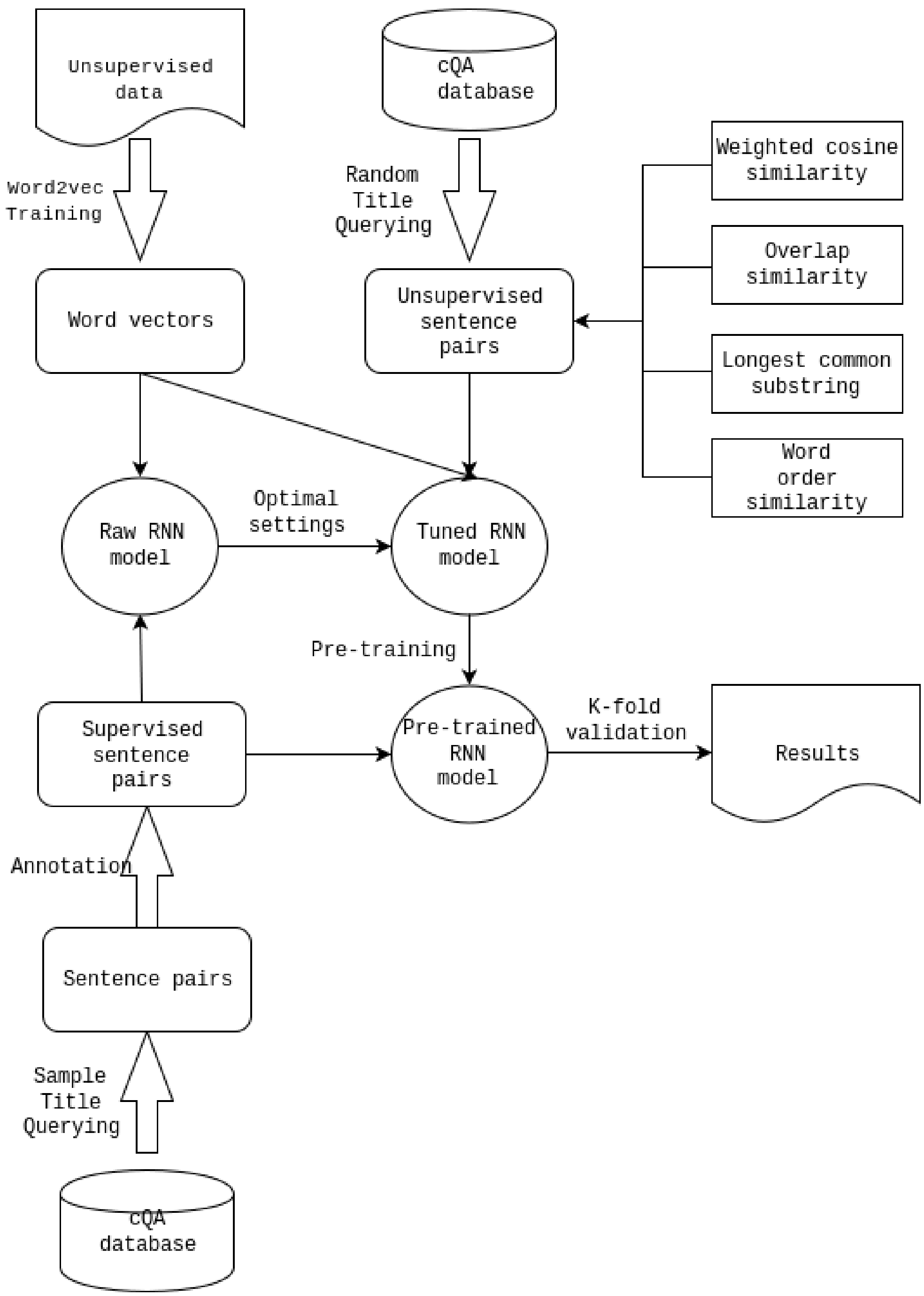

Figure 4.9: Overview of the RNN Encoder-Decoder architecture.

4.3 The RNN Encoder-Decoder Structure

The structure of our RNN model is demonstrated in Figure 4.10. It consists of two RNN layers: the Encoder and the Decoder. The query and the candidate sentence are fed into Encoder and Decoder, respectively. The structure of the RNN Encoder-Decoder model is described as follows:

1. First of all, query and candidate sentences are segmented into a list of Chinese words. Then words are mapped into word vectors using a pre-trained word embedding dictionary. Thus, we get query and candidate matrix.

2. In the encoder layer, the word vectors are fed iteratively into RNN's each time step, then, we take final time step's hidden layer as query's sentence representation.

3. The decoder layer takes in two components. Similar as the encoder layer, it takes in the candidate matrix's word vectors for each time step. In addition, it also takes the query sentence's representation as one of the input (we will describe this in detail in formula and picture in later part of the section, how the word vector and query sentence representation combine). We take the final time step's hidden layer as "similarity representation" of the two sentences.

4. As a final step, the similarity representation vector is fed into a Softmax layer, which outputs a probability representation among different similarity levels (similar, dissimilar, relevant).

Furthermore, we illustrate the structure with a more detailed example, shown in Figure 4.11. In this picture, we show the inner details of encoder and decoder, the dotted-line rectangle areas represent the two full-line rectangle areas in Figure 4.10.

Given the query words sequence $\{q_1, q_2, ..., q_T\}$ and the candidate words sequence $\{c_1, c_2, ..., c_{T'}\}$, the words are iteratively fed into each time step of the RNN. The only difference between Encoder and Decoder's memory cells is that the Decoder takes in the sentence's representation as one of the inputs for each time step. As a result, the final output of the Decoder can be considered as the "similarity representation" of the two sentences.

The RNN Encoder-Decoder formulations are derived from [10]'s Gated Recurrent Unit (GRU).

For the Encoder, formulations are computed as follows:

$$\mathbf{r} = \sigma(\mathbf{W_r q_t} + \mathbf{U_r h_{<t-1>}}) \tag{4.13}$$

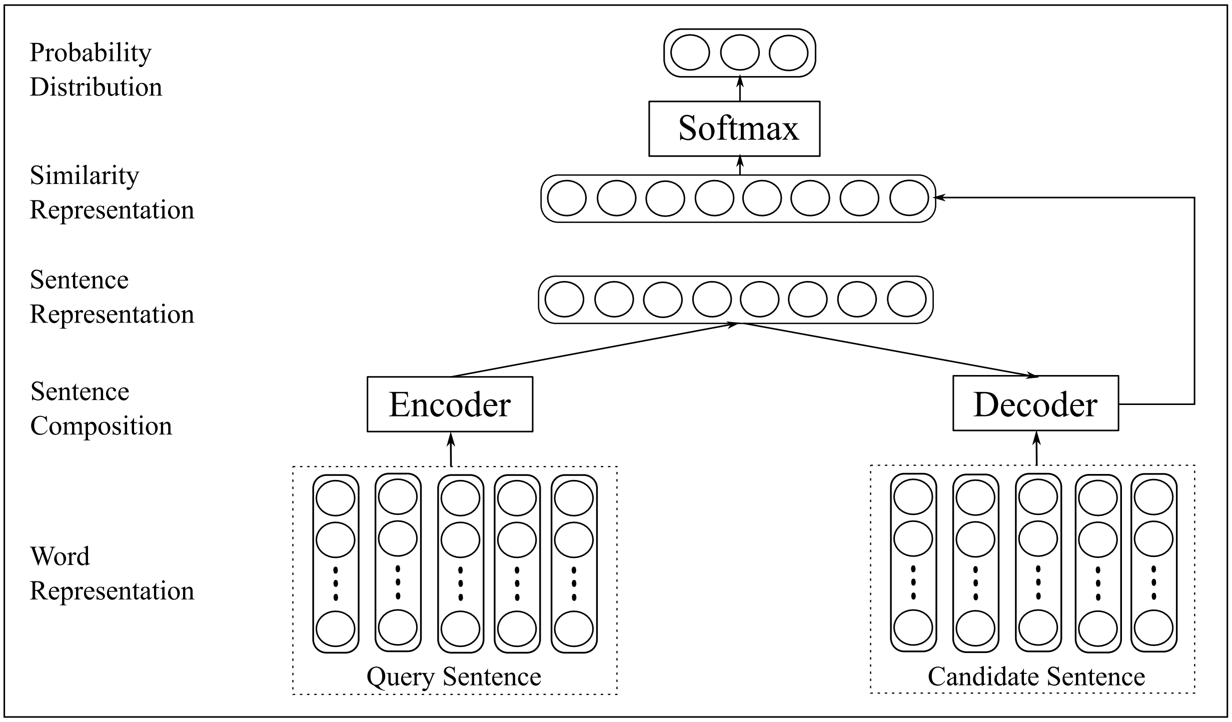

Figure 4.10: Overview of the RNN Encoder-Decoder architecture.

$$\mathbf{z} = \sigma(\mathbf{W_z q_t} + \mathbf{U_z h_{<t-1>}}) \tag{4.14}$$

$$\tilde{\mathbf{h}}_{\mathbf{<t>}} = \phi(\mathbf{W q_t} + \mathbf{U}(\mathbf{r} \odot \mathbf{h_{<t-1>}})) \tag{4.15}$$

$$\mathbf{h_{<t>}} = \mathbf{z} \odot \mathbf{h_{<t-1>}} + (\mathbf{o} - \mathbf{z}) \odot \tilde{\mathbf{h}}_{\mathbf{<t-1>}} \tag{4.16}$$

where $\mathbf{q_t}$ is the t'th word in the query sentence, $\mathbf{r}$ is reset gate, $\mathbf{z}$ is update gate, $\mathbf{o}$ is an all-one vector, $\mathbf{h_{<t>}}$ is hidden layer at time step t, $\odot$ is element-wise product, σ is sigmoid function, ϕ is tanh function. In our experiment, word vector size is 200, hidden layer size is 150, so $\mathbf{r}, \mathbf{z}, \mathbf{h_{<t>}}, \tilde{\mathbf{h}}_{\mathbf{<t>}} \in \mathbb{R}^{150}$, $\mathbf{W_r}, \mathbf{W_z}, \mathbf{W} \in \mathbb{R}^{150\times200}$, $\mathbf{U_r}, \mathbf{U_z}, \mathbf{U} \in \mathbb{R}^{150\times150}$, $\mathbf{q_t} \in \mathbb{R}^{200}$.

For the Decoder, we also take the last hidden layer of the Encoder (denoted as M) into consideration:

$$\mathbf{r}' = \sigma(\mathbf{W'_r c_t} + \mathbf{U'_r h'_{<t-1>}} + \mathbf{C_r M}) \tag{4.17}$$

$$\mathbf{z}' = \sigma(\mathbf{W'_z c_t} + \mathbf{U'_z h'_{<t-1>}} + \mathbf{C_z M}) \tag{4.18}$$

$$\tilde{\mathbf{h}}'_{\mathbf{<t>}} = \phi(\mathbf{W' c_t} + \mathbf{U'}(\mathbf{r'} \odot \mathbf{h'_{<t-1>}}) + \mathbf{CM}) \tag{4.19}$$

$$\mathbf{h'_{<t>}} = \mathbf{z'} \odot \mathbf{h'_{<t-1>}} + (\mathbf{o} - \mathbf{z'}) \odot \tilde{\mathbf{h}}'_{\mathbf{<t-1>}} \tag{4.20}$$

where $\mathbf{c_t}$ is the t'th word in the candidate sentence, $\mathbf{M}$ is the first sentence's representation. $\mathbf{C_r}, \mathbf{C_z}$ are parameters. $\mathbf{c_t} \in \mathbb{R}^{200}$, $\mathbf{C_r}, \mathbf{C_z} \in \mathbb{R}^{150\times150}$.

Finally, the softmax layer at the output of the Decoder generates a probability distribution over different levels of similarities. Hence, given the query Q and the question candidate C, the probability that they are similar, relevant, and dissimilar (i.e., the levels) is:

$$P_{\text{level}}(Q, C) = \{prob_{\text{level}_1}, prob_{\text{level}_2}, ..., prob_{\text{level}_n}\}$$

where $P_{\text{level}}(Q, C)$ represents the probability distribution of query and candidate sentences over different levels, $prob_{\text{level}_i}$ is the probability of similarity level i, $\sum prob_{\text{level}_i} = 1$.

Our objective is to minimise the categorical cross-entropy between the label A and the predicted probability P:

$$\max\{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} -A_{ij}\log P^{i}_{level_j}(Q, C)\},$$

where N is the number of samples in our dataset, L is the number of similarity levels, A_i is the annotated probability distribution for the ith sample, $P^i(Q, C)$ is the predicted one. We optimise the parameters using the stochastic gradient descent algorithm. Once the RNN Encoder-Decode is trained, it can be used in both classification and ranking. Since it outputs a probability distribution over different classes: The sentence pair could be labelled with the class that has the highest probability; or given a list of candidates, we can rank them according to their probability scores.

4.4 Pre-training Heuristics

We have introduced four character-level and word-level similarities as heuristics, to pre-train our RNN model. All of these similarities output a normalised score ranging from 0.0 to 1.0, where higher scores denote more similar sentences.

- **Weighted Cosine Similarity**: Sentences are mapped into a vector space using the bag-of-words model (with word segmentation). Typically, in Chinese there are fewer auxiliary words or particles in longer words, we apply the length as a weight to the importance:

$$S_{\cos} = \frac{\sum_{w\in Q\cap C}\text{len}(w)^2}{\sum_{w\in Q}\text{len}(w)^2\cdot\sum_{w\in C}\text{len}(w)^2}.$$

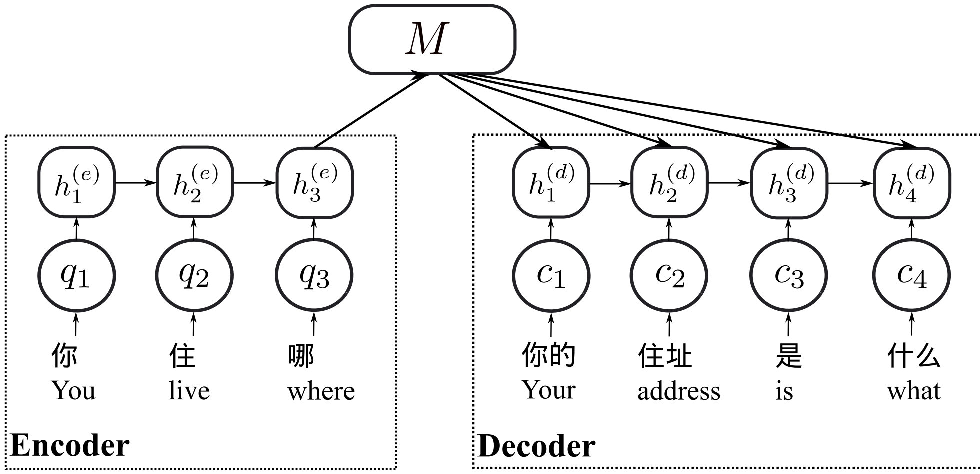

Figure 4.11: The structure of RNN Encoder-Decoder. $\{\mathbf{q}_i\}$ and $\{\mathbf{c}_i\}$ are the words sequences of Q and C, respectively. M is the output of the encoder; it is fed into each hidden layer of the decoder $h_{<t>}$.

- **Overlap Similarity**: The overlap similarity [38] is defined as the length of the common words between a Q-C pair, normalised by their total length:

$$S_{\text{olap}} = \frac{\sum_{w \in Q \cap C} \text{len}(w)}{\text{len}(Q) + \text{len}(C)}.$$

- **Longest Common Substring**: The longest common substring similarity is the length of the longest common substring between two sentences. For example, $S_{\text{lcs}}(\text{“ABCDE”}, \text{“AXYDE”}) = 2$ (the longest common substring is “DE”). The algorithm details are shown in Algorithm 1.

- **Word Order Similarity**: The word order similarity [34] considers the index of each word in the sentence. Using a bag-of-words model, the value of each dimension of the vector v is the index of that word, then:

$$S_{w\text{-order}} = 1 - ||v_1 - v_2|| / ||v_1 + v_2||.$$

With these similarities, we are able to label many query-question candidate pairs automatically. We extract queries and question candidates following the same procedure of Section 3, but instead of only 400 different questions, we select 9,693 totally different

Algorithm 1: Longest Common String Length

```
input  : Two strings, S_1 and S_2
output: Longest Common String Length of input strings, LCS
l_1 ← length(S_1) ;
l_2 ← length(S_2) ;
LCS ← 0 ;
M ← zeros(l_1+1, l_2+1) // M is a matrix of int
foreach char c_1 at position i of S_1 do
    foreach char c_2 at position j of S_2 do
        if c_1 = c_2 then
            M_{i+1,j+1} ← M_{i,j} +1
        else
            M_{i+1,j+1} ← 0
        LCS ← max(M_{i+1,j+1} , LCS)
```

questions for pre-training, and retrieve a total of 1,557,985 query-question pairs. We then assign a similar score to each pair, by linear combining the four similarities:

$$Score = \alpha S_{cos} + \beta S_{olap} + \gamma S_{w-order} + \phi S_{lcs}$$

where $\alpha + \beta + \gamma + \phi = 1$.

This $Score$ is given to each pair as the probability of the label *similar*, while $1 - Score$ for the label *dissimilar*. All the pairs are fed into the RNN Encoder-Decoder along with the pre-trained word vectors as pre-training. In our practice, we use Theano [55] as the framework, with a learning rate = 0.001 without regularisation terms. We only train the pre-training dataset for one epoch. On our servers, the training process costs six to twelve hours (varies with hidden layer dimensions).

4.5 Word Vector Training

Brief introduction of Word2Vec tool Word2Vec toolkit is developed by Mikolov et al. [39], the project link is: https://code.google.com/archive/p/word2vec/. The word vectors share two very interesting properties: 1. Words with similar meanings tend to have small cosine distance. An example is shown in Figure 4.1. 2. The trained word vectors

Table 4.1: Cosine similarity ranking given keyword “Paris”. The word candidates are collected from dictionary. The top 10 most similar words are selected from dictionary, and ranked by their cosine similarity with word “Paris”.

Word	Cosine distance
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.562391
catalonia	0.534176

have many interesting linguistic rules, for example, **vector(‘Paris’) - vector(‘France’) + vector(‘Italy’)** results in a vector, which is very close to **vector(‘Rome’)**.

The word vectors are used in the RNN model and one of the baseline evaluation methods. They are also trained from our cQA corpus (segmented by jieba[1]), using the Word2Vec [39] framework. The training takes 16 hours, results in $4,128,853$ different word vectors with 200 dimensions.

[1] *jieba* is a Python framework for Chinese word segmentation: https://github.com/fxsjy/jieba

Chapter 5

Experiments

In this section, we show the performance of our model by two evaluations: One is sentence similarity classification, and the other one is question candidates ranking.

5.1 Evaluation Metrics

In information retrieval, machine learning, and natural language processing fields, evaluation is an essential job, which tells us the effectiveness of an approach.

5.1.1 Accuracy, Precision, Recall, and F1 Measure

In this section, we briefly discuss the meaning of accuracy, precision, recall, and F1 measure. For example, we have a collection of documents, where P documents are wanted ones, N documents are unwanted ones. Our task is to judge which documents are relevant (wanted) ones, and which are not (negative). If we design an approach to retrieve the documents, we may select the documents that are not relevant, as shown in Figure 5.1.

Definition 5.1.1. True positive In the collection of selected documents, the number of relevant documents are true positives, denoted as TP.

Definition 5.1.2. False positive In the collection of selected documents, the number of irrelevant documents are false positives, denoted as FP.

Definition 5.1.3. False positive In the collection of not selected documents, the number of relevant documents are false negatives, denoted as FN.

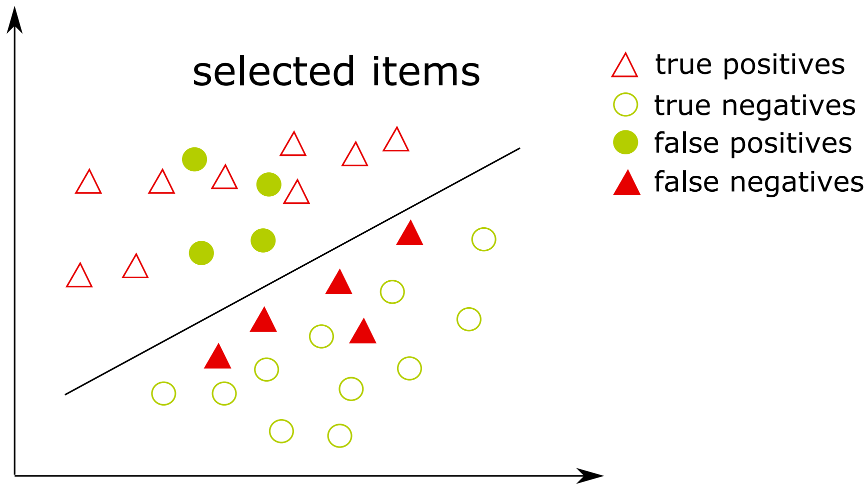

Figure 5.1: Illustration of true positive, true negative, false positive and false negative

Definition 5.1.4. True negative In the collection of not selected documents, the number of irrelevant documents are true negatives, denoted as TN.

Definition 5.1.5. Accuracy The ratio of correctly predicted documents. Calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Definition 5.1.6. Precision In the collection of selected documents, the ratio of correctly predicted documents. Calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

Definition 5.1.7. Recall In the collection of wanted documents, the ratio of retrieved documents. Calculated as follows:

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

Definition 5.1.8. F1 measure Harmonic mean of precision and recall. Can be seen as a balance between recall and precision, as neither of them can represent the overall efficiency of a model. F1 measure is calculated as follows:

$$F1 = \frac{2TP}{2TP + FN + FP} \tag{5.4}$$

5.1.2 Mean Average Precision

In our case, purely calculating the accuracy and F1 measure is not enough, as we will score each sentence pair. In Figure 5.1, sometimes we will give each document a score indicating the confidence of the predicted label. Thus, given relevant document A and irrelevant document B, if we selected both of them, we have the same accuracy; but if we rank them as AB, it will be different from BA. In order to make this difference, we resort to mean average precision (MAP), which is calculated as follows:

$$AP = \left(\sum_{i=1}^{N} P(C_i) \right) / N', MAP = \frac{\sum_{i=1}^{N} AveP(C_i)}{N} \tag{5.5}$$

where $P(C_i)$ is the precision at position i, and N' is the number of similar candidates, N is the number of all candidates.

5.2 Sentence Similarity Classification

In this experiment, we use the model to predict the sentence pair similarity as a binary classification problem: We only use the *similar* and *dissimilar* pairs in our dataset, which are the most significant classes. In this way we can also compare the accuracy, precision, recall, and F_1 score of the models.

5.2.1 RNN training

Parameter Tuning

For our model, there are two parameters we need to tune: learning rate, hidden layer.

Learning rate Let's recall Function 4.10:

$$\mathbf{W} < -\mathbf{W} - \eta \frac{\partial L(\theta)}{\partial \mathbf{W}} \tag{5.6}$$

where η is learning rate. Figure 5.2 shows an example of the loss function in three dimensional space. The learning phase is like climbing down to the bottom of the valley, and the learning rate is like step size. When you come close to the bottom and your step size

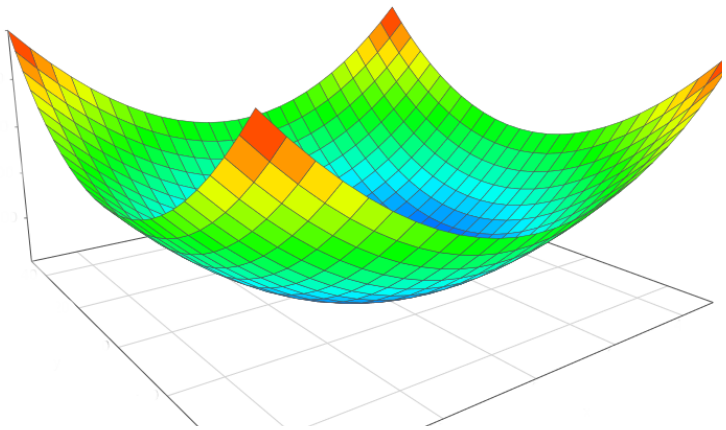

Figure 5.2: Example of loss function in 3D space. The bottom of the bowl is the point we want to reach in learning phase.

is too large, you may linger around the bottom and will never reach it. Figure 5.3[1] gives an example of how the model converges using different learning rates.

Hidden layer size Apart from learning rate, the hidden layer size is also important for training. The more neurons the hidden layer has, the stronger the neural networks representation capability is. But if we use too many neurons, over-fitting problem will emerge.

In order to find the optimal parameter, we use vanilla RNN to perform 5 fold cross validation on our data set, and get the optimal parameter accordingly. we do 2 steps:

- **Adjust learning rate**. We tested 7 different learning rates: 1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001. Hidden layer size is set to 200. The results are shown in Figure 5.1.
- **Adjust hidden layer size**. We test 8 different hidden layer sizes: 1000, 700, 500, 300, 200, 100, 50. The learning rate is set to 0.001. The results are shown in Figure 5.2.
- **Grid search**. We picked up 0.0005, 0.001 from learning rate set and 100, 150, 200 from hidden layer size set. The results are shown in Figure 5.3.

[1]Image source: http://neuralnetworksanddeeplearning.com/chap3.html

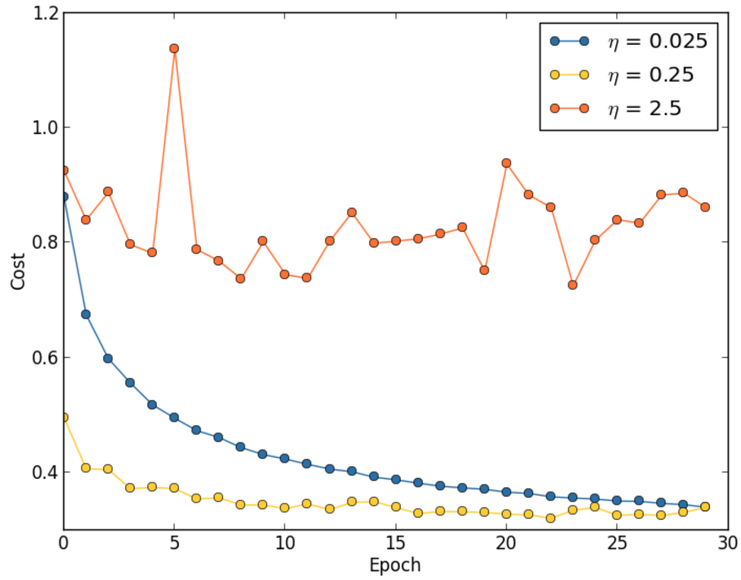

Figure 5.3: Example of loss convergence with different learning rates. If the learning rate is too large, the loss will fluctuate. When the learning rate is appropriate, the loss will shrink smoothly. (Image source: `http://neuralnetworksanddeeplearning.com/chap3.html`)

Pre-training

As mentioned before, we select 9,693 totally different questions for pre-training, and retrieved a total of 1,557,985 query-question pairs from our indexing service. We then assign a similar score to each pair, by linear combining the four similarities:

$$Score = \alpha S_{cos} + \beta S_{olap} + \gamma S_{w-order} + \phi S_{lcs}$$

where $\alpha + \beta + \gamma + \phi = 1$.

Table 5.4 shows scoring examples. We can see that the unsupervised pairs are not 100% correct.

Cross-validation

After the pre-training, we perform a five-fold cross-validation on our annotated data set, with the learning rate of 0.001, and the hidden layer size of 150. We do not deliberately

Table 5.1: Learning rate tuning.

Learning rate	**Acc.**	**Prec.**	**Rec.**	F_1
1	0.6146	–	–	–
0.5	0.6252	0.5591	0.1300	0.2109
0.1	0.5989	0.4753	0.3937	0.4307
0.05	0.6358	0.5391	0.3781	0.4445
0.01	0.7200	0.6319	0.6545	0.6430
0.005	0.7726	0.6935	0.7347	0.7135
0.001	0.7978	0.7328	0.7481	0.7404
0.0005	0.7964	0.7315	0.7451	0.7383
0.0001	0.7569	0.7008	0.6441	0.6713

Table 5.2: Hidden layer size tuning.

Hidden layer size	**Acc.**	**Prec.**	**Rec.**	F_1
50	0.8007	0.7559	0.7132	0.7339
100	0.8033	0.7524	0.7295	0.7408
150	0.8081	0.7653	0.7243	0.7442
200	0.7978	0.7328	0.7481	0.7404
500	0.7904	0.7234	0.7384	0.7308

tune our parameter. For each fold we train 50 epochs.

5.3 Question Candidates Ranking

In this experiment, we test different models' abilities to rank the matching degree given the candidate list. We evaluate the retrieved candidates with mean average precision (MAP) within the ranking context – the mean value among the precisions of the candidate similarities. We select 26 topics out of 128 which has the percentage of similar candidate ranging from 40% to 80%, and compared our model with Sim-Avg and logistic regression (LR-Sim and LR-Avg).

This capability makes it possible to fit in more applications which require probability distributions or continuous scores.

Table 5.3: Grid search tuning.

Hidden layer size	**Learning rate**	**Acc.**	**Prec.**	**Rec.**	F_1
100	0.0005	0.7849	0.7019	0.7682	0.7335
100	0.001	0.8033	0.7424	0.7496	0.7460
150	0.0005	0.8038	0.7284	0.7830	0.7547
150	0.001	0.8081	0.7653	0.7243	0.7407
200	0.0005	0.7964	0.7315	0.7451	0.7383
200	0.001	0.7978	0.7328	0.7481	0.7404

Table 5.4: Examples of unsupervised query-question pairs.

Query	**Question**	**Score**
什么PDF阅读器好用 Which pdf reader is convenient to use	PDF阅读器有什么用? How to use pdf reader?	0.9105
显示器进水了怎么办啊 The display is watered , what can I do	三星液晶显示器进水怎么办 Sumsung LED display is watered, what can I do	0.7174
电脑运行期间关掉显示器有好处吗 Is it better to turn off display while my computer is running	为什么电脑显示器不能关掉? Why I cannot turn off my computer display?	0.5566
怎样健身才能让胳膊变得粗壮 How to work out to make arms stronger	怎样使手腕变得粗壮 How to make wrist stronger	0.3866

5.4 Baselines

We use the following four methods as our baselines:

- Average of the heuristic sentence similarities (Sim-Avg): The average of the four similarities for pre-training is taken as the similarity score for each Q-C pair. Only those with scores higher than a threshold θ are considered similar. The procedure is depicted in Figure 5.4.

- Bag-of-features model: We compute the four similarities of each Q-C pair, as features to form a vector. Classified with support vector machine (SVM-Sim) or logistic regression (LR-Sim).

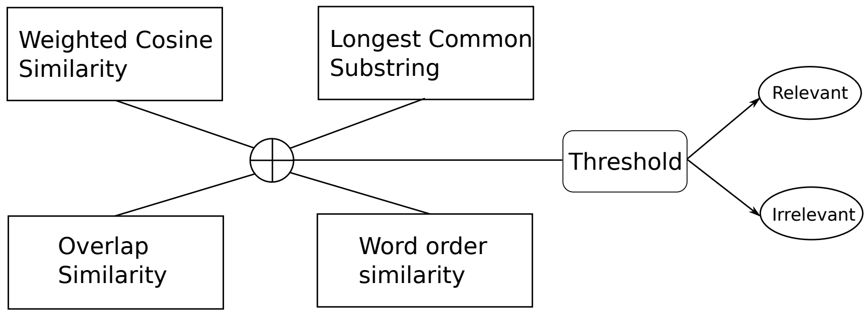

Figure 5.4: Average of the heuristic sentence similarities

- Sentence representation: Instead of using the four similarities to represent a sentence, we use the average of all the word vectors of a sentence as its representation, then use SVM (SVM-Vec) and logistic regression (LR-Vec) for classification. See Figure 5.5 for details.

- Similarity matrix (SimMat): We adopt [49]'s similarity matrix method. Instead of using the phrase vector from their RAE, we directly use our pre-trained word vectors. The similarity matrix is shown in Figure 5.6.

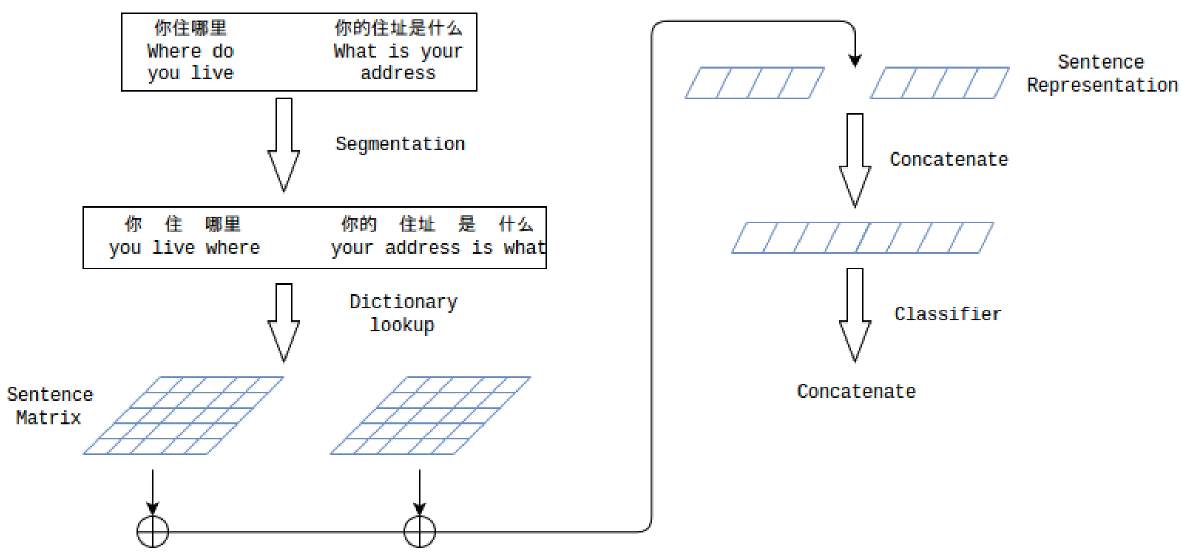

Figure 5.5: Sentence Representation baseline pipline.

We used the Scikit Learn [45] toolkit for SVM (with the RBF kernel) and LR.

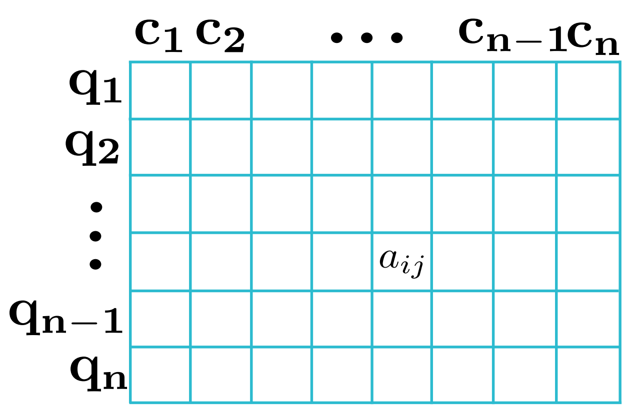

Figure 5.6: Similarity matrix example. $\mathbf{c_i}$ and $\mathbf{q_i}$ are word vectors of candidate and query sentence, respectively. a_{ij} is the matrix's value, which is the cosine similarity of $\mathbf{q_i}$ and $\mathbf{c_j}$

Chapter 6

Results and Discussion

The results of classification and ranking MAP are shown in Table 6.1.

Obviously, the straight-forward Sim-Avg method can reach very high precision with a higher threshold, but with the cost of an unacceptable recall rate. When measuring with the F_1 score, its capability can reach an accuracy of about 0.7.

With word vectors, the semantic information has been fully utilised, hence the results with -Vec are better than -Sim. Between them, the SVM model has slightly higher precisions than the logistic regression model. These results show the capability of traditional

Table 6.1: Comparison of the models on classification and ranking.

Model	**MAP**	**Acc.**	**Prec.**	**Rec.**	F_1
Sim-Avg, $\theta = 0.9$	0.8701	0.6412	**0.9793**	0.0705	0.1316
Sim-Avg, $\theta = 0.8$	0.8701	0.6965	0.9359	0.2280	0.3667
Sim-Avg, $\theta = 0.6$	0.8701	0.7162	0.6256	0.6567	0.6408
LR-Sim	0.8778	0.7469	0.7015	0.5973	0.6452
LR-Vec	0.7670	0.7998	0.7549	0.7117	0.7326
SVM-Sim	–	0.7472	0.7609	0.5014	0.6045
SVM-Vec	–	0.8116	0.8127	0.6641	0.7309
SimMat	–	0.6933	0.6395	0.4680	0.5405
RNN (150)	–	0.8081	0.7653	0.7243	0.7442
RNN, Pre-trained (150)	**0.8814**	0.8393	0.7789	**0.8142**	**0.7962**
GRU (150)	–	0.8273	0.7778	0.7726	0.7752
GRU, Pre-trained (150)	0.8646	**0.8434**	0.8158	0.7667	0.7905

Table 6.2: Examples of the similarity prediction results. RNN is the pre-trained model in this table.

Query	Question	Label	Sim-Avg	LR-Vec	RNN
狗不能吃什么 What can't dogs eat	狗产后吃什么好 What can dogs eat after they give births	0	0.7503	0.2564	0.0354
女属虎的和男属兔配吗 Does a woman born in the year of Tiger match a man born in the year of Rabbit	属虎男和属兔女配吗 Does a man born in the year of Tiger match a woman born in the year of Rabbit	0	0.8894	0.6976	0.0222
螃蟹多少钱一个 How much is one crab	现在市场上螃蟹的价钱 The current market price of crabs	1	0.2675	0.3320	0.9709
集体户口如何迁回老家 How to move collected registered residence address to my hometown	户籍迁移程序：我的户口都是集体户口，现在想迁移回家，需要哪些手续 The procedure to move residence address: I have collected registered residence, I want to change it to my home town, what is theprocedure	1	0.3530	0.1207	0.0002

models on this task.

We can also see that [49]'s similarity matrix performs badly on our task. We do not use RAE's phrase vector; besides, the sentence lengths are also different. In their experiments with the MSRP corpus, the sentences are all very long (mostly > 20), making their pooling step helpful. But our dataset's sentence length varies from 1 to 76, making fix-sized pooling more difficult. This also reveals that this pooling step is also easily influenced by different data set nature.

Among all algorithms, the RNN model achieves better balance between precision and recall. Also, we can see with pre-training, RNN's performance is dramatically boosted, which reveals that our framework can capture more information at the first pre-training stage, and perform a lot better in the succeeding processes. Moreover, the ranking result (MAP) also shows that the RNN model is better.

We further investigate some examples, shown in Table 6.2. For Sim-Avg, its classification output depends on threshold, but as a value of over 0.7, the threshold must be set high to classify it into 0 (dissimilar). For LR and RNN, they give a positive label (similar) if the output < 0.5. The first example shows RNN can better recognise negations, which is very important in question-answering. The second example shows RNN takes word order information into account, which solve one of the hard-to-solve problems to some extent. The third example shows by using word vectors, we can better capture semantic information, as "cost" and "price" are closely related. This greatly improve the performance of the model in practice.

References

[1] Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. The evaluation of sentence similarity measures. In *Data warehousing and knowledge discovery*, pages 305–316. Springer, 2008.

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

[3] James Allan, Courtney Wade, and Alvaro Bolivar. Retrieval and novelty detection at the sentence level. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 314–321, 2003.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155, 2003.

[5] William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics, 2012.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[7] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "Siamese"

time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(04):669–688, 1993.

[8] Wanxiang Che, Zhenghua Li, and Ting Liu. LTP: A Chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16, 2010.

[9] Wanxiang Che, Valentin I Spitkovsky, and Ting Liu. A comparison of Chinese parsers for Stanford dependencies. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, pages 11–16, 2012.

[10] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[12] Tatiana Almeida Souza Coelho, Pável Pereira Calado, Lamarque Vieira Souza, Berthier Ribeiro-Neto, and Richard Muntz. Image retrieval using multiple evidence ranking. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):408–417, 2004.

[13] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*, 2005.

[14] Zhendong Dong, Qiang Dong, and Changling Hao. HowNet and its computation of meaning. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 53–56, 2010.

[15] Abdessamad Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 16–23, 2003.

[16] Asli Eyecioglu and Bill Keller. ASOBEK: Twitter paraphrase identification with simple overlap features and SVMs. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 64–69, 2015.

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

[18] Rafael Ferreira, Luciano de Souza Cabral, Rafael Dueire Lins, Gabriel Pereira e Silva, Fred Freitas, George DC Cavalcanti, Rinaldo Lima, Steven J Simske, and Luciano Favaro. Assessing sentence scoring techniques for extractive text summarization. *Expert systems with applications*, 40(14):5755–5764, 2013.

[19] Ricardo Ferreira, Rafael Dueire Lins, Fred Freitas, Beatriz Avila, Steven J Simske, and Marcelo Riss. A new sentence similarity method based on a three-layer sentence representation. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 110–117, 2014.

[20] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378–382, 1971.

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

[22] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[23] Martin Griffiths. *International relations theory for the twenty-first century: an introduction*. Routledge, 2007.

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

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

[26] Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 84–90, 2005.

[27] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[28] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

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

[30] Jeongwoo Ko, Luo Si, and Eric Nyberg. A probabilistic framework for answer selection in question answering. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 524–531, 2007.

[31] Zornitsa Kozareva and Andrés Montoyo. *Paraphrase Identification on the Basis of Supervised Machine Learning Techniques*, pages 524–533. Springer Berlin Heidelberg, 2006.

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

[33] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

[34] Thomas K Landauer, Darrell Laham, Bob Rehder, and Missy E Schreiner. How well can passage meaning be derived without using word order? A comparison of latent semantic analysis and humans. In *Proceedings of the 19th annual meeting of the Cognitive Science Society*, pages 412–417, 1997.

[35] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[36] Yuhua Li, David McLean, Zuhair Bandar, James D O'shea, Keeley Crockett, et al. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1138–1150, 2006.

[37] Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, 1996.

[38] Donald Metzler, Yaniv Bernstein, W. Bruce Croft, Alistair Moffat, and Justin Zobel. Similarity measures for tracking information flow. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 517–524, 2005.

[39] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[40] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, 2008.

[41] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

[42] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

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

[44] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, 2002.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[46] Long Qiu, Min-Yen Kan, and Tat-Seng Chua. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26, 2006.

[47] Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 464–471, 2007.

[48] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, 2015.

[49] Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*, pages 801–809, 2011.

[50] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161, 2011.

[51] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[52] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[53] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[54] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, 2015.

[55] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[56] Ngoc Phuoc An Vo, Simone Magnolini, and Octavian Popescu. FBK-HLT: An effective system for paraphrase identification and semantic similarity in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 29–33, 2015.

[57] Ellen M Voorhees and DM Tice. Overview of the TREC-9 question answering track. In *TREC*, 2000.

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

[59] Wei Xu, Chris Callison-Burch, and Bill Dolan. SemEval-2015 task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 1–11, 2015.

[60] Xiaobing Xue, Jiwoon Jeon, and W. Bruce 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*, pages 475–482, 2008.

[61] Jiang Zhao and Man Lan. ECNU: Leveraging word embeddings to boost performance for paraphrase in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 34–39, 2015.