Convolutional Neural Network for Sentence Classification

by

Yahui Chen

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, 2015

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 goal of a Knowledge Base–supported Question Answering (KB-supported QA) system is to answer a query natural language by obtaining the answer from a knowledge database, which stores knowledge in the form of (entity, relation, value) triples. QA systems understand questions by extracting entity and relation pairs. This thesis aims at recognizing the relation candidates inside a question. We define a multi-label classification problem for this challenging task. Based on the word2vec representation of words, we propose two convolutional neural networks (CNNs) to solve the multi-label classification problem, namely Parallel CNN and Deep CNN. The Parallel CNN contains four parallel convolutional layers while Deep CNN contains two serial convolutional layers. The convolutional layers of both the models capture local semantic features. A max over time pooling layer is placed on the top of the last convolutional layer to select global semantic features. Fully connected layers with dropout are used to summarize the features. Our experiments show that these two models outperform the traditional Support Vector Classification (SVC)–based method by a large margin. Furthermore, we observe that Deep CNN has better performance than Parallel CNN, indicating that the deep structure enables much stronger semantic learning capacity than the wide but shallow network.

Acknowledgements

I would like to thank all the people who made this possible. First and foremost, I want to express profound gratitude towards my supervisor Dr. Ming Li for his support. His invaluable detailed advices on project and thesis encourage me a lot. Second, I would like to thank Dr. Pascal Poupart. He provides insight and expertise that assisted my thesis. Third, I would like to thank Dr. Chrysanne Di Marco and Dr. Khuzaima Daudjee to read my thesis and to provide valuable advices. Finally, My sincere thanks go to my father and all my friends for their encouragement and support.

Dedication

This is dedicated to my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

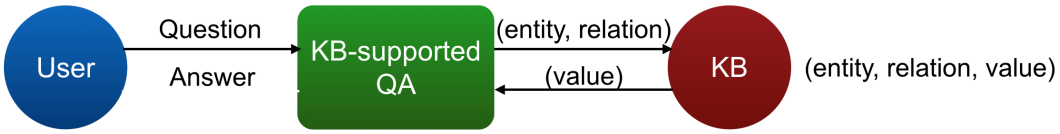

Figure 1.1: KB-supported QA.

Natural language processing (NLP) is the focus of artificial intelligence research and has many applications: machine translation, named entity recognition (NER), question answering (QA), etc. The purpose of a QA system is to automatically answer questions posed by humans in a natural language. Knowledge–Base supported (KB–supported) QA system obtains answers by querying from a structured knowledge database, which stores tuples of (entity, relation, value), and generating answers in natural languages according to the query result, as shown in Figure 1.1. For example, for the question "*Who's the president of the United States?*" the entity is "*USA*", the relation is "*be–president–of*", and the value is "*Barack Obama*". Understanding human questions, especially extracting the entity and relation candidates, is the first and vital step toward implementing the whole system. Many traditional methods depend on keywords or templates matching. But they rely heavily on hand–crafted rules, which cannot be scaled up. To leverage human labour in constructing the keywords or templates, some recent machine learning algorithms have been proposed to automatically learn features and measure semantic distances between queries and known domains.

1.2 Problem Definition

This thesis defines a multi–label classification problem for extracting the relation candidates from a question. We target a widely used question dataset [15], which is crawled from WikiAnswer and consists of a set of questions with over 19K relations. We assume that these open–domain questions have only first–order relations, which we call single–relation questions, for example, "*Who's the president of the United States?*" has a first–order relation, but "*Who's the wife of the United States' president?*" has a second–order relation. Single–relation questions are the most commonly observed ones in QA sites [16]. However, since human expressions or understanding could be ambiguous, each question may have several relation candidates, for example, "*What is the primary duty of judicial branch?*" has relation candidates "*be–primary–responsibility–of*", "*be–primary–role–of*", and "*have–role–of*". Thus we address the problem as recognizing the relations inside a question in a multi–label manner.

1.3 Contributions

We explore various deep learning models to solve the proposed multi–label recognition problem. At the first step, we exploit the widely used word2vec [33] [35] [36] to represent each word as a 300 dimensional vector, and the whole sentence as a matrix by stacking all the word vectors. Word2vec converts the semantic relations between words into the distance of their vectors, for example, word2vec('Paris') - word2vec('France') + word2vec('China') = word2vec('Beijing').

Based on the matrix representation of each sentence, we propose two kinds of convolutional neural networks (CNNs): Parallel CNN and Deep CNN. Convolutional layers of both networks can learn phrases, such as "where do ... live", and "the population of". Parallel CNN is a shallow network but has multiple parallel convolutional layers. Deep CNN, on the contrary, has multiple serial convolutional layers. Our experiments show that both Parallel and Deep CNN outperform the traditional Support Vector Classification (SVC)–based method by a large margin. Furthermore, we observe that Deep CNN has better performance than Parallel CNN, indicating that the deep structure enables much stronger semantic learning capacity than the wide but shallow network.

1.4 Thesis Organization

Section 2 mainly background knowledge of the structures and components of CNNs. Section 3 introduces recent research on CNNs for NLP tasks. Section 4 presents the dataset, tools, and environment used in this work. Section 5 describes our method and shows experimental results. Section 6 concludes this thesis.

Chapter 2

Background

2.1 Deep Neural Network

Deep learning has shown powerful feature learning skills and achieved remarkable performance in computer vision (CV) [8] [43], speech recognition [11] [21], and natural language processing (NLP) [9]. Deep neural network is a kind of deep learning method. The difference between deep neural network (DNN) and shallow artificial neural network (ANN) is that the former contains multiple hidden layers so that it can learn more complex features. It has several variants: convolutional neural network, recurrent neural network, and recursive neural network. DNNs have forward pass and back propagation. The parameters of networks are updated according to learning rate, cost function via stochastic gradient descent during the back propagation. In the following, we briefly introduce the structures of different DNNs applied in NLP tasks.

2.1.1 Convolutional Neural Network

Convolutional neural networks (CNNs) learn local features and assume that these features are not restricted by their absolute positions. In the field of NLP, they are applied in Part–Of–Speech Tagging (POS), Named Entity Recognition (NER) [9], etc.

Figure 2.1 shows a two–layer CNN. For the green node $h_0 = f(W \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} + b) =$

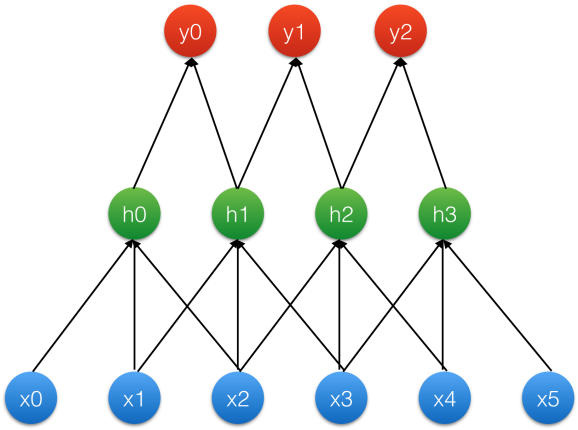

Figure 2.1: Convolutional Neural Network. Neurons in CNN are locally connected with neurons in previous layer. Weights of the same filter are shared across the same layer.

$f(w_0x_0 + w_1x_1 + w_2x_2 + b)$ and for the green node $h_1 = f(W \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b) = f(w_1x_1 + w_2x_2 + w_3x_3 + b)$. W is shared by the same filter in the same layer.

2.1.2 Recurrent Neural Network

The limitation of convolutional neural network is that they take fixed–sized inputs and produce fixed–sized outputs. Recurrent neural networks (RNNs) can operate over sequential input and predict sequential output. They can do one–to–one, one–to–many, many–to–one, many–to–many jobs. They can be used in machine translation [34] and other NLP tasks.

Figure 2.2 shows a simple recurrent neural network with three layers: input layer x, hidden layer h and output layer y. Horizontal arrows stand for time changing. Input sequence: $x_1, x_2, \ldots, x_T$. For each time step t, $h_t = f(W^{hh}h_{t-1} + W^{hx}x_t)$ and $y_t = g(W^{hy}h_t)$, where W^{hh}, W^{hx} and W^{hy} are parameters shared across time sequence. Hidden layer's states are influenced by all the previous inputs. It also has a bidirectional structure to incorporate both forward and backward inputs. RNN has a *vanishing or exploding gradient* problem, as shown in Figure 2.3, while initializing weight matrix to identity matrix [46] and using ReLU activation function [29] address this problem to a certain degree.

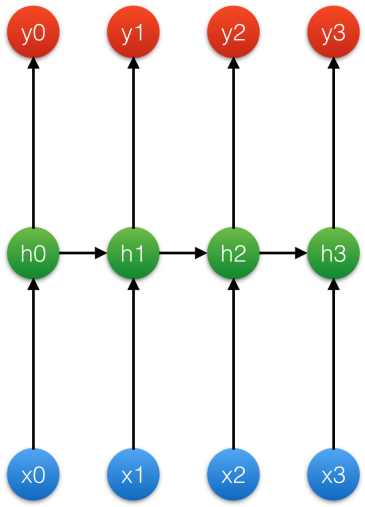

Figure 2.2: Recurrent Neural Network. RNN takes input sequence. Weights of hidden units are updated according to current input and previous weights of hidden units at each time step. Outputs of RNN are calculated according to current hidden units state.

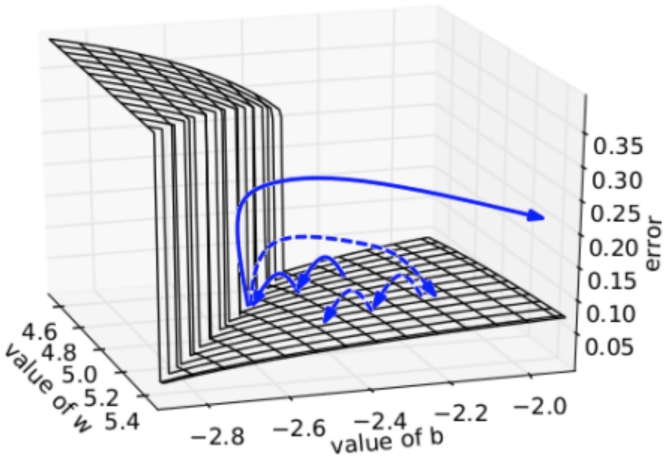

Figure 2.3: Error Surface of a Single Hidden Unit RNN [41].

2.1.3 Recursive Neural Network

Recursive neural networks (RNNs) have been applied to multiple NLP tasks, such as sentence classification [47].

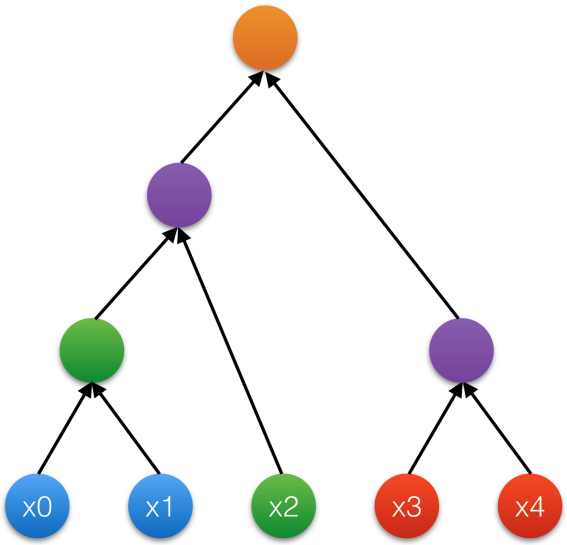

Figure 2.4: Recursive Neural Network.

Figure 2.4 shows a simple recursive neural network. Each node takes two children as inputs. $h = f(Wx + b)$ and $y = U^T h$. For example, for the green node, the parent of x_0 and x_1, $h_{01} = f(W \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + b)$, and for the purple node, the parent of h_{01} and x_2, $h_{012} = f(W \begin{pmatrix} h_{01} \\ x_2 \end{pmatrix} + b)$. In CNNs, weights are shared within the same filter, while in RNNs, weights are shared across different layers. Recursive neural networks have different composition functions: *Matrix–Vector RNNs*, *Recursive Neural Tensor Networks*, *Tree LSTM*, etc.

Recursive neural networks require parsers to get the semantic structures of the sentences. Recurrent neural networks are good at dealing with learning time–sequential features. Convolutional neural networks have good performances in classification and are used as models for the task described in this thesis.

2.2 Motivation and History

Convolutional Neural Networks are inspired by a cat's visual cortex. Visual cortex contains a complex arrangement of cells. These cells are responsible for detecting small sub–fields of the visual field, called receptive fields. The sub–fields are tiled to cover the whole visual field. These cells act as local filters over the input space and are well–suited to exploit the strong spatially local correlation present in natural images.

Neocognitron was introduced by Fukushima in 1980 [18] and improved in 1998 by LeCun, Bottou, Bengio, and Haffner [30]. They proposed the famous LeNet–5 — a convolutional neural network. Then it was generalized by Behnke [6], and pre-digested by Simard and his collaborators in 2003 [45]. Convolutional neural networks perform well on problems such as recognizing handwritten numbers, but the computational power at that time limited their ability to solve more complex problems until the rise of efficient GPU computing.

2.3 Basic Assumption

The convolutional layer is based on the assumption that features are learned regardless of their absolute positions. This is reasonable in many cases, for example in image learning, if detecting a horizontal edge is important at some location in the image, it should also be useful at other locations.

Convolutional layers focus on learning local features. In natural language processing, if in the sentence "give me an example of thank you letter" *example of* has been learned as a feature, then it should also be recognized in sentence "what is an example of scientific hypothesis". But *example of* may not have any relation with *thank you letter* or *scientific hypothesis*. For example in audio recognition, features of time spans of audio clips are learned instead of that of the whole input audio.

2.4 Review of Discrete Convolution Definition

Recall the definition of convolution for a 1D signal [12]. The *discrete convolution* of f and g is given by:

$$o[n] = f[n] * g[n] = \sum_{u=-\infty}^{\infty} f[u]g[n-u] = \sum_{u=-\infty}^{\infty} f[n-u]g[u]. \tag{2.1}$$

This can be extended to 2D as follows:

$$o[m,n] = f[m,n] * g[m,n] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u,v]g[m-u,n-v] \tag{2.2}$$

2.5 Volumes of Neurons

The neurons in convolutional neural networks are arranged in three dimensions: *depth*, *width*, and *height*.

If the network is for image classification, images are in size of $3 \times 32 \times 32$ (three colour channels, 32 wide, 32 high). The size of input layer is $3 \times 32 \times 32$. The size of hidden layer is $12 \times 16 \times 16$ in which 12 is the number of feature maps and 16 is the width and height of a feature map. The size of output layer is $10 \times 1 \times 1$ where 10 is the number of classes to be learned.

If the network is for sentence classification, a sentence has 34 words and each word is represented by 300 dimensional vector. The size of input layer is $1 \times 34 \times 300$. The size of hidden layer might be $256 \times 17 \times 1$ where 256 is the number of feature maps. The output layer has $46 \times 1 \times 1$ dimensions for 46 class classification.

2.6 Architecture

Three types of layers build up a convolutional neural network: *Convolutional Layer*, *Pooling Layer*, and *Fully Connected Layer*.

2.6.1 Convolutional Layer

Convolutional layers have attributes shown below:

- Sparse connectivity: makes each neuron focuses on local features.
- Weight shared: increases learning efficiency by reducing the number of free parameters being learnt.

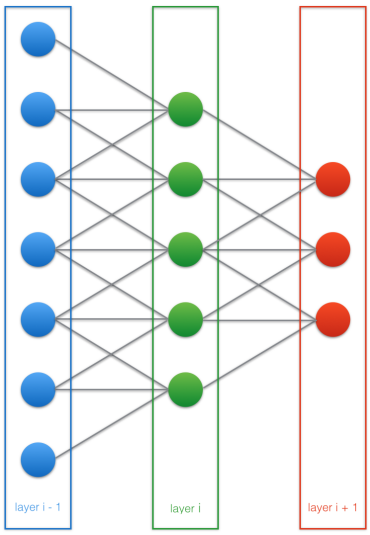

Figure 2.5: Sparse Connectivity.

In Figure 2.5, in layer i, a neuron is connected with contiguous neurons which are the subset of the neurons in layer $i-1$. Connection between two connected neuron represents convolution of *filter* (*kernel*) and input. Each filter has smaller size along width and height but has the same depth as the input.

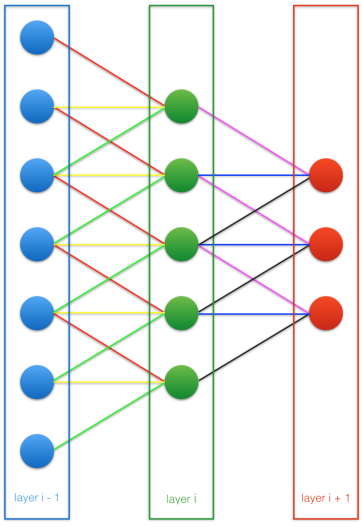

Figure 2.6: Shared Weights. Connections with same colour share weights.

Each filter learns a *feature map*. In the forward pass, when applying convolutional computing, each filter is slid across the width and height and the *dot product* is computed between the entries of filter and the corresponding inputs. In Figure 2.6, weights of the same colour are shared. In the back propagation process, calculating the gradient of a shared weight is to sum up the gradients of the parameters being shared.

A convolutional layer always has a set of filters. Feature maps learned by different filters are stacked along depth dimension.

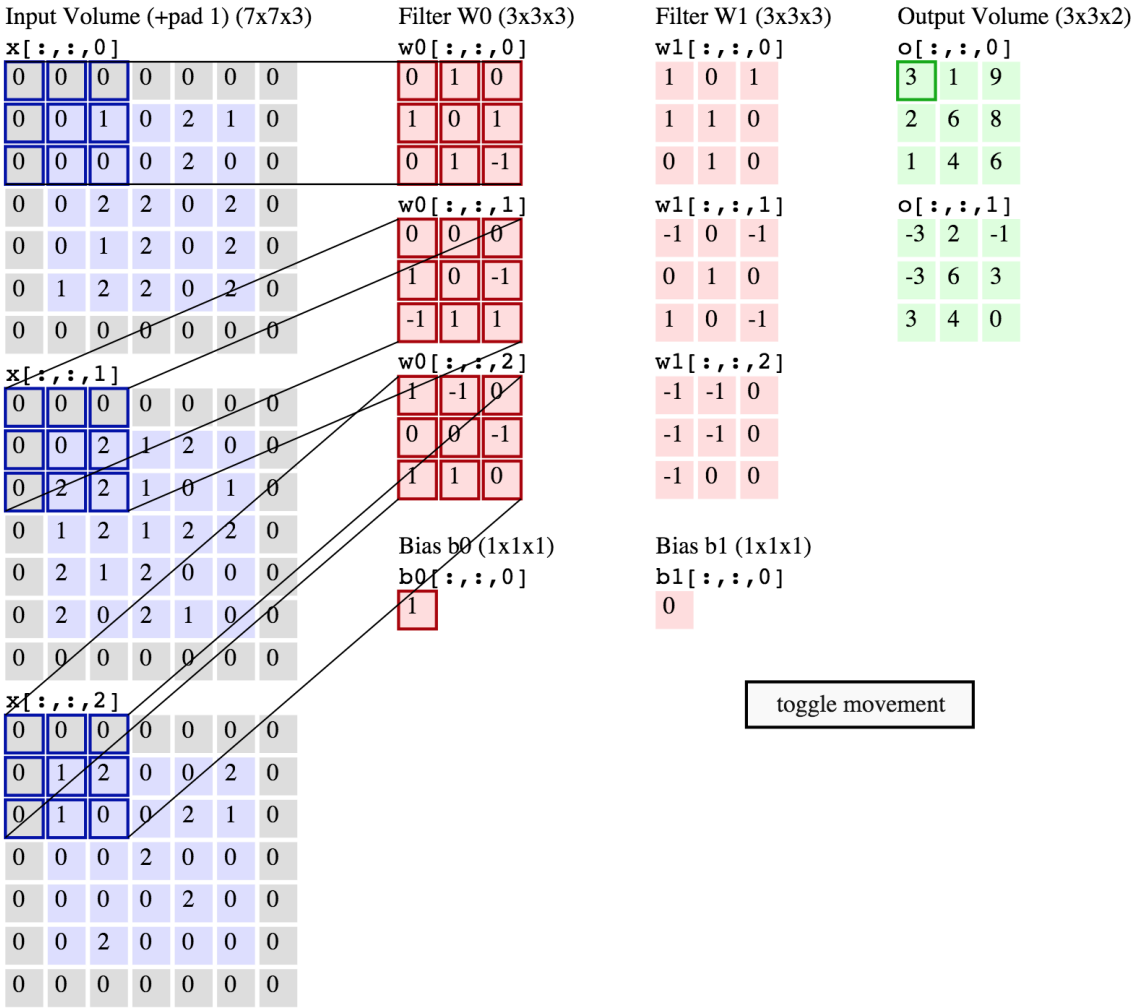

Figure 2.7: Convolutional Layer [26].

In Figure 2.7, input has size of $d_i \times w_i \times h_i = 3 \times 5 \times 5$. Input is padded with 0s of width $w_p = 1$ and height $h_p = 1$. The depth of output (or the number of feature maps to be learned) d_o is 2. The size of filter is $d_o \times d_f \times w_f \times h_f = 2 \times 3 \times 3 \times 3$. The strides when filter is slid along width w_s and height h_s are both 1. The feature map h_k, which is learned by the filter k, is determined by the weights W_k and bias b_k as follows:

$$h_k = f(W_k * x + b_k) \tag{2.3}$$

, where f is an activation function which will be introduced in Subsection 2.6.5. The volume of output should be:

$$d_o \times (\frac{w_i - w_f + 2w_p}{w_s} + 1) \times (\frac{h_i - h_f + 2h_p}{h_s} + 1) \tag{2.4}$$

The number of parameters to be learned should be:

$$(w_f \cdot h_f \cdot d_f + 1) \cdot o_d \tag{2.5}$$

i.e., The output has size of $2 \times 3 \times 3$. The number of parameters to be learned is $(3 \times 3 \times 3 + 1) \times 2 = 56$.

In regular neural networks, every neuron is fully connected with all neurons in the previous layer. If the sizes of input and output are the same, the number of parameters of a fully connected neural network becomes $(w_i \cdot h_i \cdot d_i + 1) \cdot w_o \cdot h_o \cdot d_o = (7 \times 7 \times 3 + 1) \times 3 \times 3 \times 2 = 2664$. The number of parameters in a convolutional neural network is positively correlated with the size of the filter, while that in a regular neural network is positively correlated with the size of the input and the output. But the size of filter is much smaller than that of input and output. The number of parameters in regular neural network is very large as every layer is fully connected with neighbour layers, which sometimes makes learning process overfitting.

2.6.2 Pooling Layer

To describe a large matrix, one natural approach to down sample is to aggregate statistics. Common methods include computing the *mean value*, *max value* and *L2–norm* of particular size. By doing this, the problem of overfitting is addressed to a certain degree.

Figure 2.8 shows an example of a 24×9 matrix doing 3×3 max pooling with stride three. The depth is one in this example. Pooling only applies to width and height.

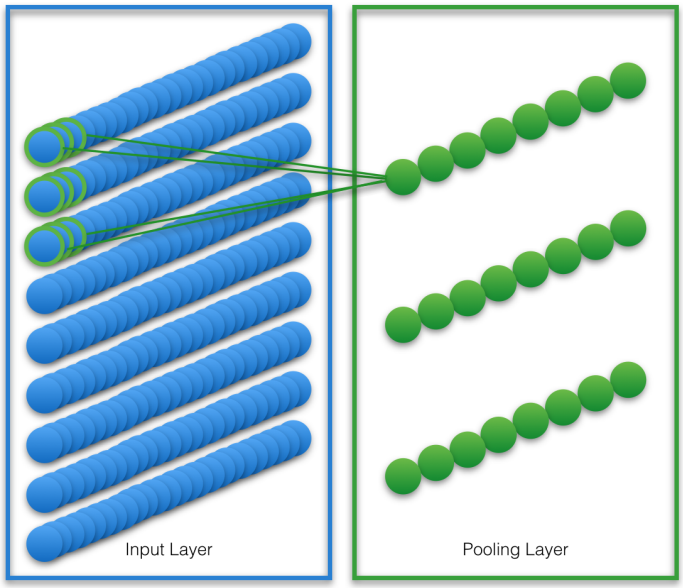

Figure 2.8: Pooling Layer.

The width, height and depth of input are w_i, h_i and d_i. The pooling size is $d_p \times w_p \times h_p$. Usually let $d_p = d_i$. The stride of pooling has the size of $w_s \times h_s$. It is not common to use zero–padding for Pooling layers. The output should have size:

$$d_i \times (\frac{w_i - w_p}{w_s} + 1) \times (\frac{h_i - h_p}{h_s} + 1) \tag{2.6}$$

In most cases, input is pooled non–overlapping, i.e., $w_s = w_p$ and $h_s = h_p$. So the output has size:

$$d_i \times \frac{w_i}{w_p} \times \frac{h_i}{h_p} \tag{2.7}$$

, which reduces the size of output by $\frac{1}{w_p \cdot h_p}$.

In the forward pass, indexes are recorded during pooling in order to do back propagation.

2.6.3 Fully Connected Layer

Convolutional neural networks always have several fully connected layers following convolutional layers. Neurons in fully connected layers have full connections with all neurons in

the previous layer. The structure of a fully connected layer is same as that of layer in a regular neural network.

2.6.4 Dropout

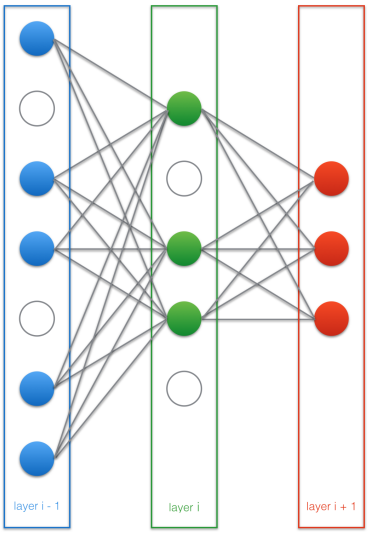

Figure 2.9: Dropout.

Dropout is a technique to prevent neural networks from overfitting and approximate a way to combine exponentially different neural network architectures [49]. When training the model, the unit to be dropped out has a probability p to be *temporarily* removed from the network, as shown in Figure 2.9. It will be ignored when calculating input and output both in the forward pass and the back propagation progress. *Temporarily* means this unit is only dropped out when training this specific sample. This prevents units from co–adapting too much. A layer with n units can be seen as 2^n possible thinned neural networks. When testing the model, all units will not be dropped out and their weights will be multiplied by p. By doing this, 2^n networks with the same parameters are combined into one neural network.

Usually, dropout is applied only to fully connected layers (except the last layer), not to convolutional layers or pooling layers.

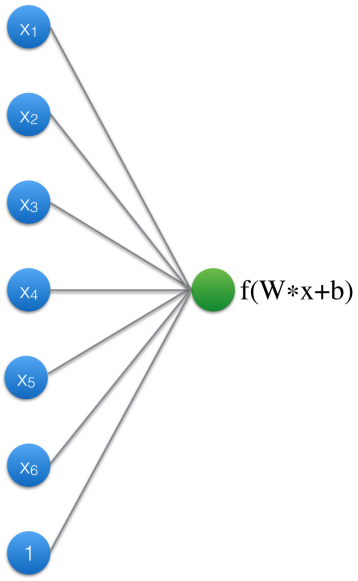

Figure 2.10: Activation Function Applied to a Neuron.

2.6.5 Activation Function and Cost Function

Without activation functions, a layer neural network can only define linear hypotheses. Before calculating the output of a neuron, the value is applied a activation function, as shown in Figure 2.10.

Several activation functions can be applied in neural networks.

- Sigmoid Function

$$f(z) = \frac{1}{1 + exp(-z)} \tag{2.8}$$

$$f'(z) = f(z)(1 - f(z)) \tag{2.9}$$

$f : \Re \mapsto [0, 1]$

- Hyperbolic Tangent (tanh)

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.10}$$

$$f'(z) = 1 - f(z)^2 \tag{2.11}$$

$f : \Re \mapsto [-1, 1]$

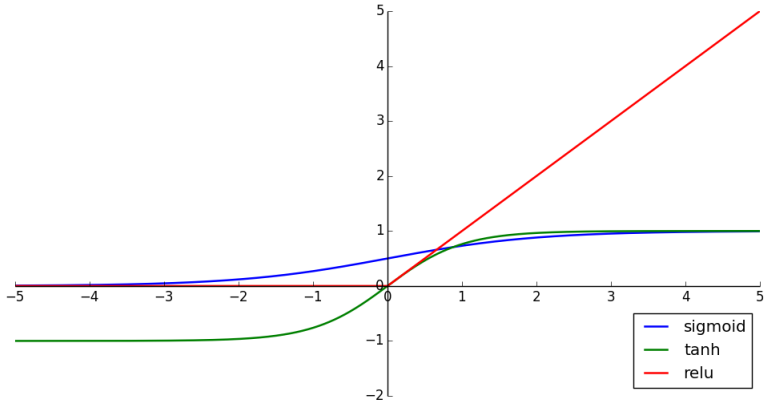

Figure 2.11: Activation Functions. This figure shows sigmoid, tanh and ReLU function. As seen from the figure, sigmoid's output range is $[0, 1]$, while tanh's output range is $[-1, 1]$ and ReLU's output range is $[0, +\infty]$

- Rectifier (ReLU)

$$f(z) = max(0, z) \tag{2.12}$$

$$f'(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geqslant 0 \end{cases} \tag{2.13}$$

$f : \Re \mapsto [0, +\infty]$

Loss function is also called cost function. For multi–label task, binary cross entropy is the most common used loss function.

- Binary Cross Entropy

$$z(t, o) = -(tlog(o) + (1 - t)log(1 - o)) \tag{2.14}$$

2.6.6 Common CNN Architectures

Most common convolutional neural networks follow the pattern below [26]:

$$\begin{aligned} &Input \\ &\rightarrow [(Convolutional \rightarrow Activation)^* \rightarrow Pooling?]^* \\ &\rightarrow (FC \rightarrow Dropout? \rightarrow Activation)^* \\ &\rightarrow Output \end{aligned} \tag{2.15}$$

, where "*" indicates that this layer might be repeated multiple times and "?" stands for optional occurring. Input layer is followed by multiple convolutional and pooling layers, then is followed by several fully connected layers with dropout.

Chapter 3

Related Work

Convolutional neural networks have been widely used in POS tagging [43], chunking, NER, semantic role labeling [10], searching queries and Web documents [44], sentence classification [13] [27], semantic modelling [25], relation classification [14], and other NLP tasks.

3.1 Single–Convolutional–Layer CNNs

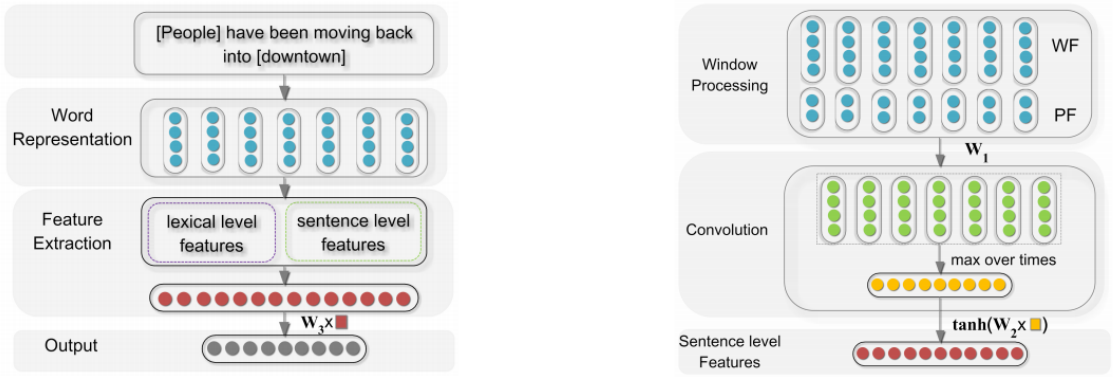

Figure 3.1: Neural Network for Relation Classification and Framework for Extracting Sentence Level Features [55]. In the right hand figure, WF stands for word features and PF stands for position features.

Zeng et al. [55] exploit a neural network to classify questions. Lexical level features are extracted from word embeddings. Sentence level features are learned by a one layer convolutional neural network. Then both lexical and sentence level features are fed into a neural network to predict the relationship of two given nouns in a sentence, as shown in Figure 3.1. This model is experimented on the SemEval–2010 Task 8 dataset (a question set with 10 labels [20]).

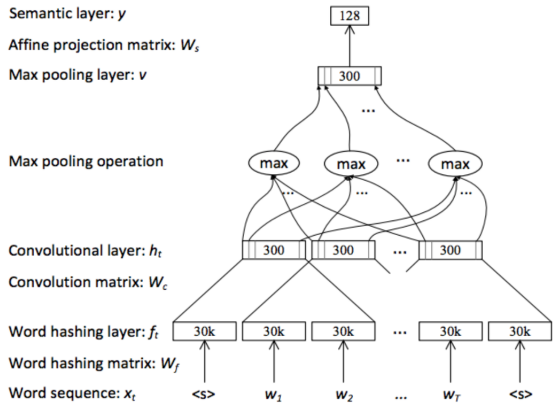

Figure 3.2: CNN Model [44].

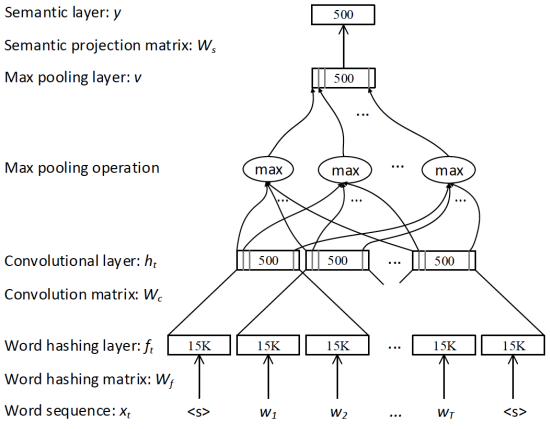

Figure 3.3: CNN Model [51].

Shen et al. [44] and Yih et al. [51] present similar convolutional neural networks. They both transform word into vector using letter–tri–gram. Then word vectors are fed into a convolutional layer, followed by a max over–time pooling layer and a fully connected layer as output layer. Figure 3.2 shows the model for queries and web documents searching [44]. Shen et al. [51] test the model on a question set from a commercial search engine. Yih et al. use a question dataset, which is the same dataset as in this thesis, and train a model for relation extraction and another model for entity extraction, as shown in Figure 3.3. The authors define this problem as a multi–class classification, i.e., given a query returning one relation each time while returning 150 top–scoring candidates.

Kim [27] trains a network with one convolutional layer followed by a max–over time pooling, and a fully connected layer with dropout and softmax output layer for sentence classification, as shown in Figure 3.4. This "one convolutional layer" consists of 3 parallel convolutional layers with different filter sizes. The model is trained with two channels – only the parameters of one channel are updated in training progress. word2vec is input feature. This model is experimented on: movie reviews with positive/negative labels [40] (MR), Stanford Sentiment Treebank, (SST–1, which is dataset with movie reviews with

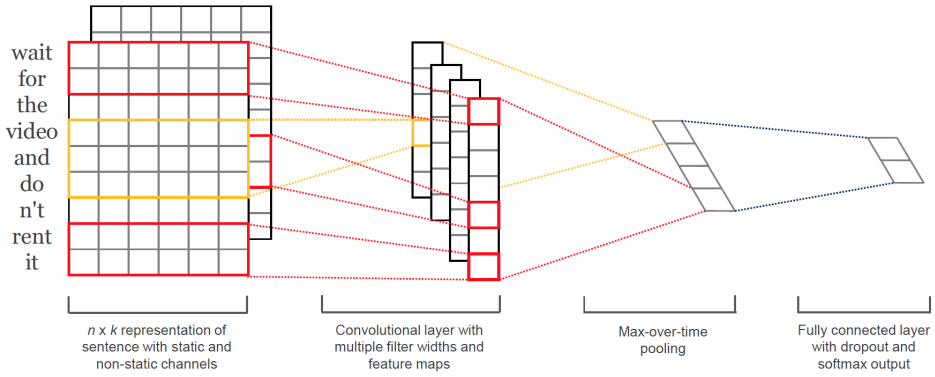

Figure 3.4: CNN Model for Several Sentence Classification Tasks [27].

very positive, positive, neutral, negative, very negative label [48]), same dataset as SST–1 but only positive/negative labels, sentences with subjective/objective labels [39] (Subj), Text REtrieval Conference question dataset with 6 labels [32] (TREC), customer reviews with positive/negative labels [23] (CR), opinion dataset with positive/negative labels [52] (MPQA).

3.2 Multi–Convolutional–Layer CNNs

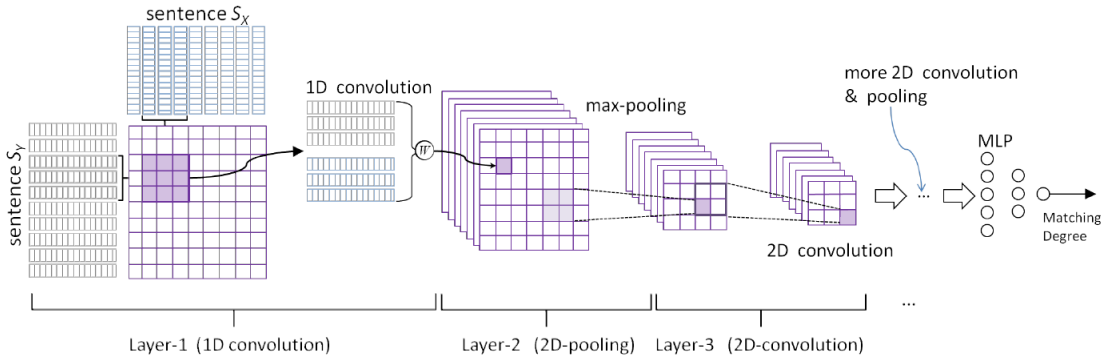

Figure 3.5: ARC–II Model [22].

Hu et al. [22] propose a convolutional neural network model for matching sentences. The authors apply a 1 dimension convolution followed by 1 dimension max pooling, multiple 2

dimension convolutions and pooling layers, and multiple fully connected layers, as shown in Figure 3.5. It takes embedding of words in the sentences aligned as input and outputs matching degree. The approach is tested on sentence completion [31], matching a response to Weibo, and MSRP dataset [42]

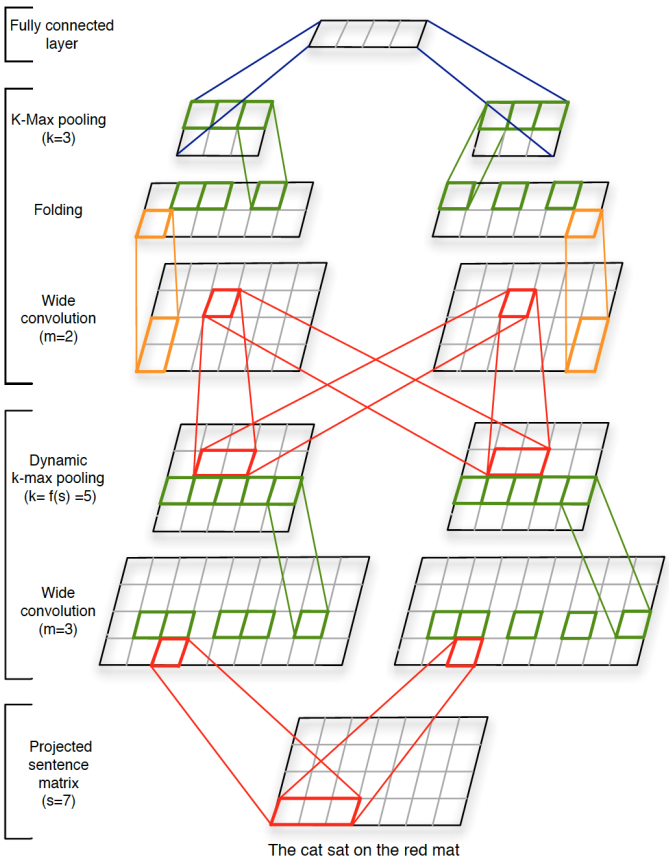

Figure 3.6: DCNN Model for Modeling Sentence [25].

Kalchbrenner et al. [25] design a Dynamic k–Max Pooling Convolutional Neural Network (DCNN) for sentence modelling. The authors apply several wide one–dimensional convolution layer followed by feature maps folding operation and k–max pooling layer, and a fully connected layer as output, which is shown as Figure 3.6. K–max pooling is to chose k highest values among inputs and keep their original orders. This model is tested on SST–1, SST–2, 6–type question categorization in the TREC dataset, and Twitter sentiment prediction task (tweets with positive/negative labels). Compared with Kim's model,

DCNN performs better on SST–1 and TREC, while worse on SST–2.

CNNs with only one convolutional layer have good performance on different tasks. Kim's design [27] consists of parallel convolutional layers which is different from others'. Some other authors propose deeper CNNs. This thesis proposes a single–layer CNN and two multi–layer CNNs and compares their performance on different datasets.

Chapter 4

Dataset and Environment

4.1 Dataset

This thesis focuses on classifying single–relation questions. Example questions of this type include: "What is the birthday of Barack Obama?", "Where does a giant swallowtail butterfly live?". Single–relation questions are the most common type of questions observed in various community QA sites [16]. In a single–relation question, relation and entity are two elements to be understood. After the relation and the entity are extracted, the answer can be generated from knowledge base (KB) such as Freebase [2], DBpedia [1], etc.

4.1.1 Data Format

In this thesis, we download the dataset from knowitall.cs.washington.edu/paralex/ [16]. These questions are crawled from WikiAnswer. Typos and grammar errors are very common in the dataset. "labeled.txt" contains 608,650 examples. Each example is in the form of $(question, query1, query2, \ldots)$, for instance, "*what be the difference btw isolation transformer and step up and step down transformer ? 2 1 755225 1605804 2 1 775854 2464747 2 1 887236 1605804 2 1 890251 1605804 2 0 1503166*". Each query is encoded in the form $2\#ORDER\#REL\#ENT$. $\#REL$ represents index of relation constants in the query which can be looked up in "vocab.txt", such as *755225* stands for *be–function–of.r*. Each question might have several relations. All question–queries list tuples are processed to generate question–relations list tuples in the form of $(question, \#REL1, \#REL2, \ldots)$. The example mentioned above becomes "*what be the difference btw isolation transformer and step up and step down transformer ? 755225 775854 887236 890251 1503166*".

4.1.2 Answerable Queries Coverage

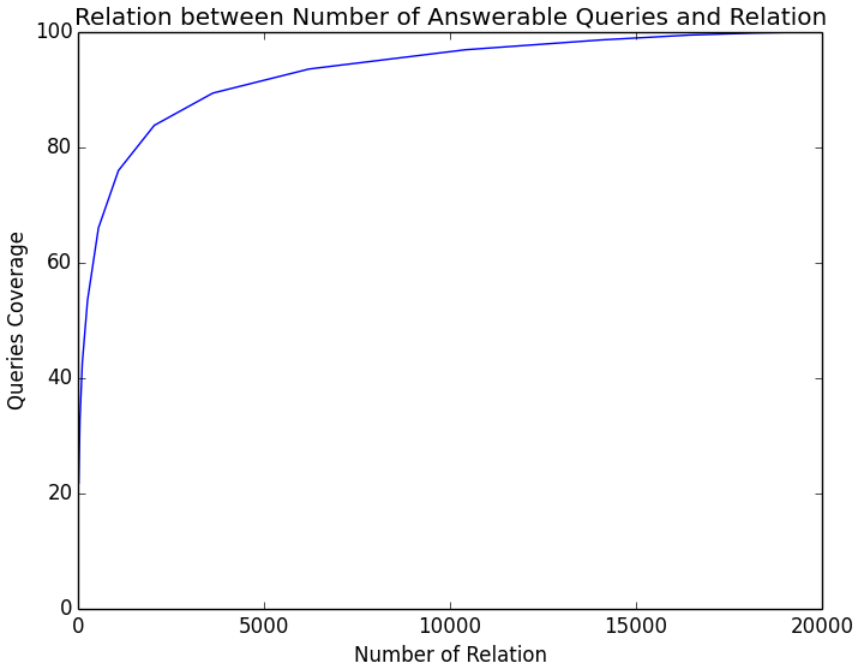

Figure 4.1: The Curve of Coverage of Queries Answerable and Number of Relation.

Figure 4.1 shows the relation between the coverage of answerable queries and the number of relations. This curve increases steeply before coverage reaches 80% but grows slowly later. 21 relations cover more than 20% of the queries. But it requires about 16,000 additional relations to increase the coverage from 80% to 100%. It makes sense that most of the problems people concern on WikiAnswer are only a small subset of knowledge.

4.2 Tool and Environment

4.2.1 word2vec

Word2vec is a continuous distributed representation of words [33] [35] [36]. A 300 dimensional vectors trained on part of Google News dataset which contains 3 million words and phrases are used in this experiment.

4.2.2 NER

NER labels named entities. Stanford NER has three models: a four–class model trained for CoNLL, a seven–class model trained for MUC and a three–class model trained for both [17] [19]. Models support both capitalization sensitive and ignored classifiers.

4.2.3 CUDA

In order to run experiments on GPU, *CUDA driver* and *CUDA Toolkit* are needed for Nvidia's GPU–programming toolchain. CUDA Toolkit is downloaded from developer.nvidia.com, which contains an *nvcc* program – a compiler for GPU code.

4.2.4 Python Libraries

- *Theano* is a Python library that defines, optimizes, and evaluates mathematical expressions involving multi–dimensional arrays efficiently and has transparent use of GPU [5] [7] [28].
- *Keras* is a Theano–based deep learning Python library [3]. Keras is used as library to build CNNs.
- *scikit–learn* is a Python library for machine learning [4]. scikit–learn is for building a support vector classifier.

4.2.5 Spelling Corrector

Spelling Corrector is a tool to correct typos [38]. It is used to correct spelling errors in dataset with pre–trained 3 million words and phrases (GoogleNews–vectors–negative3000) as dictionary. As questions of are crawled from WikiAnswer, dataset contains many typos and grammar error. Spelling corrector is helpful to remove noises in dataset.

4.2.6 Experiment Environment

All the experiments are tested with the computer with configuration described as follows:

- OS system: Ubuntu 14.04LTS

- Processer: Intel Core i7–4790K CPU @ 4.00GHz × 8
- Memory: 16GB 1333 MHz DDR3
- GPU: NVIDIA GeForce GTX TITIAN X
- JDK: 1.8.0_45
- Python: 2.7.6
- NumPy: 1.9.2
- SciPy: 0.15.1
- Theano: 0.7.0
- Keras: 0.1.1

Chapter 5

Main Results

5.1 Methodologies

5.1.1 Data Processing

Raw Data

Dataset Index	#Sentences	#Classes	Min #Sentences per Class
1	138280	22	8000
2	202002	55	4000
3	261681	120	2000

Table 5.1: Three Datasets. Each subset is chosen from the whole dataset according to minimum sentences per class. For example, for subset 1, only classes which have more than 8000 sample sentences will be chosen.

Three datasets were used for experiments. Table 5.1 shows the number of sentences, classes and the minimum number of sentences per class of each dataset. Each dataset splits $\frac{1}{10}$ for test[1].

[1]Actually, k–fold cross validation is better than conventional validation. The former more properly estimates model prediction performance. But training a model takes up to 9000s per epoch. 10–fold cross validation needs 10 times training and testing processes. As the size of dataset is very large, and results of different choice of subset to be the test set do not differ with each other a lot. So we still use $\frac{9}{10}$ as train set and $\frac{1}{10}$ as test set.

Then sentences were corrected spelling errors using GoogleNews–vectors–negative300 as a dictionary. All name entities in sentences were replaced by "LOC", "PER" and "ORG" with Stanford NER [36] 3 class model trained for both CoNLL and MUC in capitalization ignored mode.

Sentence Space

As described in Subsection 4.2.1, each word can be represented by a 300D floats vector, e.g.,a word is a 1×300 vector. A sentence of w words can be represented by a $w \times 300$ matrix. For convenience, we add a dummy depth dimension to make a sentence a 3D tensor, as shown in Figure 5.1.

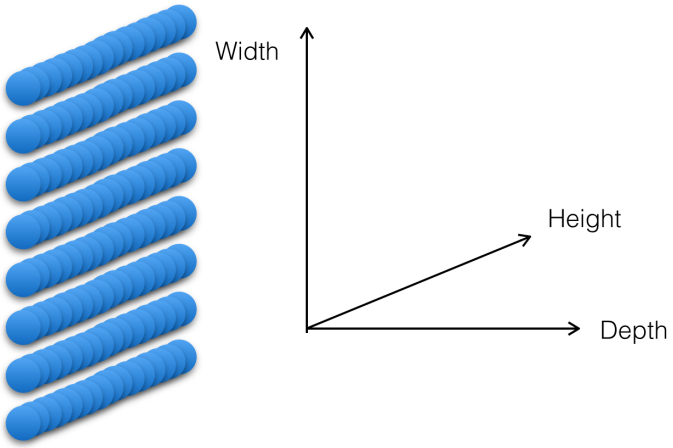

Figure 5.1: Sentence Space. Depth is one, width is the number of words in a sentence, and height is three hundred which is the dimension of word2vec.

Actually, after using Spelling Corrector to correct typos, there are still many unknown words in the datasets. Some of them are caused by connecting two words without a space, like "*thechinese*", others are noise, such as "*httpwikianswerscomqwhat*". There are three strategies to assign a vector to unknown words:

- Randomly generate a 300 dimensional vector.
- Assign word2vec with closest edit distance.
- Assign word2vec with closest position distance.

Position distance of two words is defined as Manhattan distance between their letter–bigrams vectors. Given a word, after adding word boundary symbols, we divide it into a sequence of letter–bigrams. For example, '*word*' is divided into '^*w*', '*wo*','*or*','*rd*' and '*d$*', $v(word) = (0, \ldots, 1, \ldots, 1, \ldots, 0)$ where 1s are indexes of '^*w*', '*wo*','*or*','*rd*' and '*d$*' in the bigrams dictionary.

Since each sentence in the dataset has at most 18 words, it can be represented by a $1 \times 18 \times 300$ tensor. Zero paddings are added at the end of a sentence if it has less than 18 words.

Data Padding

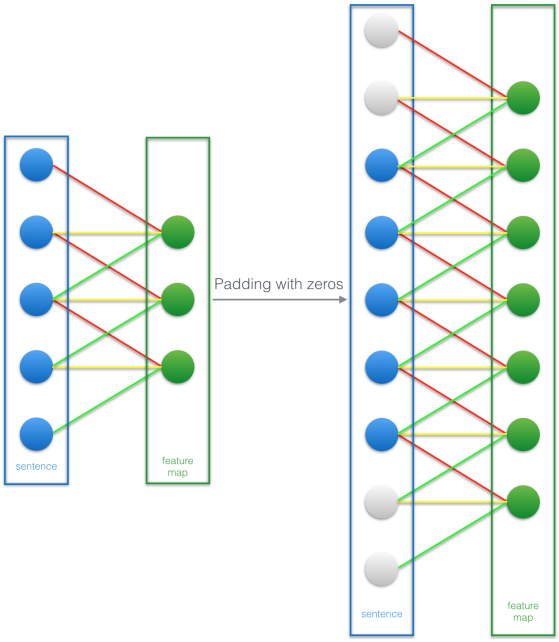

Figure 5.2: Padding Zeros.

Words at the margins are considered fewer times than words in the middle. It is unfair to marginal words. Figure 5.2 shows an example. The convolutional filter has size of three. The first and the last neurons are convoluted once, while the second and the forth neurons are convoluted twice. A simple solution is to add dummy words before and after the sentence, which actually implements *wide convolution* [25]. In the above example, after padding two zeros at the beginning and the end, filter weights can reach all range of

input and each input is convoluted 3 times. The feature map of the narrow convolution is a subsequence of the feature map of the wide convolution. The input of CNN is a $1 \times 26 \times 300$ tensor.

5.1.2 Multi–Label Classification

Sentence	Relations
The old name of Bangkok?	bangkok.r be–in–in.r be–last–name–of.r be–official–name–of.r be–other–name–for.r be–real–name–of.r be–traditional–name–for.r use–to–be–call.r
What be SSID broadcasting?	be–function–of.r be–know–as.r be–purpose–of.r be–requirement–for.r function.r have–function–of.r have–purpose–of.r
What be the primary duty of judical(typo: judicial) branch?	be–judicial–branch–of.r be–primary–of.r be–primary–responsibility–of.r be–primary–role–of.r be–role–of.r have–role–of.r

Table 5.2: Multi–Label Task for Relation Classification Example.

Given a sample, the output of a multi–class classification task has one dimension, but multiple possible values, for example, given a picture the model predict it to be a leaf. But the output of a multi–label classification task has multiple dimensions such as predict it to be leaf, yellow and autumn. Table 5.2 is an example to show that given one question, there might be several possible relation candidates.

Label Space

In supervised learning, training set has a set of examples in the form of (x, y) such that x is the feature and y is the label. X is the input space while Y is the output space. The number of label dimensions of each sample is not constant. Sample s_1 might have label $y = (y_1, y_2, y_3)$ while sample s_2 might have label $y = (y_1, y_4)$. But the dimensions of output space Y is limited. First of all, if the dimensions of Y is N, these labels can be indexed from 0 to $N - 1$. Then $\forall y \in Y$ can be represented by an N dimension binary vector like $(0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0)$ where indexes $(i_1, \ldots, i_k)$ of 1s indicate $y = (i_1, \ldots, i_k)$.

5.1.3 Models

CNN Models	Deep CNN	Parallel CNN
1^{st} **Conv Layer**	(27, 1, 3, 1)	(512, 1, 2, 300) (512, 1, 3, 300) (512, 1, 4, 300) (512, 1, 5, 300)
2^{nd} **Conv Layer**	(2048, 27, 3, 300)	-
Pooling Size	(22, 1)	(25, 1) (24, 1) (23, 1) (22, 1)
1^{st} **Fully Connected Layer**	(2048, 256)	(2048, 256)
2^{nd} **Fully Connected Layer**	(256, #Classes)	(256, #Classes)

Table 5.3: Parameters for CNNs.

Deep CNN

Figure 5.3 shows the structure of Deep CNN. Each sentence tensor is *convoluted* with a filter of size $27 \times 1 \times 3 \times 1$. The number of feature maps produced by the first convolutional layer is 27 and the semantic window size is 3. According to Equation 2.4 the output of this layer has a volume of $27 \times 24 \times 300$. Then the output of the first convolutional layer is *convoluted* with a filter with size of $2048 \times 27 \times 3 \times 300$ where 2048 is the number of feature maps produced by the second convolutional layer, 27 is the depth of input, 3 is the semantic window size, and 300 is the same as the dimensions of word2vec. The layer

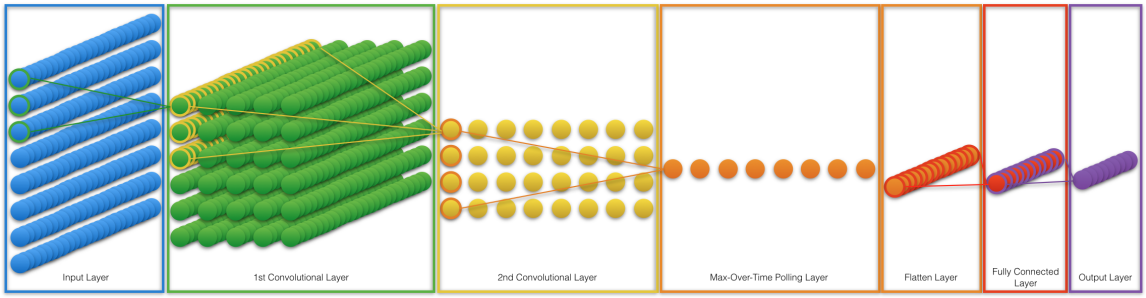

Figure 5.3: Deep CNN.

produces a $2048 \times 22 \times 1$ output. This layer learns local semantic features *up to 5 word wide*, which means that it can capture not only feature of consequent 5 words, but also phases like "take ... into consideration", "where do ... live". Filters' strides of these 2 layers are $f_s = 1$ and all these convolutional layers utilize $ReLU$ activation function.

Each feature map is a sequential local semantic coding of the sentence. To choose the most significant global feature of the sentence, each feature map h_k:

$$\vec{h_k} = [c_1, c_2, \ldots, c_{22}], (k = 0, 1, \ldots, 2048) \tag{5.1}$$

is applied with a *max–over–time pooling* along depth [10] as follows:

$$\hat{h_k} = max\{c_1, c_2, \ldots, c_{22}\}, (k = 0, 1, \ldots, 2048) \tag{5.2}$$

The pooling size is set to be 22×1. According to Equation 2.7, the pooling result has a size of $2048 \times 1 \times 1$.

Then the global semantic feature chosen by max–pooling layer is flattened, which is fed to the following *fully connected layer* with 0.5 *dropout* rate and $ReLU$ activation function. This fully connected layer will take all significant global semantic features into consideration when producing output.

At last, a *fully connected layer* with *sigmoid* activation function is followed as the output layer.

Appendix A.1 shows implementation for building Deep CNN model.

Binary cross entropy is chosen as the loss function.

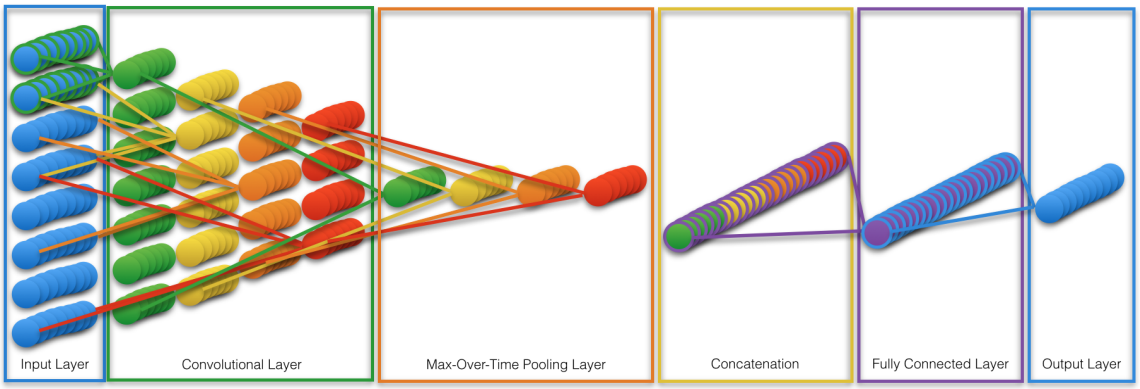

Figure 5.4: Parallel CNN.

Parallel CNN

Parallel CNN is shown in Figure 5.4. It convolutes input with 4 different size filters (f_0, f_1, f_2, f_3) respectively. Filters strides $f_s = 1$. Depth of output $d_o = 512$ (i.e., 512 feature maps). These filters have different widths $w_f = 2, 3, 4, 5$. Different size filters can learn different length of phrases. Then all these convolutional layers apply $ReLU$ activation function. Feature maps (h_0, h_1, h_2, h_3) produced by these 4 different filters have sizes of $(512 \times 24 \times 1)$, $(512 \times 22 \times 1)$, $(512 \times 23 \times 1)$, and $(512 \times 22 \times 1)$. Each feature map h_k^j:

$$\vec{h_k^j} = [c_1, c_2, \ldots, c_{26-w_f^j+1}], (k = 0, 1, \ldots, 512; j = 0, 1, 2, 3) \tag{5.3}$$

is applied with a *max–over–time pooling* along depth as follows:

$$\hat{h_k^j} = max\{c_1, c_2, \ldots, c_{26-w_f^j+1}\}, (k = 0, 1, \ldots, 512; j = 0, 1, 2, 3) \tag{5.4}$$

Pooling sizes are 25×1, 24×1, 23×1, 22×1 corresponding to different sizes of filters. Then all the feature maps produced by different filters are *concatenated* and flattened, which is a fed to a *fully connected layer* with 0.5 *dropout* rate and $ReLU$ activation function.

At last, a *fully connected layer* with *sigmoid* activation function is followed as the output layer.

Appendix A.2 shows implementation for building Parallel CNN model.

ADADELTA

ADADELTA is a kind of gradient descent learning rate method [54]. It takes each dimension's first order information into consideration when updating learning rate. This approach has several advantages:

- Dynamic learning rate per dimension.
- Small amount of computation each iteration.
- Hyperparameters chosen do not affect result significantly.

In the experiment, $\epsilon = 1e^{-6}$, $\rho = 0.95$, and $\eta = 1$ are set to be hyperparameters of ADADELTA, where ϵ is a constant controlling the decay rate and η is a global learning rate shared by all dimensions.

5.1.4 Evaluation Metrics

As described in Section 5.1.2, each sample (query) might have several relation candidates. Label of each sample is converted into an N–dimensional sparse binary vector, where N is the number of all possible relations.

When evaluating the result, *precision*, *recall* and F_1 are chosen to be evaluation metrics.

$$Precision = \frac{TP}{TP + FP} \tag{5.5}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.6}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5.7}$$

, where T stands for true, F for false, P for positive and N for negative. For example, if the ground truth of a sentence s_1 is (y_1, y_2, y_3), and the predicted label is (y_1, y_4), then $Precision = \frac{1}{2}$ and $Recall = \frac{1}{3}$. F_1 score is their harmonic mean $F_1 = \frac{2}{5}$

There are three methods to calculate average statistic: *micro–average*, *macro–average* and *sample–average*.

- Micro–average: calculate metrics by counting the total TP, FN and FP globally.

	Dataset	Micro–Ave. F1	Macro–Ave. F1	Sample-Ave. F1
Baseline - SVC	1	0.51	0.49	0.37
	2	0.43	0.41	0.33
	3	0.39	0.34	0.30
Parallel CNN	1	0.68	0.68	0.58
	2	0.50	0.45	0.37
	3	0.38	0.29	0.27
Deep CNN	1	0.72	0.72	0.64
	2	0.64	0.60	0.53
	3	0.50	0.44	0.39

Table 5.4: F_1 Scores of Different Models. SVC stands for a linear kernel Support Vector Classifier.

- Macro–average: calculate metrics of each label and get their average value.
- Sample–average: calculate metrics for each instance and get their average value.

Jackson and Moulinier [24] state "No agreement has been reached ... on whether one should prefer micro– or macro–averages in reporting results. Macro–averaging may be preferred if a classification system is required to perform consistently across all classes regardless of how densely populated these are. On the other hand, micro–averaging may be preferred if the density of a class reflects its importance in the end–user system". Micro–average and macro–average have different statistical meanings. Sample–average is for multi–label classification. So we use all these three strategies in this thesis.

5.2 Experiments and Results

Table 5.4 shows results of different models. We use a linear kernel Support Vector Classifier (SVC) as the baseline model. This SVC fits one classifier per class and is handled according to a one–vs–the–rest scheme, which means for each classifier the class is fitted against all the other classes. SVC, Parallel CNN and Deep CNN are tested on dataset 1, 2, and 3. All CNN models are trained with 30 epochs.

5.2.1 Baseline versus CNNs

Results show that Parallel CNN and Deep CNN perform remarkably well, giving competitive results against SVC, on dataset 1 and dataset 2, while Parallel CNN has similar results on dataset 3 compared with SVC.

5.2.2 Parallel versus Deep CNN

Deep CNN has better performance than Parallel CNN on dataset 1, 2 and 3. Deep CNN has two convolutional layers while Parallel CNN has four parallel convolutional layers. Although the number of feature maps learned by the second convolutional layer in Deep CNN (2048) is the same as that of the convolutional layer in Parallel CNN, and maximum length of phrases learned by convolutional filters of both models are five, the multi–layer CNN is more powerful than the single–layer CNN.

We can either increase the number of convolutional layers or the number of neurons within each layer to improve the performance by sacrificing training time.

5.2.3 Further Observations

In Deep CNN, the first convolutional layer acts as masks of phrases, which masks "which predator eat leopard" into "which X eat X". The second convolutional layer learns phrases. Table 5.5 shows samples recognized by the filters of the second convolutional layer. Each filter in the 2^{nd} convolutional layer learns at least one phrase ideally. By stacking feature maps learned by different filters, the convolutional layer can recognize large number of phrases.

Figure 5.5 shows precision against recall at various threshold settings of each class on dataset 1. A high area under the PR curve (AUC) represents both high recall and high precision. AUCs range from 0.66 to 0.95. Average AUC is 0.70. Figure 5.6 shows TP against FP at various threshold settings each class on dataset 1. Receiver operating characteristic (ROC) is another metric for evaluating the performance of a binary classifier. ROCs range from 0.90 to 0.99. Average ROC area is 0.97. Closer the curve is to upper left corner, lower the false negative rate and false positive rate are.

Previous results show that the proposed CNNs have good performance on sentence classification. Convolution works well on learning local features (phrases) and max pooling can choose the most significant global feature. As the dataset gets larger, a small CNN is

Index of Neuron	Sentence Samples	Outputs of Neuron
105	be the ultimate warrior	5.25173
	about *the importance of* the	4.70589
	what *the importance of* the	4.61927
	be *the importance of* the	4.49637
	fact about asteroid ? $	4.41866
	be ther *importance of* the	4.25051
	and *the importance of* production	4.11493
	be *the importance of* science	3.708
	be *the importance of* mountain	3.69332
	be *the importance of* herbal	3.69251
137	*the population of* bilous be	5.32066
	the population of LOC of	4.36888
	the population trend of LOC	4.30513
	the population of LOC tx	4.25834
	the population of LOC ny	4.18075
	the population of LOC LOC	4.14987
	the population of LOC beach	4.14069
	the population of LOC only	4.12767
	the population of LOC be	4.09867
	the population o LOC ?	4.08928
283	*name of who invent* the	4.60916
	the people who invent the	3.68543
	the person who invent softball	3.63067
	what cosmic radiation bombard the	3.51603
	^ scientist *who invent* the	3.5001
	^ 8 *who invent* the	3.4487
	^ ^ *who invent* the	3.40215
	black person *who invent* the	3.39316
	slam dunk *who invent* the	3.29019
	the first pen *invent* at	3.27618

Table 5.5: Samples Ranking with Descending Scores of Neurons in 2^{nd} convolutional layer. This result is trained on dataset 1 with Deep CNN. 137^{th} neuron in 2^{nd} convolutional layer learns features of phrases like *the population of*. In the table, ^ and $ stand for blank words before and after the sentence.

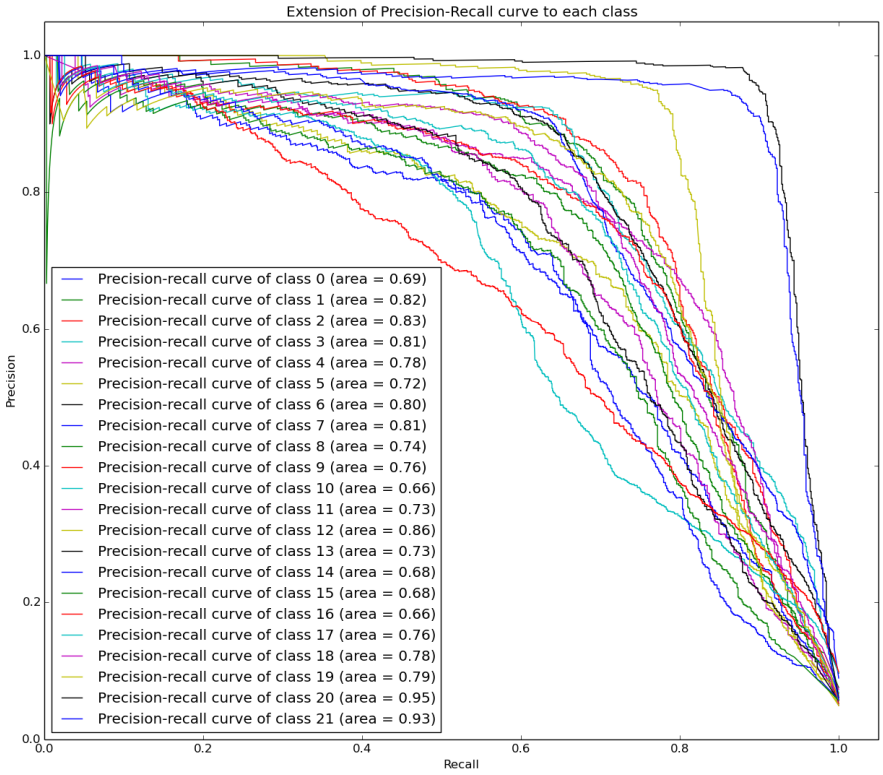

Figure 5.5: Precision Recall Curves.

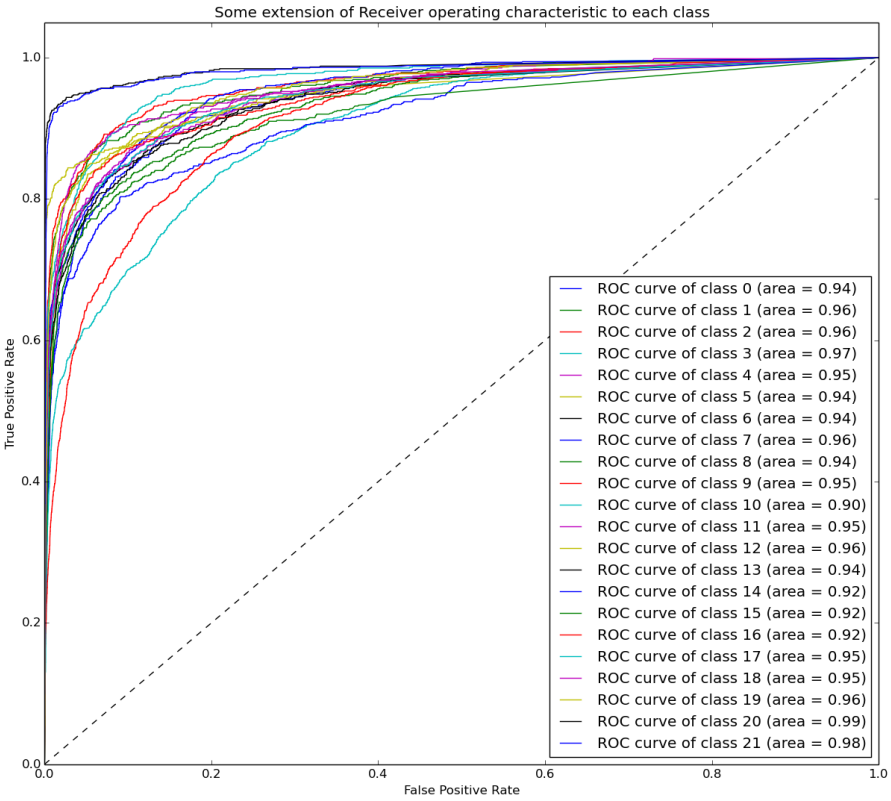

Figure 5.6: Receiver Operating Characteristic Curves.

no longer capable of learning all the information, thus we need to increase the number of parameters in the network, so our experiments compared performance of models with different number of parameters. Tradeoff between performance and efficiency was a criterion in choosing models.

Chapter 6

Conclusions

6.1 Summary

This thesis defines a multi–label classification problem for extracting the relation candidates from a question. We propose two deep learning methods, Parallel CNN and Deep CNN, to predict multiple relation candidates of a question. Parallel CNN consists of four parallel convolutional layers, while Deep CNN has two stacked convolutional layers. Convolutional layers of both the models capture local semantic features. A max over time pooling layer is placed on top of the last convolutional layer to select global semantic features, followed by a fully connected layer with dropout to summarize the features. Our experiments show that both Parallel and Deep CNN outperform the traditional Support Vector Classification (SVC)–based method by a large margin. Furthermore, we observe that Deep CNN has better performance than Parallel CNN, indicating that the deep structure enables much stronger semantic learning capacity than the wide but shallow network. Our method lays a firm foundation for implementing a high performance QA system.

6.2 Future Work

In this work, we chose several hundreds of most frequently appeared relations as our target. However, the overall dataset contains more than 19,000 relations and many of them are rarely present. The limitation of our model is that it is impossible to adapt our CNNs for the 19,000 relations recognition problem by simply adding more output neurons. One

possible solution could be first hierarchically clustering all the relations, and then using cascade network classifiers for each cluster.

On the other hand, the word2vec model we used in this work was trained on the Google News corpus. As shown in [50], the word2vec space trained in social media corpus is different from that in carefully edited prose. Thus, we could retrain the word2vec on the WikiAnswer corpus toward improving our performance.

In summary, we conclude that following steps to complete a KB–supported QA system are needed:

- Map the relations predicted by the model to those used in the knowledge database, such as DBpedia or Freebase (offline).
- Recognize the entities with POS tagging, or NER, or other NLP tools.
- Query the database with the entities and mapped relations.
- Retrieve the answer from the database and generate a sentence accordingly.

APPENDICES

Appendix A

Python Implementation for Building CNNs

A.1 Deep CNN

```
def Deep_CNN():
  #Two Convolutional Layers with Pooling Layer
  model = Sequential ()
  model.add(Convolution2D(27, 1, 3, 1, border_mode='valid', activation='
                                    relu'))
  model.add(Convolution2D(2048 , 27, 3, 300 , border_mode='valid',
                                    activation='relu'))
  model.add(MaxPooling2D(poolsize=(21, 1)))

  #Fully Connected Layer with dropout
  model.add(Flatten())
  model.add(Dense(2048 , 256 , activation='relu'))
  model.add(Dropout(0.5))

  #Fully Connected Layer as output layer
  model.add(Dense(256 , len(label_set), activation='sigmoid'))
  adadelta = Adadelta(lr=1.0, rho=0.95, epsilon=1e-6)
  model.compile(loss='binary_crossentropy', class_mode='multi_label',
                                    optimizer=adadelta)
```

A.2 Parallel CNN

```
def Deep_CNN():
  filter_shapes = [[2, 300], [3, 300], [4, 300], [5, 300]]
  pool_shapes = [[25, 1], [24, 1], [23, 1], [22, 1]]

  #Four Parallel Convolutional Layers with Four Pooling Layers
  model = Sequential()
    sub_models = []
    for i in range(len(pool_shapes)):
        pool_shape = pool_shapes[i]
        filter_shape = filter_shapes[i]
        sub_model = Sequential()
        sub_model.add(Convolution2D(512 1, filter_shape[0], filter_shape[
                                            1], border_mode='valid',
                                            activation='relu'))
        sub_model.add(MaxPooling2D(poolsize=(pool_shape[0], pool_shape[1]
                                            )))
        sub_models.append(sub_model)
    model.add((Merge(sub_models, mode='concat')))

    #Fully Connected Layer with dropout
    model.add(Flatten())
    model.add(Dense(2048, 256, activation='relu'))
    model.add(Dropout(0.5))

    #Fully Connected Layer as output layer
    model.add(Dense(256, len(label_set), activation='sigmoid'))
    adadelta = Adadelta(lr=1.0, rho=0.95, epsilon=1e-6)
    model.compile(loss='binary_crossentropy', class_mode='multi_label',
                                        optimizer=adadelta)
```

References

[1] Dbpedia website. http://wiki.dbpedia.org/.

[2] Freebase website. http://www.freebase.com.

[3] Keras website. http://keras.io/.

[4] scikit-learn website. http://scikit-learn.org/dev/index.html.

[5] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[6] Sven Behnke. *Hierarchical neural networks for image interpretation*, volume 2766. Springer Science & Business Media, 2003.

[7] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[8] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

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

[10] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[11] George Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing (receiving 2013 IEEE SPS Best Paper Award)*, 20(1):30–42, January 2012.

[12] Steven B. Damelin and Jr Willard Miller. The mathematics of signal processing. page 232. Cambridge University Press, 2011.

[13] Cícero Nogueira dos Santos and Maıra Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, 2014.

[14] Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics*, pages 626–634, 2015.

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

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

[17] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[18] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[19] The Stanford Natural Language Processing Group. Stanford named entity recognizer (ner). http://nlp.stanford.edu/software/CRF-NER.shtml.

[20] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics, 2009.

[21] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[22] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050, 2014.

[23] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[24] Peter Jackson and Isabelle Moulinier. *Natural language processing for online applications: Text retrieval, extraction and categorization*, volume 5. John Benjamins Publishing, 2007.

[25] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014.

[26] Andrej Karpathy. Stanford cs231n: Convolutional neural networks for visual recognition. http://cs231n.github.io/.

[27] Yoon Kim. Convolutional neural networks for sentence classification. In *Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, 2014.

[28] LISA lab. Theano website. http://deeplearning.net/software/theano/index.html, 2015.

[29] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941, 2015.

[30] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[31] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

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

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[34] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

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

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

[37] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, Caroline Suen, Adam Coates, Andrew Maas, Awni Hannun, Brody Huval, Tao Wang, and Sameep Tandon. Ufldl tutorial. http://ufldl.stanford.edu/tutorial/.

[38] Peter Norvig. Spelling corrector. http://norvig.com/spell-correct.html.

[39] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual*

Meeting on Association for Computational Linguistics, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[40] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

[41] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[42] Vasile Rus, Philip M McCarthy, Mihai C Lintean, Danielle S McNamara, and Arthur C Graesser. Paraphrase identification with lexico-syntactic graph subsumption. In *FLAIRS conference*, pages 201–206, 2008.

[43] Cicero D. Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826. JMLR Workshop and Conference Proceedings, 2014.

[44] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 373–374, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.

[45] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, pages 958–, Washington, DC, USA, 2003. IEEE Computer Society.

[46] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proceedings of the ACL conference*, 2013.

[47] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

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

[49] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[50] Luchen Tan, Haotian Zhang, Charles L.A. Clarke, and Mark D. Smucker. Lexical comparison between wikipedia and twitter corpora by using word embeddings. In *Proceedings of ACL*, Beijing, China, June 2015. Association for Computational Linguistics.

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

[52] Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210, 2005.

[53] Mo Yu, Matthew Gormley, and Mark Dredze. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*, 2014.

[54] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[55] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344, Dublin, Ireland, 2014.