Word and Relation Embedding for Sentence Representation

by

Trideep Rath

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2017 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Baoxin Li
Yezhou Yang

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

In recent years, several methods have been proposed to encode sentences into fixed length continuous vectors called sentence representation or sentence embedding. With the recent advancements in various deep learning methods applied in Natural Language Processing (NLP), these representations play a crucial role in tasks such as named entity recognition, question answering and sentence classification.

Traditionally, sentence vector representations are learnt from its constituent word representations, also known as word embeddings. Various methods to learn the distributed representation (embedding) of words have been proposed using the notion of Distributional Semantics, i.e. "meaning of a word is characterized by the company it keeps". However, principle of compositionality states that meaning of a sentence is a function of the meanings of words and also the way they are syntactically combined. In various recent methods for sentence representation, the syntactic information like dependency or relation between words have been largely ignored.

In this work, I have explored the effectiveness of sentence representations that are composed of the representation of both, its constituent words and the relations between the words in a sentence. The word and relation embeddings are learned based on their context. These general-purpose embeddings can also be used as off-the-shelf semantic and syntactic features for various NLP tasks. Similarity Evaluation tasks was performed on two datasets showing the usefulness of the learned word embeddings. Experiments were conducted on three different sentence classification tasks showing that our sentence representations outperform the original word-based sentence representations, when used with the state-of-the-art Neural Network architectures.

To Mom and Dad

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Chitta Baral for his immense support and guidance throughout my master's degree here at ASU. This work has only been possible because of the knowledge I have gained undertaking his courses. The lessons learnt would help me immensely in my future endeavors.

I would also like to thank my committee members Dr. Baoxin Li and Dr. Yezhou Yang, who offered guidance and support whenever I needed. The valuable suggestions and critical comments have helped me to improve this work significantly.

I would also like to thank Somak Aditya, my friend and collaborator, for his constant motivation and answering all my queries patiently throughout this thesis work. I would also like to thank Arpit Sharma for all the encouragement and his guidance.

Last but not the least, I would like to thank my mom, dad and brother for encouraging me and standing throughout my life - words don't do justice to everything you've done for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Sentence representations play an important part in various Natural Language Processing (NLP) applications. Traditionally, many of the advances in NLP were based on using this meaningful representation of sentences. A representation of a sentence could be sequence of part-of-speech tags of a sentence, parse tree from syntactic parser Chen and Manning (2014), semantic representation from semantic parsers Sharma *et al.* (2015) or a vector representations of a sentence.

Deep learning NLP systems use the vector representation of the sentence as an input to perform various classification or prediction based tasks. Recently, with the advent of various deep learning architectures in NLP, various NLP tasks have achieved state-of-the-art performance like part-of-speech tagging, chunking, named entity recognition, semantic role labelling, syntactic parsing(Chen and Manning (2014), Dyer *et al.* (2015)), sentiment analysis(Socher *et al.* (2013), Kalchbrenner *et al.* (2014)), machine translation(Sutskever *et al.* (2014), Tai *et al.* (2015), Luong *et al.* (2015), Cho *et al.* (2014)), and question answering(Bordes *et al.* (2015), Kumar *et al.* (2015)).

One major reason for the success achieved by these deep learning architectures is due to the learning distributed word representation in vector space (Bengio *et al.* (2003), Mikolov *et al.* (2013a), Pennington *et al.* (2014)). Word representation is a vector for the word and is also termed as word embedding. The distributed representation (embedding) of words are learnt using the notion of distributional Semantics

Firth (1957). In this formulation, instead of using one-hot vectors by indexing words into a vocabulary, word embeddings are learnt by the surrounding words onto a low dimensional and dense vector space that encodes both semantic and syntactic features.

There also have been research to learn the representation of relation, which provides information of how two words are linked in a sentence using the similar methods after parsing the large textual data using dependency parsers Chen and Manning (2014). These relation embeddings have shown to capture rich linguistic and syntactic information Mikolov *et al.* (2013b). The effectiveness of using both word and relation embedding together have also been observed in tasks like knowledge extraction Bordes *et al.* (2011) and aspect term extraction Yin *et al.* (2016)).

The learnt word embeddings are then used to form the sentence representation of a sentence. One simple and common approach for representing a vector of a sentence is by summing or averaging the word embeddings of words participating in the sentence. Various other neural network based modelling methods like Recurrent Neural Network Elman (1990), Convolutional Neural Network LeCun *et al.* (1998), Recursive Neural Network Socher *et al.* (2010) has also been used to form the vector representation of the sentence. But most of the models use word embeddings as the fundamental unit to encode sentence in vector space. There have been few attempts but it still lacks clarity about the effectiveness of using a composite model of both word and relation embeddings for sentence representation.

In our work, we jointly learn the distributed representation of words and relations in the same vector space. The representation of words are learned from its linear context of words and the representation of relations are learned by the word it links after parsing using Stanford dependency parser Chen and Manning (2014). Then using the learnt word and relation embedding we experiment with various composition methods for sentence representation and perform evaluation using three common sentence classification tasks Question classification Li and Roth (2002), Sentiment classification Socher *et al.* (2013) and Subjectivity classification Pang and Lee (2004) on state-of-the-art classification techniques (SVM Cortes and Vapnik (1995), CNN LeCun *et al.* (1998) and LSTM Hochreiter and Schmidhuber (1997)).

1.2 Contribution

To the best of our knowledge, this is the first attempt to explore the effectiveness of the composition based model for sentence representation using word and relation embeddings. To summarize, we make the following contributions:

- We put forward a model to jointly learn the word and relation embeddings which could be used as off-the-shelf features in various NLP tasks.
- We propose a sentence representation, composed of the learnt word and relation embedding and we empirically demonstrate that the composition-based representation outperforms the original skip-gram based word-embedding for various classification tasks across models.
- We make the word and relation embeddings publicly available for the community.

Chapter 2

BACKGROUND

The work presented in this thesis is related to two main research areas: Artificial Neural Networks, Vector space modelling. Below is a brief introduction about these topics.

2.1 Artificial Neural Networks

Neuron

A neuron is the basic computational unit in deep neural network architectures, which is loosely inspired by a biological neuron. It receives n scalar inputs, $x_1, x_2, x_3..., x_n$ and the neuron consist of corresponding weights, $w_1, w_2, w_3..., w_n$ and a bias b. The output y of the neuron is the computed by:

$$y = f(z) \tag{2.1}$$

$$z = \sum_{i=1}^{n} w_i.x_i + b \tag{2.2}$$

Where f is the activation function. The most common activation functions are the Sigmoid, Tanh or ReLu.

The above could also be written in vector form as follows.

$$z = w^T x + b \tag{2.3}$$

Where x is n dimensional vector input, corresponding to the w, n dimensional weight vector and a bias b associated with the neuron.

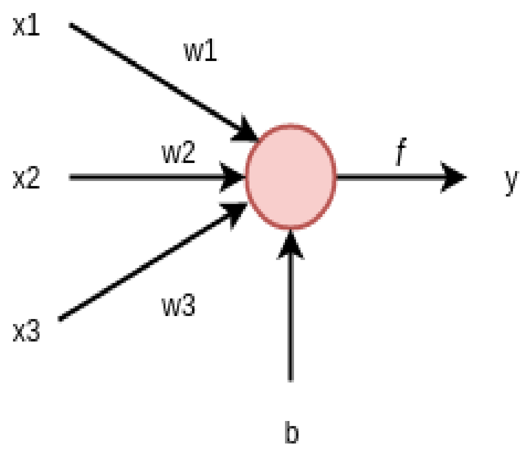

Figure 2.1: Neuron

Multilayer Neural Network

If k neurons share the same input, this is called one-layer neural network with k neurons in its first layer. Now if you stack another layer with l neurons whose input is the output from the first layer then we have a two layer neural network. Theoretical proofs Cybenko (1989) have existed that using two layer neural network with non-linear activation functions are universal approximators i.e. any continuous function can be approximated using this neural network. Now we can perform N-class classification using this neural network with a special output layer called the softmax layer: each neuron in this output layer calculates the probability of the class given an input.

Given a training dataset $D_{train} = \{(x_1, t_1), (x_2, t_2), ..., (x_n, t_n)\}$. We decide a cost function $J(\theta)$, where θ are the parameters i.e weights and bias of the neural network to be optimized to approximate the functional mapping of $x \to t$. $J(\theta)$ is often taken as squared error for regression problems or cross entropy for classification problems. A training algorithm called backpropagation method is employed in order to optimize the w and b of the neural network to reduce the value of cost function. The figure 2.2, shows a neural network with an input x, then two hidden layers with weights W1 and

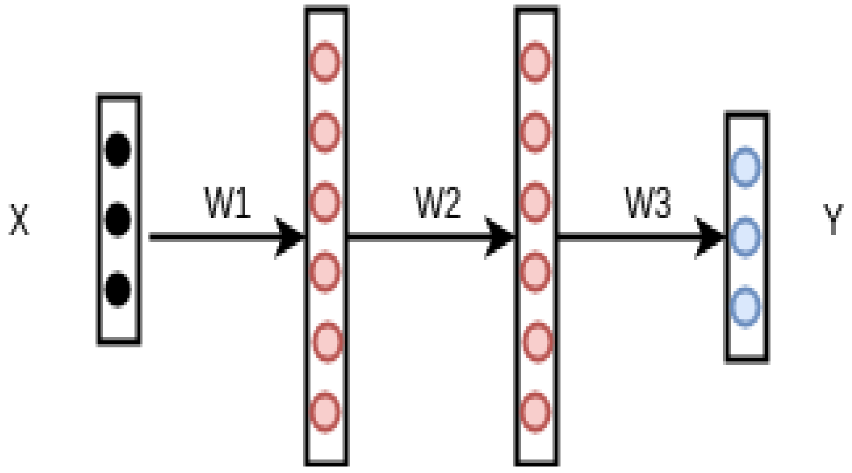

Figure 2.2: Multilayer Neural Network

W2 respectively. The final softmax layer with weight W3 to perform classification.

Recurrent Neural Network

This is a popular neural network architecture which is used to tackle sequence learning problems. As sentence is sequence of words, this architecture is quite popular for various NLP tasks proposed by Elman (1990). A recurrent neural network architecture has at least one directed ring in its structure. This has been used in tasks like machine translation (Sutskever *et al.* (2014) and language modelling (Mikolov *et al.* (2010)).

In simple RNN, a vector x_t is input to the network at time t. The hidden layer h, which has activation h_{t-1} before x_t, which plays a role to capture memory of whole previous input history $(x_0, x_1, x_2, x_3, ..., x_{t-1})$. At time t, the hidden layer updates the activation by:

$$h_t = tanh(Wx_t + Uh_{t-1} + b) \tag{2.4}$$

Where W, U and b are the parameters of the network, *tanh* is a non linear

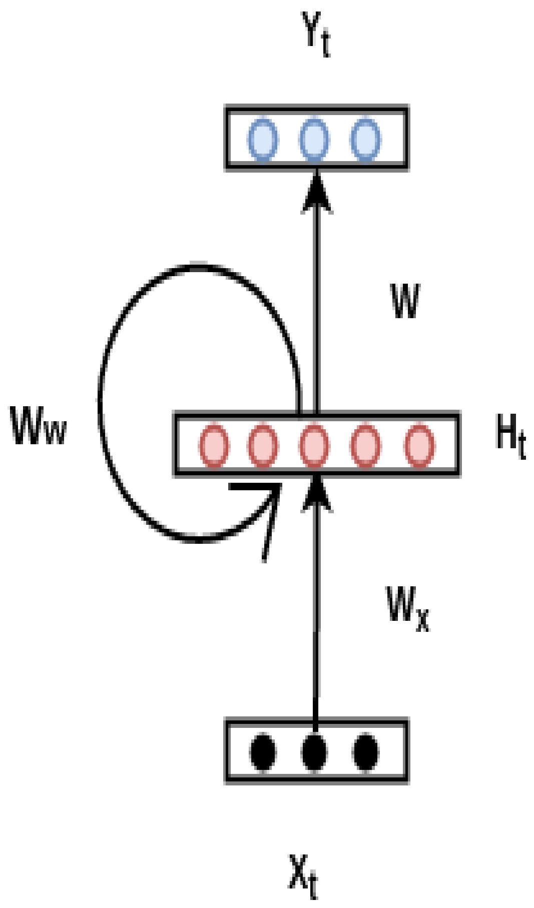

Figure 2.3: Simple Recurrent Neural Network

activation function also called hyperbolic tangent function. The figure 2.3, describes the RNN unit.

Traditional RNN suffer from the problem of vanishing or the exploding gradient, where the gradients decay or can grow exponentially as they propagate over time. To address these problem a variant of this model was proposed, Long Short Term Memory (LSTM) by Hochreiter and Schmidhuber (1997). This new model introduced gating mechanism and vector of memory cells. We thus have the input gate i_t, the forget gate f_t , the output gate o_t, the memory cell c_t, input at time t as x_t, and the hidden state h_t. The LSTM unit manipulates a collection of vectors described by following equations:

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$

$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$

$$u_t = tanh(W^u x_t + U^u h_{t-1} + b^u)$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1}$$

$$h_t = o_t \odot tanh(c_{t-1})$$

Where $\odot$ is the element wise multiplication. σ is the sigmoid function.

Gates i_t, f_t, $o_t \in [0, 1]^d$, control at timestep t how the input is updated, how much previous cell is forgotten, and the exposure of the memory to form the hidden state vector respectively.

Convolutional Neural Network

Originally invented for computer vision, CNN models have shown effectiveness in NLP and have achieved great success for various tasks as in semantic parsing, search query retrieval Shen *et al.* (2014) and sentence modeling Kalchbrenner *et al.* (2014). Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features LeCun *et al.* (1998).

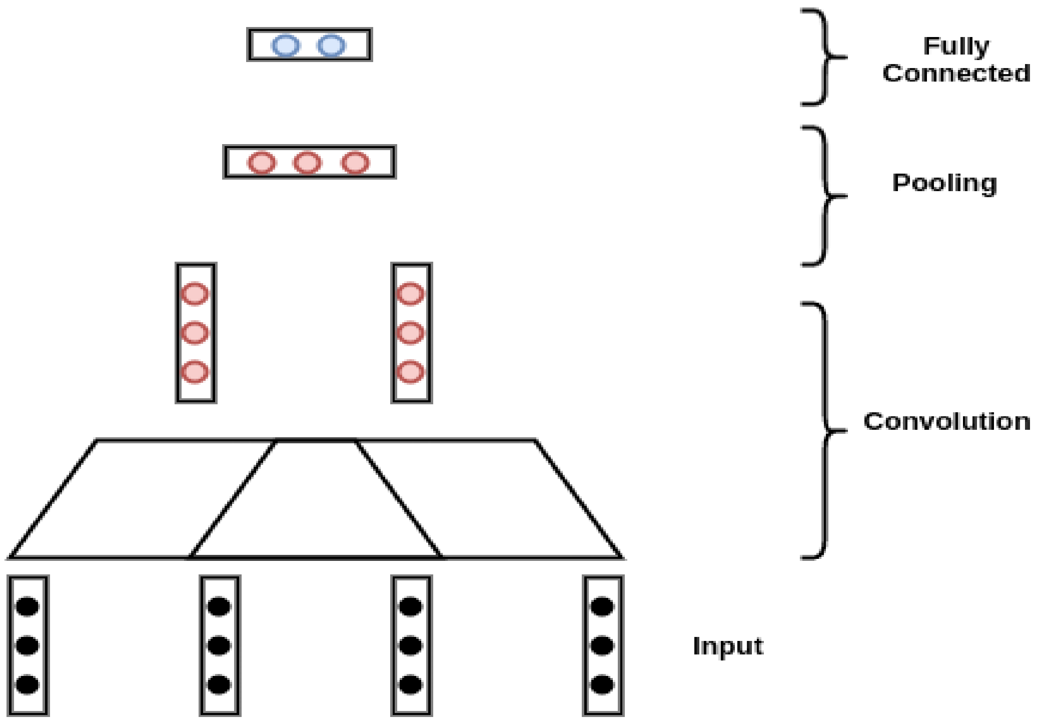

Figure 2.4: Convolutional Neural Network

Let $x_i \in R^k$ be the k-dimensional word vector corresponding to the i-th word in the sentence. A sentence of length n is represented as

$$x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n \tag{2.5}$$

Where $\oplus$ is the concatenation operator. A convolution operation involves a filter $w \in R^{hk}$, which is applied to a window of h words to produce a new feature. A feature ci is generated from a window of words $x_{i:i+h1}$ by:

$$c_i = f(w.x_{i:i+h-1} + b) \tag{2.6}$$

Here b is the bias term; f is a non-linear function such as tanh or Relu. This filter is applied to each possible window of words in the sentence to produce a feature map:

$$c = [c_1, c_2, ..., c_{n-h+1}] \tag{2.7}$$

Max-pooling operation Collobert (2011) over the feature map is taken to take the maximum value

$$\hat{c} = max(c) \tag{2.8}$$

$\hat{c}$ is the feature corresponding to this particular filter. The idea is to capture the most important feature with the highest value for each feature map. This pooling scheme deals with variable sentence lengths. Multiple filters (with varying window sizes) obtain multiple features. These features from this layer are then passed to a fully connected softmax layer whose output is the probability distribution over labels.

2.2 Vector space modelling for NLP

The notion of distributional semantics was introduced by Firth (1957), which means that the meaning of the word is determined by its context words. Building word meaning with distributional semantics is always in an unsupervised learning fashion and done using large textual data. Many approaches to represent word meaning by vectors have been proposed, from simple counting to recent advanced machine learning techniques Collobert *et al.* (2011), Collobert (2011), Mikolov *et al.* (2013b).

Co-occurence count method

Context of a target word in a sentence are all words standing before or after the target word not exceeding k positions. This is called a window context. A very simple method is to count how many times a word u co-occurs with the target word

w. This is equivalent to estimating the conditional distribution $P(U = u|W = w)$ using maximum likelihood. For each word u count the times w co-occurs within a defined window size k. The count vector is extracted containing all of those counts, which is used to represent the target word w. The conditional distribution is simple by:

$$P(U = u|W = w) = \frac{\#u \ and \ w \ co - occur}{\#w} \tag{2.9}$$

Function words (e.g., a, an, how, what), which are extremely frequent, carry little information about the meanings of target words. But in the above approach, every word plays an equal role. Thus a popular weighting scheme is used called point-wise mutual information:

$$PMI(u, w) = \frac{P(U = u|W = w)}{P(U = u)} \tag{2.10}$$

Where $P(U = u)$ is the probability that u appears in the corpus. The vectors produced are high dimensional (e.g. 2000 or more). Dimensionality is reduced is using method like PCA or non-negative matrix factorization.

Prediction based method

Various approaches to learn the embeddings based on prediction have been explored by Bengio *et al.* (2003), Mikolov *et al.* (2013b). In this section we explain the most commonly used and considered state-of-art model for word embeddingst the Skipgram and CBOW model by Mikolov *et al.* (2013b).

Skipgram and CBOW These two model are very similar in their approaches and work in a linear context. For example for the sentence given as a sequence of words: $W_{i-2}, W_{i-1}, W_i, W_{i+1}, W_{i+2}$. If we want to learn the representation of

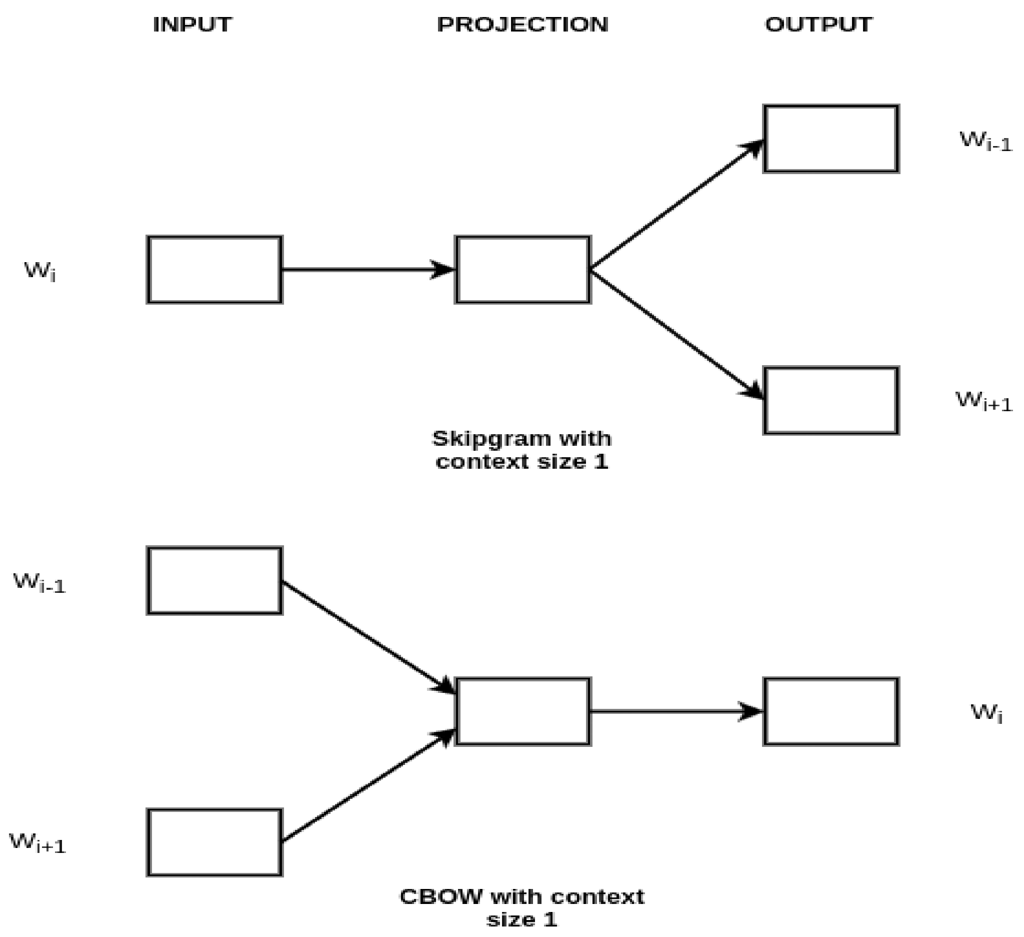

Figure 2.5: CBOW and Skipgram model

the W_i: Skipgram model tries to optimize the probability of the context words i.e $W_{i-2}, W_{i-1}, W_{i+1}, W_{i+2}$ given the center word i.e. W_i; CBOW model tries to optimize the probability of the center word W_i given the context words $W_{i-2}, W_{i-1}, W_{i+1}, W_{i+2}$. This is shown in the 2.5

We explain the skipgram model in detail below. Skipgram tries to optimize the probability of context word(c) given the center word(t). The training is performed on large corpus of data based on linear context i.e the prediction of words is done on a small window of linear context. A simple optimization technique is used to simplify the computation by using negative sampling i.e to maximize the probability of word and context from the same data D, and minimizing the probability of word and context not from the same data D.

The probability of target context pair (t, c), being observed in the data is given by :

$$P(D = 1|t, c) = \sigma(v_t.v_c) \tag{2.11}$$

where v_t and v_c are the target and context word embeddings, and σ is the sigmoid function. For a negative sampled pair (t, c) not being observed in the data is given by:

$$P(D = 0|t, c) = 1 - \sigma(v_t.v_c) \tag{2.12}$$

The objective function of the model becomes:

$$\arg\max_{v_t, v_c} \sum_{t,c \epsilon D} log\sigma(v_t.v_c) + \sum_{t,c \epsilon D'} log\sigma(-v_t.v_c) \tag{2.13}$$

The model with one hidden layer learns two sets of weights for each word: one for embedding the words to a low dimensional space in the hidden layer weights, it is referred as embedding layer weights, and the other in the projection layer referred as projection layer weights. The resulting word vectors have shown to capture linguistic regularities like

$$\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} = \overrightarrow{queen}$$

This embedding method is considered as the state-of-the-art and various deep learning methods have used the pretrained word embeddings as input for various NLP tasks. In our work, we use a variant of this model to encode the representation of word and relation in the same vector space.

Chapter 3

RELATED WORK

3.1 Word and relation embedding

There have been many methods of deriving word embeddings in the NLP community. Most recently, many neural-network based language modelling methods have been proposed to generate dense representations of words (Bengio *et al.* (2003), Mikolov *et al.* (2013a)). These methods of word embeddings have shown to capture semantic and syntactic properties (Turney and Pantel (2010). Most common and considered the state of the art are the Skipgram and CBOW based word2vec model Mikolov *et al.* (2013b) and Glove model by Pennington *et al.* (2014) which performs training based on word co-occurrence statistics from a corpus. Both of these models have been explained in section 2.2

There also have been various other works, which are variant of the word2vec based model. Dependency based word embedding by Levy and Goldberg (2014), is one such work which learns the embedding of words not with a linear context, but using the syntactic contexts that are derived from produced dependency parse tree of a sentence. The embedding of the words capture more functional and less topical similarity and showed to perform much better on the WordSim-353 dataset Finkelstein *et al.* (2001). Another work to capture word embedding based on the syntactic structure of the sentence called the C-PHRASE model Pham *et al.* (2015), which uses a CBOW based word2vec model. For example (taken from the same paper), given the sentence: "A sad dog is howling in the park", C-PHRASE model optimizes the context prediction for: dog, sad dog, a sad dog, a sad dog is howling but not for: "howling in", as these two words do not form a syntactic constituent. They showed that training the words using the syntactic structure performs better on various lexical tasks as well as many sentential tasks like sentiment classification. In our approach, we have also adopted similar methods discussed to train the word embedding based on the syntactic structure of the sentence.

Various approaches to learn the representation of relations have also been investigated and used in knowledge bases Bordes *et al.* (2011), Neelakantan *et al.* (2015), Lin *et al.* (2015), relation classification Liu *et al.* (2015), aspect term extraction Yin *et al.* (2016) and dependency parsing Bansal (2015). As per the work done by Bordes *et al.* (2011), in which they learn the representation of both the relation and entities. The relation is modelled as a translation vector that connects the vectors of two entities. Rather than using raw textual data to learn the representation, it optimizes an objective over all the facts in a knowledge base. These embeddings are then used for fact extraction in knowledge bases and also for reasoning missing facts.

Bansal (2015) proposed a method to learn dependency link embedding using the skipgram based model. The dependency links between words in a sentence were extracted using the MST parser McDonald *et al.* (2005). They used these learnt link embeddings as features and reported strong accuracy for dependency parsing experiments and constituent parsing re-ranking task. In this work, the representation of the links are learnt using the context of words they link but as a concatenated unit of parent(p) and child(c) i.e. p_c and also their grandparent dependency relation. They serialize a tuple in the following manner :

"$d_{<D>}\ gl_{<GL>}\ p\_c\ l_{<L>}\ d_{<D>}$"

Where l is the relation label, gl is the grand parent relation label, p_c is the parent and child word as a concatenated unit and signed bin distance d. The model is learnt using the skipgram model with window-size of 2 and dimension size of 100.

Yin *et al.* (2016), proposed a method to jointly learn the word and relation representation by parsing huge amount of raw textual data using stanford dependency parse and extracting Chen and Manning (2014) all triples (w_1, w_2, r), where w_1 and w_2 denote two words and r is the dependency link between them. Their method optimized the objective function w1 + r = w2, to learn the representation of words and relation, then enhanced the word embeddings by leveraging the linear context in a multi-task learning manner. They reported state-of-the-art performance on aspect term extraction tasks using semeval dataset. Their best model was using a composition model of word and relations embedding. They also reported better results than the dependency based embedding Levy and Goldberg (2014) and skipgram based embeddings Mikolov *et al.* (2013b).

3.2 Sentence representation using syntactic information

The above mentioned approaches to learn the representation of words have also been applied to learn the representation of phrases and sentences. Le and Mikolov (2014) introduced paragraph vector, which was based on the above mentioned skip-gram model to learn representations of sentences, paragraphs, and documents. In this work a sentence or a paragraph is represented by a vector and trained to predict the words in the document. They showed performance better than the bag of words approach earlier adopted for various tasks. There also have been methods to create an internal representation of a sentence directly using deep learning methods like Recurrent Neural Networks and Convolutional Neural Networks as discussed in 2.1. These models have shown to be highly successful for various NLP task. But in these mentioned approaches, linguistic or syntactic information for sentence representation was largely ignored.

There have been prior research using deep learning methods to represent sentences using these linguistic or syntactic information. Recursive neural network(RNN), a model proposed by Pollack (1990) and popularized in NLP with a series of work done by Socher *et al.* (2010), Socher *et al.* (2011),Socher *et al.* (2012) have been highly successful and shown better performance in various sentence classification tasks like sentiment analysis. The RNN model takes a sentence, syntactic tree and the vector representations for the words in the sentence as input, and applies neural network to compute recursively, the vector representation for all phrases in a tree to a sentence representation in the final step.

Ma *et al.* (2015), in his work proposed a neural network architecture called the Dependency based Convolutional neural network (DCNNs), to use the linguistic information in the Convolutional Neural Network. The modifications was brought to the sequence of the input in the sentence. While the other CNN models Kim (2014), put word in its sequential context, this model considered word and its parent, grand-parent, great-great grand parent, and siblings using a dependency tree. With this approach the long distance information in the sentence was preserved. They demonstrated superior performance over the previous CNN based methods. In our work, we do not use the syntactic structure of the sentence but the relation information between the words. Thus our work is parallel to these previous work and could be applied to them.

Komninos and Manandhar (2016), in his work proposed a method jointly learn the embeddings of word and dependency context and use the embedding of words and also the dependency context to form a sentence representation, which is one similar work to ours. Every sentence is first parsed using the Stanford parser Chen and Manning (2014), and then the embeddings of word and dependency contexts are trained. For example

Sentence: John loves Mia

Dependency structure: $nsubj(loves, John); dobj(loves, Mia)$

Dependency context of word *loves*: $nsubj\_John$, $dobj\_Mia$

Dependency context is defined as dependency (d) concatenated with the word (w) as d_w. In this work, the authors show the effectiveness of the composition model for enriching sentence representation using the dependency context. They showed that using the extra syntactic information i.e dependency context embedding improved performance on various sentence classification tasks. In our work we explore the joint learning of the embeddings of words and relations from the Stanford dependency parser $(nsubj, dobj)$. Joint learning of these fundamental units of a structured representations offers more flexibility in terms of its usages in different composition methods for sentence representation. In our work, we also explored some of these composition methods proposed in the previous work by Komninos and Manandhar (2016) and also experimented with some new composition methods. The proposed method of learning embeddings of words and relations is general and can applied to learn the same for outputs of different semantic parsers such as K-Parser (Sharma *et al.* (2015), Scene Graph Parser Schuster *et al.* (2015)).

Chapter 4

METHODOLOGY AND IMPLEMENTATION

In this chapter, we have provided with a brief overview of the various methodology adopted and with their implementation details. This work can be seen as a three step process: model to implement the embedding of the words and relations, representing sentence using these word and relation embeddings and developing various classification methods for performing sentence classification. The overview can be seen as in figure 4.1. Firstly, raw text from Wikipedia is parsed using the Stanford parser Chen and Manning (2014), with this parsed output the context of word and relations are extracted. With the extracted context the embeddings of word and relations are trained as described in section 4.1. These embeddings can then be used with various composition models for sentence representation as described in section 4.2. The sentence representation is then used with the various classification models described in section 4.3 to perform classification.

4.1 Embedding Models

Our approach to learn the word embeddings is built on top of the SkipGram based model as explained in 2.2. In this section we discuss two methods based on Skipgram model one for learning only the word embedding based on linear context, which serves as our baseline and other for jointly learning the word and relation embedding based on output of the dependency graph. In the word and relation model we also explain the modification made to the Skipgram model to jointly learn word and realtion embeddings.

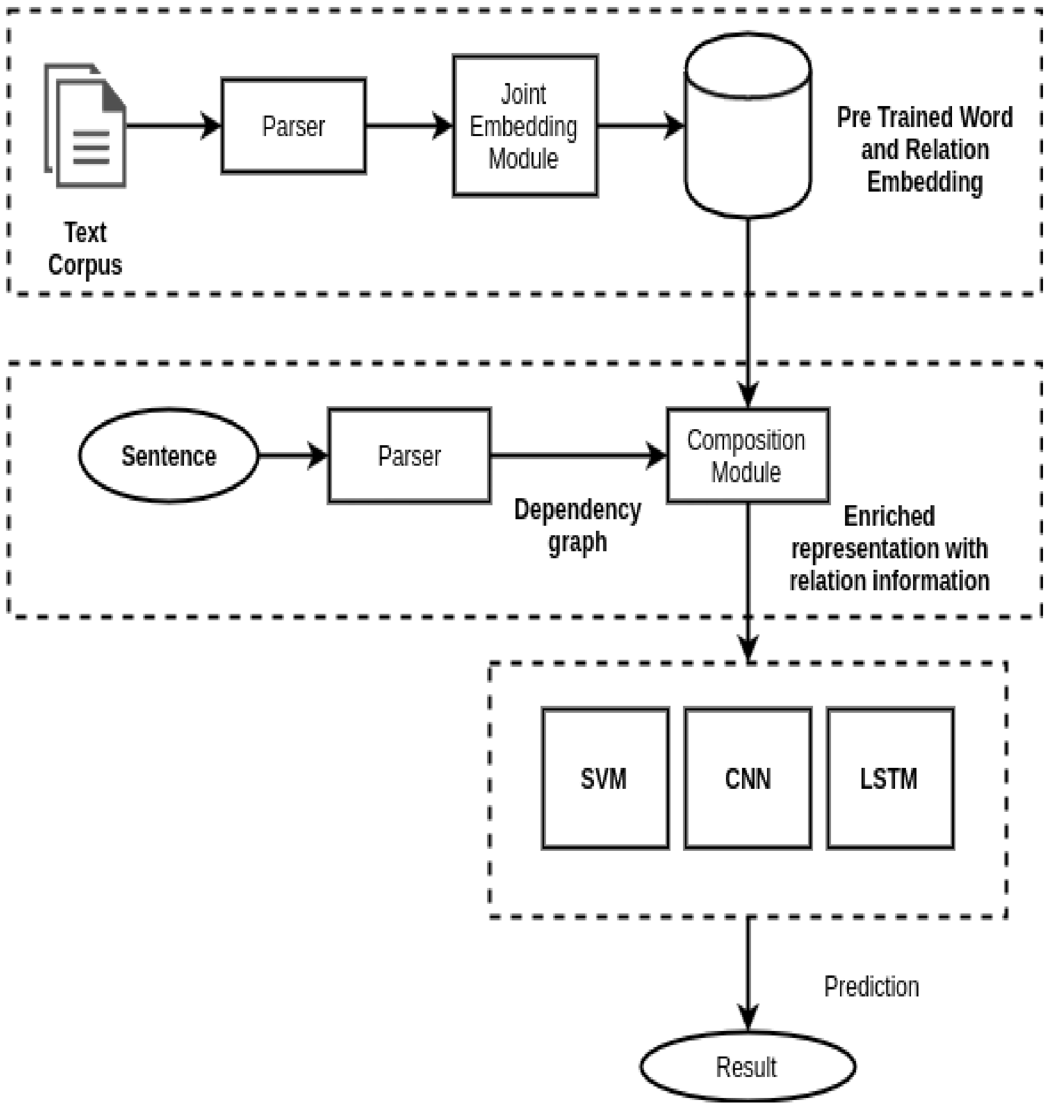

Figure 4.1: Methodology Overview

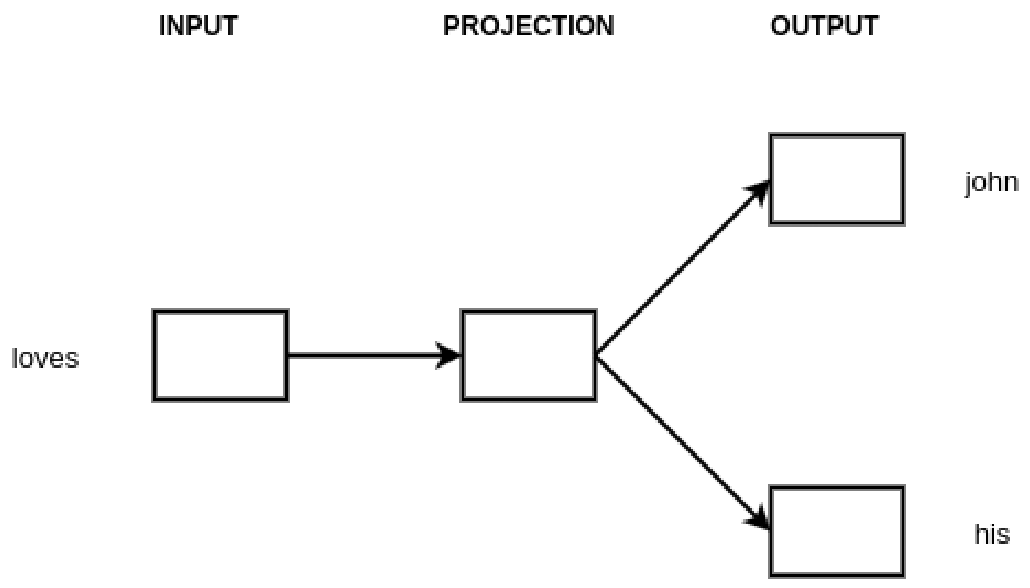

Figure 4.2: Window based Skipgram Model

Window based Skipgram model (Win5)

This is the standard skipgram model that considers target-context word pairs inside a windows of 5 words to the right and to the left of the target word. The window size for every target instance in the corpus is uniformly sampled from [1,5], range , effectively providing a weighting scheme for context words according to their distance from the target word. A negative sampling of 15 words is considered while training. The sentence representation using these word embeddings are considered as baseline for sentence classification evaluation.

An example of training context for this model if window of 1 is considered also described in Figure 4.2:

Sentence: John loves his wife.

Context of “loves”: John, his

Context of “his”: loves, wife

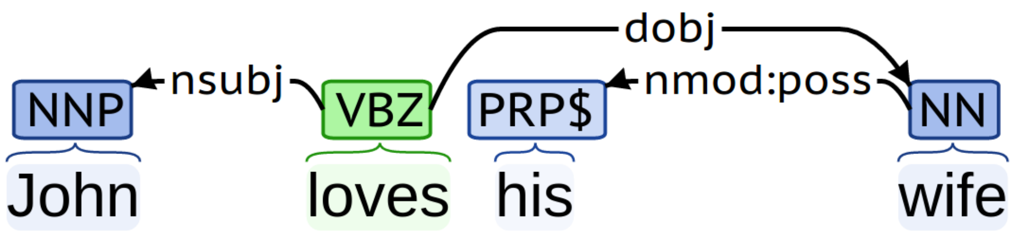

Figure 4.3: Dependency graph

Word and relation model (WR)

We train our word and relation embedding based on their contexts. We take all edges of the dependency tree to form logical triplets of the format (w1 r w2) e.g. (cup of:nmod coffee). We train the word embeddings similar to as done in the Win5 but the context of the target words are the words which are one or two hop distance away in the dependency tree. The relations are trained with the context of words it links i.e. for r the context is w1 and w2. The negative sampling parameter for words and relations are also chosen differently. An example of training context for this model is considered using the parse of the sentence as given in Figure 4.3. The context is also described in Figure 4.4 .

Sentence: John loves his wife.

Context of “loves”: John, wife

Context of “his”: wife

Context of “nsubj”: John, loves

Context of “dobj”: loves, wife

Corpus

A lot of previous embedding skipgram models (Levy and Goldberg (2014), Komninos and Manandhar (2016), have performed the training of word and sentence evaluations

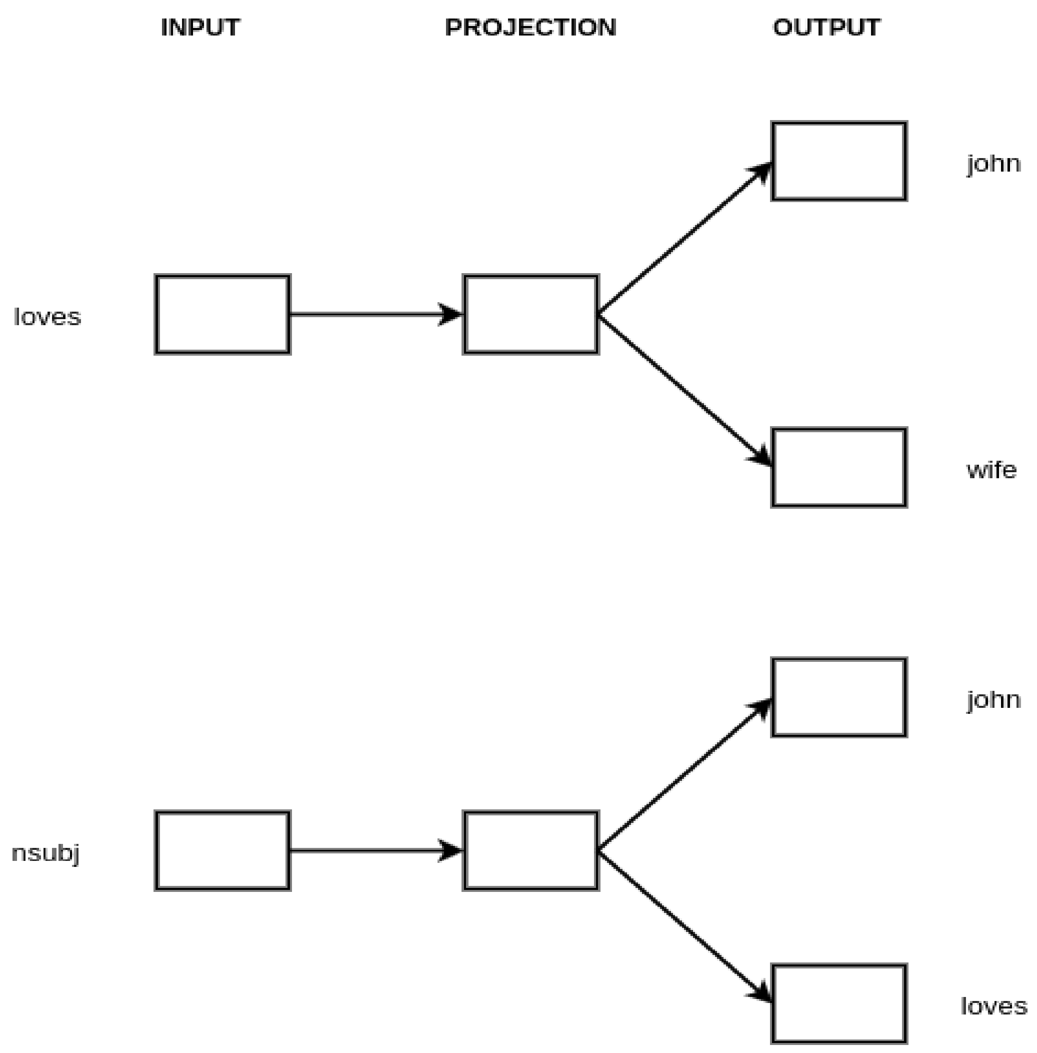

Figure 4.4: Word and Relation Skipgram Model

using the English Wikipedia dumps. For our model, we also performed the training on the first 500 million words of English Wikipedia 2016 dump. We removed the words that appeared less than 100 times in the corpus.

Implementation Details

We trained 300 dimensional vectors using skipgram variants on the above mentioned dataset 4.1. Preprocessing was done to remove words appearing less than 50 times.

Training was done by applying negative sampling with 15 for words and 20 for relations. 10 iterations over the entire corpus was done using stochastic gradient descent method. We applied the following commonly used methods during training: negative samples were drawn according to their unigram distribution raised to the power of $\frac{3}{4}$, linear decay of learning rate with initial $\alpha = 0.25$, and subsampling of target words with probability given by $p = \frac{f-10^{-5}}{f} - \sqrt{\frac{10^{-5}}{t}}$, where f is the words frequency. Dependency parsing for WR was done with the Stanford Neural Network dependency parserChen and Manning (2014) using Enhanced Universal Dependency tags Schuster and Manning (2016).

For implementation of this model, we used the word2vec [1] software as the base code to implement the basic skipgram. We built the arbitrary context and arbitrary negative sampling functionality on top of it.

4.2 Sentence feature representation

We create different sentence representation for three different embeddings Win5 and WR. Vw is the vector of word w trained for using Win5 and WR. Vr is the vector of relation r trained using the WR method.

Win5: Every word is represented as the vector of the Vw trained by the Win5 based method

$$X = V_w \tag{4.1}$$

WR Words: Every word is represented as the vector of the V_w trained by our method

$$X = V_w \tag{4.2}$$

WR WavgR : Every word is represented as the weighted average of word and its

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

relations

$$X = \frac{1}{2}\, V_w + \frac{1}{2k} \sum_{i=1}^{k} V_{ri} \tag{4.3}$$

i.e. For example given in Figure 4.3, for the sentence "John loves his wife", the representation of "John" is found formed using the above equation as:

$$X_{John} = \frac{1}{2}\, \overrightarrow{John} + \frac{1}{2}\overrightarrow{nsubj}$$

WR WavgWR : Every word is represented as the weighted average of word and dependency context

$$X = \frac{1}{2}\, V_w + \frac{1}{4k} \sum_{i=1}^{k} (V_{ri} + V_{wi}) \tag{4.4}$$

i.e. For example given in Figure 4.3, for the sentence "John loves his wife", the representation of "John" is found formed using the above equation as:

$$X_{John} = \frac{1}{2}\, \overrightarrow{John} + \frac{1}{4}(\overrightarrow{nsubj} + \overrightarrow{loves})$$

WR WConc: Every word is represented as the concatenated vector of word and dependency context, forming a 600 dimensional vector

$$X = V_w \oplus \frac{1}{2k}\, \sum_{i=1}^{k} (V_{ri} + V_{wi}) \tag{4.5}$$

i.e. For example given in Figure 4.3, for the sentence "John loves his wife", the representation of "John" is found formed using the above equation as :

$$X_{John} = \overrightarrow{john} \oplus \frac{1}{2}(\overrightarrow{nsubj} + \overrightarrow{loves})$$

WR Triplet: In this section rather than representing every word in sentence, we represent every triplet from the dependency graph of the sentence. Every triplet i.e.

(word, relation, word) from the stanford parse of a sentence is concatenated creating a 900 dimension vector. For example given in Figure 4.3, for the sentence "John loves his wife", this could also be represented in the triplet format as follows:
(loves, nsubj, John)
(wife, nmod:poss, his)
(loves, dobj, wife)

Every triplet is hence represented as per the below equation:

$$X = V_{w1} \oplus V_r \oplus V_{w2} \tag{4.6}$$

For example (loves, nsubj, John) is represented as :

$$X = \overrightarrow{loves} \oplus \overrightarrow{nsubj} \oplus \overrightarrow{John}$$

For all the above models except the triplet representation (WR Triplet) all word representations (X) are used as a sequence of embeddings respecting the order of the sentence to become the input for the CNN and LSTM. For the SVM BoE, all the vectors of the words in the sentence are averaged. During our evaluation of embeddings, we did not perform any updates during training of CNNs and LSTMs.

4.3 Classification Methods

SVM with averaged embeddings

We create a sentence representation by averaging embeddings of sentence features (words and dependency contexts). This can be considered the equivalent of a Bag-of-Words sentence representation in the embedding space, hence called Bag-of-Embeddings (BoE). We then train a classifier by applying a Support Vector Machine with a Gaussian kernel:

$$K(x, x') = exp(-\gamma \|x - x'\|^2) \tag{4.7}$$

Sentence is parsed and all its relations are extracted. Using the process described in section 4.2, it is transformed with a sequence of $\overrightarrow{X}$, of words or triplets. The below equation describes the input vector of the sentence:

$$\overrightarrow{Sentence} = \frac{1}{n} \sum_{i=1}^{n} (\overrightarrow{X_i}) \tag{4.8}$$

Where n is the number of X's for the sentence.

Implementaion and Parameters: For hyperparameter tuning, we set parameter γ of the kernel to 1/k, where k is the number of features (dimensionality of embeddings), and then perform cross validation for the c parameter using the standard Win5 word embeddings in the question classification task. The SVM model was implemented in python using scikit-learn library Pedregosa *et al.* (2011). The figure 4.5, describes the above process.

Convolutional Neural Network

We have used the simple Convolutional Neural Network of Kim (2014) that has shown to perform well in multiple sentence classification tasks. The input to the network is a sentence matrix X formed by the concatenating k-dimensional word embeddings. Then convolutional filters $W \in R^{h \times k}$ is applied to every possible sequence of length h to get a feature map:

$$C_i = tanh(W.X + b) \tag{4.9}$$

The output of the filters is followed by a max pooling operation to get the feature

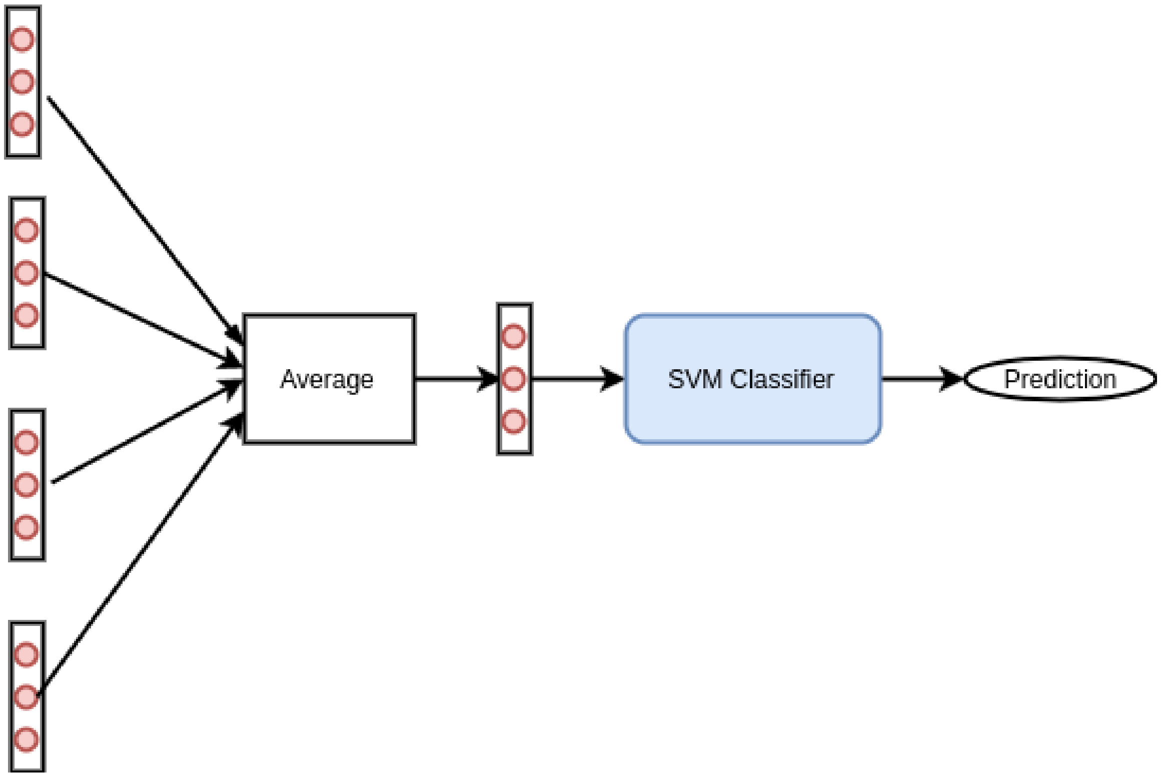

Figure 4.5: SVM sentence classification

with the highest value:

$$\hat{c} = max\mathbf{c_i} \tag{4.10}$$

Where c is the output from all the filters.

Sentence is parsed and all its relations are extracted. Using the process described in section 4.2, it is transformed with a sequence of $\overrightarrow{X}$, of words or triplets. The sequence of X's is concatenated to form a $N \times K$ matrix. The below equation describes the input vector of the sentence:

$$\overrightarrow{Sentence} = \overrightarrow{X_1} \oplus \overrightarrow{X_2} \oplus \overrightarrow{X_3} \oplus ... \oplus \overrightarrow{X_N} \tag{4.11}$$

Where n is the number of X's for the sentence.

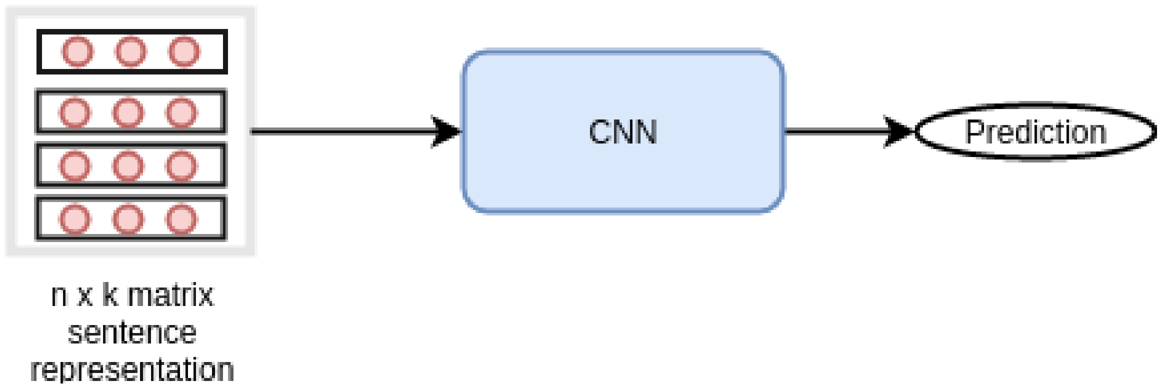

Figure 4.6: CNN sentence classification

Implementation and Parameter: The network uses multiple filters with different sequence sizes covering different size of windows in the sentence. All hyper-parameters of the network are the same as used in the original paper Kim (2014): stochastic dropout Srivastava *et al.* (2014) with p = 0.5 on the last layer, 100 filters for each filter region with filter regions of width 2,3 and 4. We perform pooling operation, the pooled features of different filters are then concatenated and passed to a fully connected softmax layer to perform the classification. We performed the optimization with Adam (Kingma and Ba (2014)) on minibatches of size 50. The CNN model was implemented in python using Theano Theano Development Team (2016) library on GPU cluster. The figure 4.6, describes the above process.

Long Short Term Memory(LSTM)

LSTM networks Hochreiter and Schmidhuber (1997) are recurrent neural networks where recurrent units consist of a memory cell c and three gates i, o and f. A sequence of input embeddings x, LSTM outputs a sequence of states h for each time step. The distribution of labels for the whole sentence is computed by a fully connected softmax layer on top of the final hidden state. Sentence is parsed and all its relations are extracted. Using the process described in section 4.2, it is transformed with a

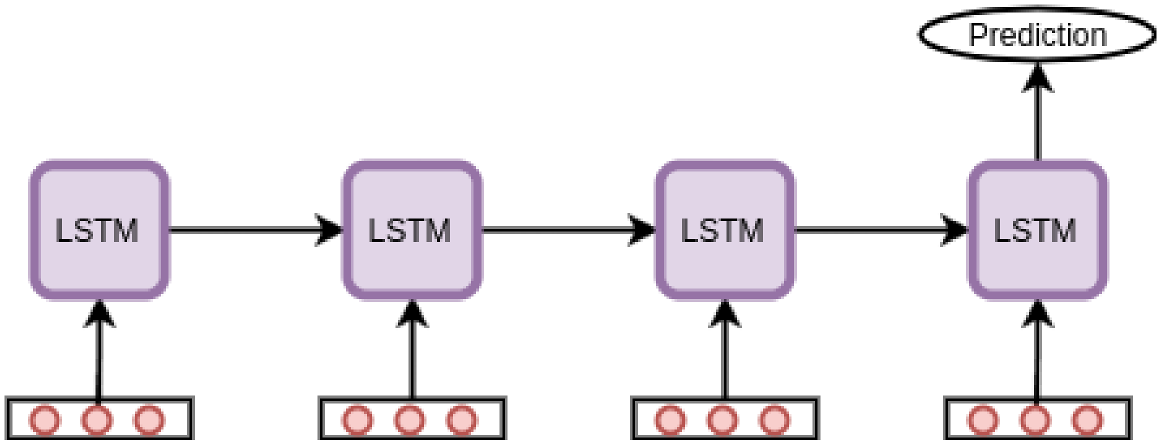

Figure 4.7: RNN sentence classification

sequence of $\overrightarrow{X}$, of words or triplets. The below equation describes the input vector of the sentence as a sequence to RNN:

$$\overrightarrow{Sentence} =< \overrightarrow{X_1}, \overrightarrow{X_2}, \overrightarrow{X_3}, ..., \overrightarrow{X_N} > \tag{4.12}$$

Where n is the number of X's for the sentence.

Implementation and Parameter: The parameters used in the network are: stochastic dropout with p = 0.25; 150 dimensions for the size of hidden layer h; Optimization is performed using Adagrad Duchi *et al.* (2011) and mini-batch size of 100. The RNN model was implemented in python using Theano Theano Development Team (2016) library on GPU cluster. The figure 4.7, describes the above process.

4.4 Implementation Challenges

At first, the implementation for training of word and relation embedding was done using NumPy Jones *et al.* (01), which is a fundamental package for scientific computing with Python. Numpy offers a fast implementation for various scientific operations.

However, due the limitation of global interpreter lock (GIL), which prevents multiple native threads from executing Python bytecodes at once, we could not achieve multithreading in Python and the rate of training of words was 20x slower than the original C code. We had to reimplement the desired functionalities of training word and relation embedding with arbitrary context and arbitrary negative sampling, with modifications made to the original word2vec software. With the C based implementation, the rate of training became 27,000 words/thread, which was 20x faster than the Python implementation. This was run on a system with 2.3GHz CPU on Linux OS with 8 cores. The total time to perform preprocessing and training on the Wikipedia corpus described in section 4.1, was about 14 hours.

One of the other challenges was to parse the complete corpus using Stanford parser. We first parsed the complete corpus to individual sentences using NLTK's Bird (2006) punct sentence tokenizer. We then split the complete corpus into 2 parts to run on two different systems using the Java based Stanford core NLP Manning *et al.* (2014) package. We parsed all the lines in the file using a total of 12 CPU cores, which took a total of 2 weeks to parse the complete corpus.

Our sentence classification evaluations was performed on: Two Neural Network Architectures i.e. LSTM and CNN; Three sentence classification datasets; and with 6 different variations of embeddings or composition based on embeddings. For each we performed 20 iterations to report the best result. Thus the total number of neural network iterations were $2 \times 3 \times 6 \times 20 = 720$. The total time to run each iteration on GPU was on an average of 30 minutes and the total evaluation took about 14 days of running time.

Chapter 5

EXPERIMENTS AND RESULTS

We perform two kinds of evaluations one for evaluating the quality of the word embedding produced and the other for evaluating our sentence representation for sentence classification tasks. In addition to this, we also performed evaluation for sentence pairs on textual entailment task.

5.1 Word similarity

This is one of the classic lexical tasks, different embedding methods are required to quantify the degree of similarity of relatedness of word pairs in terms of the cosine similarity between the corresponding word vectors. Performance was calculated using the Spearman correlation between the model and the human score. Higher the correlation, better is the performance. Below is the formula for calculating Spearman correlation:

$$\rho = 1 - \frac{6\sum d^2}{n(n^2-1)}$$

Where d is the difference between cosine similarity of words to the human score. n is the number of word pairs

In our case, We performed evaluation to compare the quality of word embeddings produced from jointly learning word and relation representation to the word embeddings produced by the skipgram model. The evaluation was performed using two word similarity datasets namely: WordSim-353 Finkelstein *et al.* (2001) and SimLex-999 Hill *et al.* (2016). The two datasets use a different notion of word similarity. Wordsim-353 mostly captures topical similarity (or relatedness), giving high similar-

Word1	**Word2**	**Human Score**
tiger	cat	7.35
book	paper	7.46
computer	keyboard	7.62
computer	internet	7.58
plane	car	5.77

Table 5.1: WordSim353 dataset

Word1	**Word2**	**Human Score**
old	new	1.58
smart	intelligent	9.2
hard	difficult	8.77
happy	cheerful	9.55
hard	easy	0.95
fast	rapid	8.75

Table 5.2: SimLex-999 dataset

ity to pair of words like cup-tea. SimLex-999 uses a more strict version of similarity, often called substitutional similarity, where the pair clothes-closet has a low similarity score and pairs like cup-glass have high similarity. Few examples of word pairs and their human scores for SimLex-999 and WordSim-353 are provided in table 5.2 and 5.1 respectively. The dataset for SimLex-999 and WordSim-353 can be found here [1] and [2] respectively.

Our model WR skipgram version achieves a higher correlation for WordSim-353

[1]www.cl.cam.ac.uk/ fh295/simlex.html

[2]www.cs.technion.ac.il/ gabr/resources/data/wordsim353/wordsim353.html

Embedding	**WordSim-353**	**SimLex-999**
Win5	0.627	**0.352**
WR	**0.664**	0.329

Table 5.3: Spearman correlation on WordSim-353 and SimLex-999 word similarity evaluation tasks.

compared to Win5, but the results are reversed for SimLex-999. This is also observed in the case of model trained on dependency context by Levy and Goldberg (2014). While similarity based evaluation makes obvious that different contextual features capture different properties of words, it is not clear which kind similarity notion is more useful when word representations are used as features for NLP tasks. For various tasks like sentence classification using word embeddings, this evaluation has not shown much correlation.

5.2 Sentence Classification

5.2.1 TREC Question Classification

The TREC Question Classification dataset Li and Roth (2002), which consists of 5452 training questions and 500 test questions. The task is to classify each question with one of six classes {location, definition, abbreviation, entity, numeric} depending on the answer they seek in the sentence. The average length of the sentence is 10 words. Some examples of this dataset is provided in table 5.4. The dataset could be found here [3] . For CNNs and LSTMs 10% of the training data were used as the validation set to select the best model among different iterations. Classification accuracy results for each input representations and classification method has been

[3] http://cogcomp.cs.illinois.edu/Data/QA/QC/

Sentence	**Label**
When was Ozzy Osbourne born	NUM
Which two states enclose Chesapeake Bay	LOC
Name a golf course in Myrtle Beach	ENTY
How did serfdom develop in and then leave Russia	DESC
What is the full form of .com	ABBR

Table 5.4: TREC dataset examples

Embedding	**SVM**	**CNN**	**LSTM**
Win5	88.60	93.4	94.2
WR Words	87.40	93.2	93.8
WR WavgR	**89.00**	92.2	92.2
WR WavgWR	84.80	94.0	95.0
WR WConc	87.80	95.0	94.6
WR Triplet	87.40	**96.2**	**95.2**

Table 5.5: TREC dataset results

reported in 5.5. The state of the art result has been achieved by using the dependency based convolutional neural network of Mou *et al.* (2015). Their model consists of a convolutional neural network that takes a dependency tree at the input layer after parsing the sentence instead of a sequence as in the LSTM based model.

5.2.2 SST-2 Sentiment Classification

The Stanford Sentiment Treebank dataset Socher *et al.* (2013) provides with fine grained sentiment polarity scores for movie reviews on the phrasal and sentence level. In our evaluation we evaluate on the binary version of the task, which con-

Sentence	**Label**
this is one of polanski's best films	POS
even as lame horror flicks go, this is lame	NEG
as a singular character study, it's perfect	POS
a turgid little history lesson, humourless and dull	NEG
the cast is uniformly excellent and relaxed	POS

Table 5.6: SST-2 dataset examples

siders only positive and negative sentiment classes. The dataset is provided with the split required for training, validation and testing i.e. 6920/872/1821 split for training/validation/testing sets. Some examples of this dataset is provided in table 5.6. We performed our evaluations on sentence level annotations. Classification accuracy results for each input representations and classification method has been reported in 5.7. The dataset could be found here [4] .

The state of the art for this dataset has reported to achieve 88% accuracy from Kim (2014), which uses the same Convolutional neural network as we have used for our evaluations. But the best model utilizes the phrasal level annotations and not sentence level annotation. Also in this specific configuration, the network (multichannel) uses two channels at the input layer, one updating the word embeddings during training and one that keeps them static as we do in our experiments. All our model for this dataset uses static versions i.e. the embeddings are not updated during the training.

[4]https://nlp.stanford.edu/sentiment/treebank.html

Embedding	**SVM**	**CNN**	**LSTM**
Win5	**77.42**	77.92	81.60
WR Words	76.77	**83.96**	**82.70**
WR WavgR	76.16	77.15	75.50
WR WavgWR	74.68	78.52	76.22
WR WConc	75.34	79.60	77.26
WR Triplet	76.83	80.25	79.88

Table 5.7: SST-2 dataset results

5.2.3 Subjectivity Classification

In subjectivity classification, the task is to classify a sentence as being subjective or objective Pang and Lee (2004). A subjective sentence is a statement which expresses an opinion or emotion and not necessarily a fact, whereas a objective sentence presents a perspective based on facts. The subjective statements are taken from reviews of rottentomatoes.com and the objective statements are taken from plots of movies from IMDB. The sentences are movie review snippets of length of atleast 10 words. The dataset contains, 5000 subjective and 5000 objective sentences. Some examples of this dataset are provided in table 5.9. We performed our evaluations on sentence level annotations. The dataset could be found here [5] . Classification accuracy results for each input representations and classification method has been reported in 5.8 We performed 10 cross validation on the dataset containing total of 10000 sentences. The best accuracy reported on this dataset comes from the Dependency sensitive Convolutional Neural Network model by Zhang *et al.* (2016), at 93.9 %. This model uses a combination of Long Short Term Memory networks and Convolutional neural

[5]www.cs.cornell.edu/people/pabo/movie-review-data/

Sentence	**Label**
when it could have been so much more	SUBJ
a work that lacks both a purpose and a strong pulse	SUBJ
a movie that doesn't aim too high , but doesn't need to	SUBJ
television made him famous , but his biggest hits happened off screen	OBJ
his father and brother are dead for so many years	OBJ

Table 5.8: Subjectivity dataset examples

Embedding	**SVM**	**CNN**	**LSTM**
Win5	**91.60**	91.70	91.45
WR Words	90.90	**92.15**	**93.05**
WR WavgR	90.15	90.95	90.35
WR WavgWR	89.15	90.20	90.15
WR WConc	89.60	89.95	91.30
WR Triplet	90.30	92.00	92.90

Table 5.9: Subjectivity dataset results

network to build the sentence representation hierarchically.

5.3 Sentence Entailment

In this entailment task (Marelli *et al.* (2014)), given two sentences A and B, model has to classify into three classes: (*i*) Entailment (A entails B), (*ii*) Contradiction (A contradicts B) and (*iii*) Neutral (A neither contradicts nor entails B). The dataset [6] is provided with the split required for training, validation and testing i.e. 4439/495/4906

[6]http://alt.qcri.org/semeval2014/task1/

Sentence	**Label**
A: Two teams are competing in a football match B: Two groups of people are playing football	Entailment
A: The brown horse is near a red barrel at the rodeo B: The brown horse is far from a red barrel at the rodeo	Contradiction
A: A man in a black jacket is doing tricks on a motorbike B: A person is riding the bicycle on one wheel	Neutral

Table 5.10: Sentence entailment dataset examples

Embedding	**CNN**
Win5	64.6
WR Words	65.1
WR Triplet	**67.2**

Table 5.11: Sentence entailment results

respectively. Some examples of this dataset is provided in table 5.10. For this task we used the the same CNN as described in section 4.3, with the sentence feature representation for each sentence pair concatenated to form the 2D input matrix. We have performed comparison with only our previous best performing composition technique of words and relations i.e. WR Triplet. The results have been shown in table 5.11.

The best accuracy reported on this dataset comes from the Attention Based CNN model by Yin *et al.* (2015), at 86.2 %. In their work, they have used a CNN which models the sentence representation based on the mutual influence between sentences.

5.4 Sentence representation methods and Classification benchmarks

In this section, we have described the commonly used sentence represenation learning methods. We have reported the best results of these representation learning methods on the three classification tasks in table 5.12. We also report the results from our method i.e. Word and Relation embedding.

NB-SVM Wang and Manning (2012): In this method a Naive Bayes SVM is applied with unigram and bigram as features for representing the sentence.

MNB: In this method a Multinomial Naive Bayes is applied with unigram and bigram as features for representing the sentence.

CBoW: In this method an average of the word vectors of the sentence is performed to get the fixed length sentence vector. The word vectors are taken from the Skipgram based word2vec model.

BRNN: Schuster and Paliwal (1997): Bidirectional recurrent neural networks is a work which uses a variant of recurrent neural network. In this network it connects two hidden layers of opposite directions to the same output. By this structure, the output layer gets information from past and future words.

CNN Kim (2014): Convolutional neural network for sentence modeling. This is the same neural network which we have used in our method for sentence representation as described in 4.3

AdaSent Zhao *et al.* (2015): In this work, sentence representation is learnt by forming a hierarchy of representations from words to phrases through recursive gated local composition of adjacent segments.

Paragraph Vector Le and Mikolov (2014): This is an unsupervised model to learn distributed representations of words and paragraphs. This method learns the representation of each document(sentence or paragraph) by a dense vector which is trained

Method	**TREC**	**SST-2**	**SUBJ**
NB-SVM	-	79.4	93.2
MNB	-	91.7	91.4
CBoW	87.3	77.2	91.3
RNN	90.2	82.3	**93.7**
CNN	93.6	81.5	93.4
AdaSent	92.4	**83.1**	93.5
Paragraph Vector	91.8	74.8	90.5
Skip-Thought	92.2	76.5	93.6
Word & Relation	**96.2**	80.2	92.9

Table 5.12: Classification accuracy of various sentence representation learning methods

to predict words in the document.

Skip-Thought Vector Kiros *et al.* (2015): This is an unsupervised model to learn distributed representations of sentences. The sentence representation is learnt using a RNN based encoder-decoder network. The encoder-decoder network is trained to reconstruct the surrounding sentences given a target sentence.

5.5 Result Analysis

Overall our evaluation shows that the sentence representation using our model of trained word and relation embeddings outperformed the baseline skipgram based model(Win5). Using the sentence representation composed of both word and relation embeddings, TREC question classification task showed great improvements across the three classification methods. For this dataset we observe that, the 600 dimensional embedding(WR Conc) and the 900 dimensional embedding(WR Triplet) performed consistently well across classification methods. Our 900 dimensional composition method(WR Triplet) also achieves state-of-the-art with 96.2% accuracy on this dataset.

For the other two tasks of Sentiment classification and Subjectivity classification, the composition based method worked better than the skipgram based word representation (Win5) model but showed decrease in performance than the word based representation (WR Words). There has been prior research Liu *et al.* (2015) of using syntactic information using the tree structured networks for the SST dataset, which also confirmed that syntax does not provide much improvement on this dataset. For subjectivity classification, we observe similar behavior that adding syntactic information does not improve the performance. For the above two tasks we observed that, using only the word embedding based sentence representation, our representation (WR Words) greatly outperformed the window based skipgram (Win5) in the neural network based classification methods (LSTM, CNN) and comparable results in the SVM based classification method.

From the various composition methods for sentence representation, we observe that triplet based composition model (WR Triplet) shows the best performance for all classification tasks across classification methods. Hence this could be a general composition method of sentence representation for providing syntactic information for several classification tasks. We also observe that composition method with weighted average of word and relation (WR WavgR) performs better for SVM model but performance does not show much improvement for the neural network based methods (CNN, LSTM). The composition method (WR Conc) with concatenation of word and it's dependency context i.e. relation and word has comparable performance as the WR triplet method. This shows that increasing the feature space with syntactic information has better performance than adding it to the same feature space.

For the task of sentence entailment, the composition based method WR Triplet showed the best performance as compared to the two word based sentence representations (Win5 and WR Words). The evaluation on this dataset provides us further evidence to use both word and relation embeddings for modeling sentence representation.

Chapter 6

CONCLUSION AND FUTURE WORKS

6.1 Summary

We presented a methodology to jointly learn the dense representation of words and relations. We presented various composition models for enriching the sentence representation using both the word and relation embeddings. We performed comparison of this enriched sentence representation to the sentence representation with only the word embedding using two sentence classification tasks, namely Question Category classification and, Sentiment Analysis. The sentence classification was done using the three state-of-the-art supervised classification techniques (SVM Cortes and Vapnik (1995), CNN Kim (2014) and LSTM Dyer *et al.* (2015)). For the task of Question Classification, we observe a noticeable improvement in prediction accuracy for most cases using the composite sentence embeddings. When used with traditional neural network architectures, the triplet-based composition model outperforms both the original skip-gram based word-embedding and our relation-enhanced word-embeddings. For the Sentiment Analysis task, we observe that using only the word embeddings gives higher accuracy than the composition models of word and relation. But the word embeddings which are jointly trained with relations, outperforms the original skip-gram based word embeddings. The triplet based composition model is still the second best performing model for this dataset.

6.2 Future Work

In our work, for the purpose of comparison with different embedding and composition techniques, little hyperparameter tuning was done on the three classification methods. The best reported results with specifically engineered systems for these tasks are: 96.0% for Question classification Mou *et al.* (2015) and 93.9% for Subjectivity classification. Even with a limited hyperparameter tuning, we beat the current state of the art for question classification task with 96.2% using WR Triplet representation with CNN and we achieve a comparable performance for subjectivity classification task with an accuracy of 92.9% using WR Triplet with LSTM. Our representation does not depend on the classification setting and it would be interesting to see if with some more hyperparameter tuning any improvement in performance.

Also for the scope of this work, we did not perform evaluations on various state of the art neural network architectures which considers the structure of the sentence as well such as Ma *et al.* (2015) and Socher *et al.* (2013). We believe these neural network architectures with the added information of relation may improve the performance

The proposed method is independent of the adopted dependency parsing technique used in our work, and can be generalized to learn embeddings for relations; as produced by other syntactic Chen and Manning (2014) or semantic parsers Sharma *et al.* (2015). It would be interesting to see the effects of using different or more relations provided by other parsers, which can further be helpful in improving the performance of downstream tasks.

REFERENCES

Bansal, M., "Dependency link embeddings: Continuous representations of syntactic substructures", in "Proceedings of NAACL-HLT", pp. 102–108 (2015).

Bengio, Y., R. Ducharme, P. Vincent and C. Jauvin, "A neural probabilistic language model", Journal of machine learning research **3**, Feb, 1137–1155 (2003).

Bird, S., "Nltk: the natural language toolkit", in "Proceedings of the COLING/ACL on Interactive presentation sessions", pp. 69–72 (Association for Computational Linguistics, 2006).

Bordes, A., N. Usunier, S. Chopra and J. Weston, "Large-scale simple question answering with memory networks", arXiv preprint arXiv:1506.02075 (2015).

Bordes, A., J. Weston, R. Collobert and Y. Bengio, "Learning structured embeddings of knowledge bases", in "Conference on artificial intelligence", No. EPFL-CONF-192344 (2011).

Chen, D. and C. D. Manning, "A fast and accurate dependency parser using neural networks.", in "EMNLP", pp. 740–750 (2014).

Cho, K., B. Van Merriënboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", arXiv preprint arXiv:1409.1259 (2014).

Collobert, R., "Deep learning for efficient discriminative parsing.", in "AISTATS", vol. 15, pp. 224–232 (2011).

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, "Natural language processing (almost) from scratch", Journal of Machine Learning Research **12**, Aug, 2493–2537 (2011).

Cortes, C. and V. Vapnik, "Support-vector networks", Machine learning **20**, 3, 273–297 (1995).

Cybenko, G., "Approximation by superpositions of a sigmoidal function", Mathematics of Control, Signals, and Systems (MCSS) **2**, 4, 303–314 (1989).

Duchi, J., E. Hazan and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", Journal of Machine Learning Research **12**, Jul, 2121–2159 (2011).

Dyer, C., M. Ballesteros, W. Ling, A. Matthews and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory", arXiv preprint arXiv:1505.08075 (2015).

Elman, J. L., "Finding structure in time", Cognitive science **14**, 2, 179–211 (1990).

Finkelstein, L., E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman and E. Ruppin, “Placing search in context: The concept revisited”, in “Proceedings of the 10th international conference on World Wide Web”, pp. 406–414 (ACM, 2001).

Firth, J. R., “Modes of meaning. papers in linguistics 1934-51, 190–215”, (1957).

Hill, F., R. Reichart and A. Korhonen, “Simlex-999: Evaluating semantic models with (genuine) similarity estimation”, Computational Linguistics (2016).

Hochreiter, S. and J. Schmidhuber, “Long short-term memory”, Neural computation **9**, 8, 1735–1780 (1997).

Jones, E., T. Oliphant, P. Peterson *et al.*, “SciPy: Open source scientific tools for Python”, URL `http://www.scipy.org/`, [Online; accessed ¡today¿] (2001–).

Kalchbrenner, N., E. Grefenstette and P. Blunsom, “A convolutional neural network for modelling sentences”, arXiv preprint arXiv:1404.2188 (2014).

Kim, Y., “Convolutional neural networks for sentence classification”, arXiv preprint arXiv:1408.5882 (2014).

Kingma, D. and J. Ba, “Adam: A method for stochastic optimization”, arXiv preprint arXiv:1412.6980 (2014).

Kiros, R., Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba and S. Fidler, “Skip-thought vectors”, in “Advances in neural information processing systems”, pp. 3294–3302 (2015).

Komninos, A. and S. Manandhar, “Dependency based embeddings for sentence classification tasks”, in “Proceedings of NAACL-HLT”, pp. 1490–1500 (2016).

Kumar, A., O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani and R. Socher, “Ask me anything: Dynamic memory networks for natural language processing”, CoRR, abs/1506.07285 (2015).

Le, Q. V. and T. Mikolov, “Distributed representations of sentences and documents.”, in “ICML”, vol. 14, pp. 1188–1196 (2014).

LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE **86**, 11, 2278–2324 (1998).

Levy, O. and Y. Goldberg, “Dependency-based word embeddings.”, in “ACL (2)”, pp. 302–308 (Citeseer, 2014).

Li, X. and D. Roth, “Learning question classifiers”, in “Proceedings of the 19th international conference on Computational linguistics-Volume 1”, pp. 1–7 (Association for Computational Linguistics, 2002).

Lin, Y., Z. Liu, H. Luan, M. Sun, S. Rao and S. Liu, “Modeling relation paths for representation learning of knowledge bases”, arXiv preprint arXiv:1506.00379 (2015).

Liu, Y., F. Wei, S. Li, H. Ji, M. Zhou and H. Wang, "A dependency-based neural network for relation classification", arXiv preprint arXiv:1507.04646 (2015).

Luong, M.-T., H. Pham and C. D. Manning, "Effective approaches to attention-based neural machine translation", arXiv preprint arXiv:1508.04025 (2015).

Ma, M., L. Huang, B. Xiang and B. Zhou, "Dependency-based convolutional neural networks for sentence embedding", arXiv preprint arXiv:1507.01839 (2015).

Manning, C. D., M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard and D. McClosky, "The Stanford CoreNLP natural language processing toolkit", in "Association for Computational Linguistics (ACL) System Demonstrations", pp. 55–60 (2014), URL `http://www.aclweb.org/anthology/P/P14/P14-5010`.

Marelli, M., L. Bentivogli, M. Baroni, R. Bernardi, S. Menini and R. Zamparelli, "Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment", SemEval-2014 (2014).

McDonald, R., F. Pereira, K. Ribarov and J. Hajič, "Non-projective dependency parsing using spanning tree algorithms", in "Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing", pp. 523–530 (Association for Computational Linguistics, 2005).

Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space", arXiv preprint arXiv:1301.3781 (2013a).

Mikolov, T., M. Karafiát, L. Burget, J. Cernockỳ and S. Khudanpur, "Recurrent neural network based language model.", in "Interspeech", vol. 2, p. 3 (2010).

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality", in "Advances in neural information processing systems", pp. 3111–3119 (2013b).

Mou, L., H. Peng, G. Li, Y. Xu, L. Zhang and Z. Jin, "Discriminative neural sentence modeling by tree-based convolution", arXiv preprint arXiv:1504.01106 (2015).

Neelakantan, A., B. Roth and A. McCallum, "Compositional vector space models for knowledge base completion", arXiv preprint arXiv:1504.06662 (2015).

Pang, B. and L. 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", p. 271 (Association for Computational Linguistics, 2004).

Pedregosa, F., 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).

Pennington, J., R. Socher and C. D. Manning, “Glove: Global vectors for word representation.”, in “EMNLP”, vol. 14, pp. 1532–1543 (2014).

Pham, N., G. Kruszewski, A. Lazaridou and M. Baroni, “Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model”, ACL/IJCNLP (2015).

Pollack, J. B., “Recursive distributed representations”, Artificial Intelligence **46**, 1, 77–105 (1990).

Schuster, M. and K. K. Paliwal, “Bidirectional recurrent neural networks”, IEEE Transactions on Signal Processing **45**, 11, 2673–2681 (1997).

Schuster, S., R. Krishna, A. Chang, L. Fei-Fei and C. D. Manning, “Generating semantically precise scene graphs from textual descriptions for improved image retrieval”, in “Proceedings of the Fourth Workshop on Vision and Language”, pp. 70–80 (2015).

Schuster, S. and C. D. Manning, “Enhanced english universal dependencies: An improved representation for natural language understanding tasks”, in “Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)”, (2016).

Sharma, A., N. H. Vo, S. Aditya and C. Baral, “Towards addressing the winograd schema challenge-building and using a semantic parser and a knowledge hunting module.”, in “IJCAI”, pp. 1319–1325 (2015).

Shen, Y., X. He, J. Gao, L. Deng and G. Mesnil, “Learning semantic representations using convolutional neural networks for web search”, in “Proceedings of the 23rd International Conference on World Wide Web”, pp. 373–374 (ACM, 2014).

Socher, R., B. Huval, C. D. Manning and A. 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”, pp. 1201–1211 (Association for Computational Linguistics, 2012).

Socher, R., C. D. Manning and A. Y. Ng, “Learning continuous phrase representations and syntactic parsing with recursive neural networks”, in “Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop”, pp. 1–9 (2010).

Socher, R., J. Pennington, E. H. Huang, A. Y. Ng and C. D. Manning, “Semi-supervised recursive autoencoders for predicting sentiment distributions”, in “Proceedings of the conference on empirical methods in natural language processing”, pp. 151–161 (Association for Computational Linguistics, 2011).

Socher, R., A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts *et al.*, “Recursive deep models for semantic compositionality over a sentiment treebank”, in “Proceedings of the conference on empirical methods in natural language processing (EMNLP)”, vol. 1631, p. 1642 (Citeseer, 2013).

Srivastava, N., G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.", Journal of Machine Learning Research **15**, 1, 1929–1958 (2014).

Sutskever, I., O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks", in "Advances in neural information processing systems", pp. 3104–3112 (2014).

Tai, K. S., R. Socher and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks", arXiv preprint arXiv:1503.00075 (2015).

Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions", arXiv e-prints **abs/1605.02688**, URL `http://arxiv.org/abs/1605.02688` (2016).

Turney, P. D. and P. Pantel, "From frequency to meaning: Vector space models of semantics", Journal of artificial intelligence research **37**, 141–188 (2010).

Wang, S. and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification", in "Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2", pp. 90–94 (Association for Computational Linguistics, 2012).

Yin, W., H. Schütze, B. Xiang and B. Zhou, "Abcnn: Attention-based convolutional neural network for modeling sentence pairs", arXiv preprint arXiv:1512.05193 (2015).

Yin, Y., F. Wei, L. Dong, K. Xu, M. Zhang and M. Zhou, "Unsupervised word and dependency path embeddings for aspect term extraction", arXiv preprint arXiv:1605.07843 (2016).

Zhang, R., H. Lee and D. Radev, "Dependency sensitive convolutional neural networks for modeling sentences and documents", arXiv preprint arXiv:1611.02361 (2016).

Zhao, H., Z. Lu and P. Poupart, "Self-adaptive hierarchical sentence model", arXiv preprint arXiv:1504.05070 (2015).