EVALUATION OF DEEP LEARNING ALGORITHMS IN SENTIMENT ANALYSIS

by

Sani Kamış

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in
Computer Engineering

Yeditepe University

2019

EVALUATION OF DEEP LEARNING ALGORITHMS IN SENTIMENT ANALYSIS

APPROVED BY:

Assist. Prof. Dr. Dionysis Goularas (Thesis Supervisor) (Yeditepe University)	[illegible] ....
Prof. Dr. Emin Erkan Korkmaz (Yeditepe University)	[illegible] ....
Assoc. Prof. Dr. Özlem Durmaz İncel (Galatasaray University)	[illegible] ....

DATE OF APPROVAL: ..../..../2019

ACKNOWLEDGEMENTS

The present document, resulting of three years of research, would not have been possible without the help, encouragements and support of many persons. I thank all of them and I present them all my gratitude.

Special thanks to my supervisor, Asst. Prof. Dr. Dionysis Goularas, who initiated this research, supported, and trusted me to complete from beginning to end.

I want to thank my manager, Özlem Topuz Öngel, for her encouragement and support in the first year.

I am grateful to the company I work for, Softtech, for academic support.

I also appreciate the participation of Assoc. Prof. Dr. Özlem Durmaz İncel to the jury and her advices throughout the process.

I want to thank Prof. Dr. Emin Erkan Korkmaz for his fruitful comments.

Finally, I would like to especially thank my wife, Serpil, for her unconditional love, understanding, support, and encouragement for this research.

ABSTRACT

EVALUATION OF DEEP LEARNING ALGORITHMS IN SENTIMENT ANALYSIS

Deep Learning (DL) techniques have played an important role in the solution of a wide range of problems. Convolutional Neural Networks (CNN) are especially good at image processing tasks. Recurrent Neural Networks (RNN) are usually applied in Natural Language Processing (NLP) tasks. Sentiment analysis has been a popular area of research because people's opinions are important for each other and people have been sharing their opinions in social media freely. In the literature, there are studies that use simple deep learning techniques or their combinations in sentiment analysis. In this thesis, we evaluate different deep learning techniques. The learning models we compare are CNN, Long Short-Term Memory Networks (LSTM), and their ensembles and combinations. Moreover, we use Support Vector Machines (SVM) in a combination. In addition to these models, we compare different word embedding techniques such as Word2Vec and Global Vectors for Word Representation (GloVe) models. We focus on sentiment analysis from Twitter Data provided in Semantic Evaluation (SemEval), which is one of the most popular international workshops on semantic evaluation. Sentiment analysis work consists of two main steps. First phase is the creation of word embeddings, Word2Vec and GloVe models are compared in this step. Second phase is supervised training. Here, we apply CNN, LSTM, SVM, and their various combinations. Additionally, we have compared voting individual results of different learning techniques and their more organic combinations. All these combinations are tried with different weights and parameters, and best scoring values of each model is compared in terms of accuracy, precision, recall, and F-score.

ÖZET

DUYGU ANALİZİNDE DERİN ÖĞRENME ALGORİTMALARININ KARŞILAŞTIRILMASI

Derin öğrenme teknikleri son zamanlarda pek çok problemin çözümünde önemli bir rol oynamaya başladı. Convolutional Neural Networks (CNN) özellikle görüntü işleme uygulamalarında çok başarılı sonuçlar vermektedir. Recurrent Neural Networks (RNN) ise genelde Doğal Dil İşleme (NLP) uygulamalarında kullanılmaktadır. Duygu analizi son zamanlarda çok poüler bir araştırma alanı olmaya başladı, çünkü insanlar birbirlerinin düşüncelerini merak ediyorlar ve sosyala medyada özgürce düşüncelerini paylaşabiliyorlar. Literatürde, derin öğrenme tekniklerini tek başına veya birden fazla tekniği bir arada kullanan çalışmalar vardır. Bu çalışmada farklı öğrenme metotlarını karşılaştırılmaktadır. Karşılaştırdğımız modeller arasında CNN, Long Short-Term Memory Networks (LSTM), Support Vector Machines (SVM) ve bu modellerin farklı kombinasyonları yer almaktadır. Modellere ilaveten Word2Vec ve Global Vectors for Word Representation (GloVe) gibi kelime vektörleri de karşılaştırılmaktadır. Bu çalışmada dünya çapında en popüler doğal dil işleme organizasyonlarından birisi olan SemEval'ın sağladığı Twitter datası ile duygu analizi yapılmaktadır. Duygu analizi iki temel aşamadan oluşmaktadır. İlk aşamada GloVe veya word2vec ile kelimeler vektöre dönüştürülmektedir. İkinci aşama ise öğretmenli öğrenme aşamasıdır. Bu aşamada, CNN, LSTM, SVM ve kombinasyonları denenmekte ve sonuçları karşılaştırılmaktadır. Ek olarak farklı öğrenme tekniklerinin ayrı ayrı denenip sonuçlarının birleştirilmesi veya farklı öğrenme tekniklerinin daha organik bir yapıda bir arada kullanıldığı modeller de karşılaştırılmaktadır. Bütün kombinasyonların faklı ağırlık ve parametrelerle denenip en iyi sonucu verenler doğruluk yüzdesi (accuracy), kesinlik (precision), hassasiyet (recall) ve F-ölçütü (F-score) açısından karşılaştırılmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

ANN	Artificial neural network
BLSTM	Bidirectional long short-term memory unit
BPTT	Back propagation through time
CNN	Convolutional neural network
CRNN	Convolutional recurrent neural network
DL	Deep learning
DNN	Deep neural network
FNN	Feed forward neural network
GloVe	Global vectors for word representation
GRNN	Gated recurrent neural network
GRU	Gated recurrent unit
LCA	Lexicon integrated convolutional neural networks with attention
LR	Logistic regression
LSTM	Long short-term memory unit
MA	Mean average
MAD	Mean absolute deviation
MAE	Mean absolute error
ML	Machine learning
MLP	Multilayer perceptron
MSE	Mean squared error
NB	Naïve bayes
NN	Neural network
NRMSE	Normalized root mean squared error
POS	Part of speech
RCNN	Recurrent convolutional neural network
ReLU	Rectified linear unit
RF	Random forest
RMSE	Root mean squared error
RNN	Recurrent neural network
SA	Sentiment analysis

SAT	Sentiment analysis in twitter
SemEval	Semantic evaluation
SST	Stanford Sentiment Treebank
SSWE	Sentiment specific word embeddings
SVM	Support vector machine
TF-IDF	Term frequency – inverse document frequency

1. INTRODUCTION

Social media has been a very popular tool for sharing ideas over the internet. People feel free to share their opinions about products, companies, services, topics on the agenda, etc. by means of social media. Since people have a high interest in the feelings and opinions of others, sentiment analysis in social media has been a very important topic. It can be beneficial for companies, service providers, and even for governments for making decisions. Twitter is one of the most popular social networks on/through which people can share their status (tweets). Twitter data are the most preferred data for sentiment analysis task in the literature because of its data structure consisting of sentences limited to 280 characters at most. It contains a short and meaningful text from which we can extract sentiments.

Sentiment analysis can be done in different ways. One of the most common ways is to calculate positivity, negativity, and objectivity scores and decide the belonging class of a text such as positive, negative, and neutral as a classification task. Other option is to find a positivity score ranging from zero to five or some other values as a quantification task. In this research we have concentrated on a three scale classification for sentiment analysis.

With the importance and popularity of sentiment analysis task, lots of algorithms have been tried for having better scores for this task. Various natural language processing and machine learning techniques are applied for sentiment classification. Recently, deep learning algorithms like CNN and LSTM networks have shown state of the art scores in sentiment analysis. These scores are high in the literature but they are not satisfactory for the implementation needs in real life. Therefore, it is an important interest area for researchers. People have been trying different combinations and ensembles of these deep learning methods for increasing accuracy. In this research, our contribution is evaluation of these deep learning methodologies and their different combination and ensembles with a common Twitter data provided by SemEval contest in a single testing environment.

Deep learning algorithms need numeric values for training and testing issues. In sentiment analysis, input data consist of text rather than numbers, so sentiment analysis task can be splitted into two main steps.

First phase of sentiment analysis is the creation of vectoral representations of words. This task is completed with word embedding techniques. The main idea in the word embedding is based on putting words in a multi-dimensional vectoral space in such a way that words are placed in terms of their relations with each other. There are a lot of successful word embedding techniques which are based on the number of occurrences of words and locations of words in sentences. Two of the most popular word embeddings are Word2Vec and Global Vectors for Word Representation (GloVe) models. We have compared two different samples of these two word embedding algorithms. Word2Vec vectors are created from SemEval tweets belonging to previous years' competitions and pre-trained Glove Vectors are provided by Stanford University Natural Language Processing Group, have been trained from two billion tweets with a large corpus size of 27 billion words.

Second phase is the deep learning models. Deep learning algorithms we use consist of supervised training. In this step, we apply several deep learning techniques, such as Convolutional Neural Networks (CNN), Long Short-Term Memory Networks (LSTM), as well as their unions and combinations. Moreover, we use Support Vector Machines (SVM) in a combination. In addition to different learning models, a regional structure similar to architecture developed by [1] is compared with standard learning models.

Finally, we compare the results obtained in the second phase in terms of accuracy, precision, recall, and F-score. We see that all these variations of deep learning techniques did not yield too different results in terms of accuracy. Data have much more important impact than the learning technique utilized. We were able to increase accuracy up to seven percent by just evaluating with different word vectors. Although self-trained word2vec word embedding were created from similar tweets which were consisting of previous years' SemEval data with 31007 tweets with 662000 corpus size and 10000 words in vocabulary, pre-trained GloVe vectors scored better since they were created from much more larger data with a huge corpus size of 27 billion words created from two billion tweets, with a vocabulary size of 1.2 million words.

In this thesis, we focus on sentiment analysis from Twitter Data provided by Semantic Evaluation (SemEval) competition. SemEval is one of the most popular international workshops on semantic evaluation like Kaggle, EMSASW, etc. SemEval provides competitors and other researchers with tweets for both training and testing data. Because of Twitter policy laws, they are not allowed to share text of the tweets, they are only able to share status IDs of tweets and researchers have to download tweets from Twitter servers with provided status IDs. Therefore, we have developed an application for downloading active tweets belonging to given tweet IDs from Twitter servers via LinqToTwitter API [2] provided by Twitter.

This thesis is organized as follows. Section 2 gives an insight in the field of Sentiment analysis by mentioning related works done in the literature. Different learning models such as CNN and LSTM are used individually and also with their ensembles and combinations. Details about these learning methods are described in Section 3. Section 4 is about our analysis and design of the learning models describing operations step by step. Section 5 includes our implementation of the models with the details of tools and technologies used in this research. In Section 6, comparison of different learning models are mentioned with the test details and evaluation results. Finally, Section 7 concludes the findings and includes some information towards future works.

2. BACKGROUND

In recent years, social media have started to play an important role on people’s lives. People are easily sharing their opinions and feelings about any subject. At the same time, users are taking other users’ comments into consideration at any research. This research can be about any topic like products, companies, services, countries, cultures, etc. Therefore, sentiment analysis has been a popular research area in computer science. By means of sentiment analysis in social media, we can easily learn people’s feelings about any service, product, idea, etc. Especially, Twitter is the best social media type for sentiment analysis since it has short texts up to 140 characters in a tweet. Another ease of Twitter is that it provides developers with APIs to get tweets from all around the world. So, Twitter is a proper big data source for easily collecting tweets all around the world. Therefore, most of the researches in sentiment analysis deal with sentiments of tweets. Bo Pang and Lillian Lee presented different methodologies that were used for sentiment analysis up to 2008[3]. Recently, Deep Neural Networks have been widely used because of its success with state of the art results in this task. Especially, CNNs [4,5] and RNNs [6] are used because CNN is good at resolving the dimensionality reduction problem and a special implementation of RNN, the LSTM networks [6], handle with success temporal or sequential data with the help of its gated structure [7].

2.1. CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional Neural Networks are mainly used for image classification task in machine learning. It is based on convolving filters on images to find out relevance to classes. In Natural Language Processing (NLP) tasks, words are converted to vectoral representation with word embedding techniques and sentences are generally represented by concatenating created word vectors. State of the art methods have proved that CNN is very successful at sentiment analysis like its success in other classification tasks with image files. Works presented in [8] and [9] are two of the leading sample works that CNN architectures can be used successfully for sentiment analysis.

SemEval 2016 top-scoring team in two-scale classification (Tweester[10]) used a combination of CNNs, topic modeling, and word embeddings generated via word2vec[11], so reached 86 percent in two-scale classification.

One of the simplest way to use deep learning techniques in sentiment analysis is the usage of CNNs after converting sentences into vectoral representations via some word embedding techniques. In a survey of deep learning methods in sentiment analysis [12], it can be seen that the word embedding is done mainly with two methods, Word2Vec [11] or GloVe [13]. In addition, alternative embedding types like FastText[14] , Stanford Sentiment Treebank (SST) are used for embedding. After that, you can get good results by some fine tuning the parameters of CNN. In general, convolution layer, some pooling layer, and fully connected layers are used for prediction. Some teams [15-17] have tried this standard CNN structure as represented in Figure 2.1[18], created by Yoon Kim, for sentiment analysis with different data & weights. It is stated that simple CNN with one convolution layer can performs remarkably well with a little parameter tuning[18].

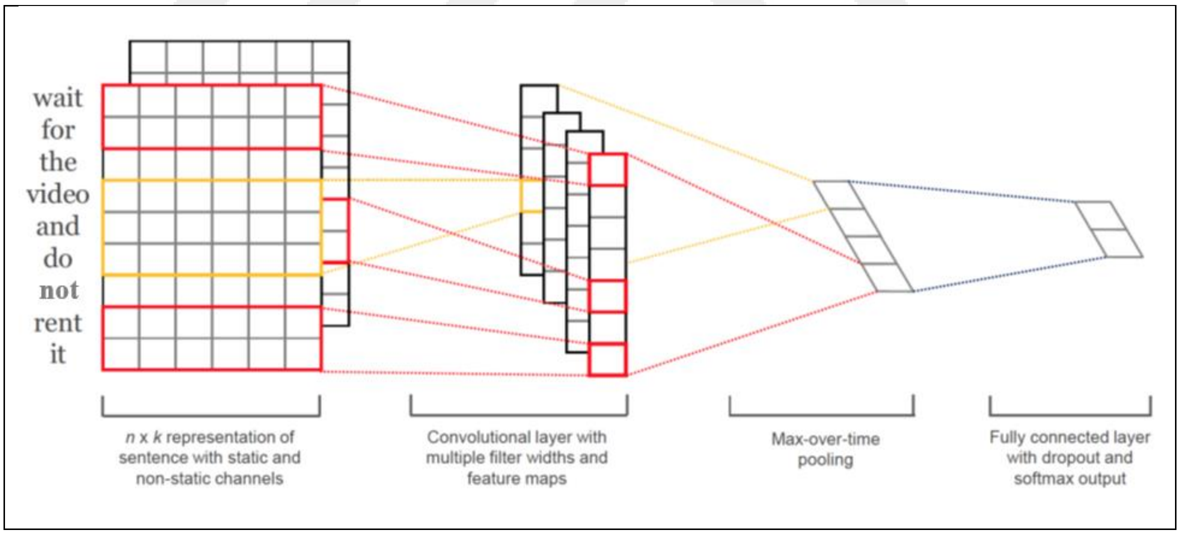

wait
for
the
video
and
do
not
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 2.1. CNN model or sentiment analysis [18]

Another brilliant option is to make use of multiple individual CNN models with different weights and parameters. After individual predictions of each CNN model, results are combined to get final prediction by using different algorithms such as soft-voting or hard-voting to combine results. In soft-voting, average possibility value of each class are calculated among all models. In hard-voting, total number of votes for each class is used for the final prediction. Accuracy of multiple CNNs can be increased with additional deep learning layers. [15-17] used multiple CNN models with different weights and parameters.

NNEMBs team [17] scored first in accuracy in SemEval 2017 by using several word embedding sets and a special gated convolution model, Recurrent Convolutional Neural Network (RCNN) explained in [19]. When predicting the labels of tweets in testing set, label probabilities of all neural network are sum up to make final decisions.

Ten convolutional neural networks and hard voting technique are used to decide the final sentiment in [15]. Each CNN network consists of a convolutional layer followed by a fully-connected layer and a soft-max on top. Ten instances of this network are initialized with the same word embeddings as inputs but with different initializations for the network weights. Number of classifiers which give the positive, negative and neutral sentiment label to each tweet are counted to decide sentiment label which have the highest number of votes.

Additional algorithms can be used in addition to CNN. In [16], two different two-layer CNNs are used and results are concatenated to a Random Forest (RF) classifier.

Usage of different embeddings in addition to word embeddings is used in the literature. Some researchers have made use of lexicon embeddings in addition to word embeddings in order to increase accuracy in sentiment analysis. Topic, lexical, part of speech (POS), sentiment, character-level embeddings are used.

Lexicon embedding and word embedding are concatenated by Adullam team at SemEval-2017 [20]. In order to improve on existing CNN based sentiment analyser, lexicon embedding and attention embedding were integrated into the proposed sentiment analyser. The proposed lexicon integrated convolutional neural networks with attention (LCA) consists of three input features such as word embeddings, lexicon embeddings, and attention embeddings.

Concatenation of word embedding and sentiment embedding vectors are used in [21] similar to Adullam[20] at SemEval-2017. Difference is that they do not use attention embedding.

Two different embeddings such as Word2Vec and Sentiment Specific Word Embeddings (SSWE) are used in [22]. Because the dimensionality of vectors in SSWE is 50, it is extended to 300 dimensional vectors by padding the 250 randomly generated numbers to the end.

EICA team [23] concatenates word embedding with topic embedding at SemEval 2017. They use two different CNNs, one for word embedding and the other is for topic embedding. Results are concatenated and input to a softmax layer to get a final prediction.

Lexical, part-of-speech and sentiment embeddings are used for CNNs and results are concatenated and fed to a final Deep Neural Network (DNN) by SENSEI-LIF team[24] . This model has ranked 2nd at SemEval-2016[25] task 4.

System developed in [26] combines CNN and Logistic Regression (LR). So, it uses both embedding features and various features like lexicons and dictionaries. The final prediction for sentiment is a combination of predictions given by both classifiers. The architecture contains two separate CNNs: one is for word-based input maps while the other is for character-based input maps. Fully connected layer is used for final prediction of CNN. They have observed that, the performance of the system does not increase, but drops by simply adding LR features to CNN features.

The system developed by Tweester team[10] is comprised of multiple independent models such as neural networks, semantic-affective models and topic modeling that are combined in a probabilistic way. Each model is used to predict a tweet's sentiment (positive, negative or neutral) and a late fusion scheme is adopted for the final decision. The motivation behind the development of various systems for sentiment classification is that different systems may capture different aspects of the sentiment, and by combining them we can predict more accurately the sentiment of tweets.

In [9], a new deep neural network architecture that jointly uses character-level, word-level and sentence-level representations is presented to perform sentiment analysis.

Another interest area in sentiment analysis is to predict sentiments from images instead of shot texts like tweets. Sentiment analysis from Flickr and Twitter images is done in the research [27].

2.2. LONG SHORT-TERM MEMORY NETWORKS

Usage of Recurrent Neural Networks (RNN) is another important Deep Learning method for Sentiment Analysis. RNNs process input sequentially, so they can take positions of words into consideration in a sentence to predict the sentiment value. But, RNNs have two important problems which are called vanishing gradient problem & exploding gradient problem [28]. LSTM is a special form of RNNs which overcomes vanishing gradient problem by keeping cell states through gates[7]. Technically speaking, model structure is the same as CNN architecture. Firstly, tweets are cleaned. Secondly, word embeddings are created & tweets are represented in a vectoral format by combining word vectors. Later, vectoral representations of tweets are fed to an LSTM layer. Finally, a softmax layer and a fully connected layer are applied for final prediction.

In [29] the efficiency of RNN was demonstrated as they outperformed the state of the art methods. DataStories team at Semeval 2017 uses a 2- layer bi-directional LSTM for sentiment analysis[30]. Deep Learning (DL) model of DataStories is stated in Figure 2.2. They have ranked first in Subtask 4 which is to predict whether a tweet is positive, negative, or neutral. BUSEM team[31] have tried two different deep learning models, one of which was LSTM in SemEval 2017. GRU networks, introduced in 2014 [32] can be used effectively instead of LSTM [33] with showing similar results because it has similar architecture with LSTM by making use of gates.

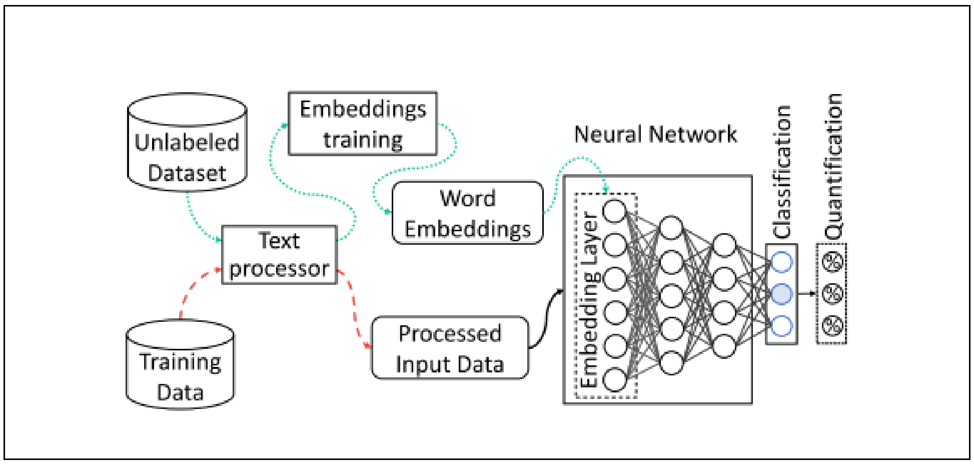


Figure 2.2. Deep learning model for sentiment analysis [30]

2.3. SUPPORT VECTOR MACHINES

Support Vector Machines are as popular as CNNs in Sentiment Analysis task. There are a lot of researches in the literature those make use of SVMs in Sentiment Analysis in Twitter (SAT) with alternative features [15-29]. As we have seen in previous sections, convolutions are generally based on different word embeddings or lexical embeddings. SVMs are also based on different type of embeddings. In addition, SVMs are fed with a large variety of extracted features. Most common features used in addition to embeddings are; n-gram features, negation features, number of hashtags, number of emoticons, TF-IDF features, number of capitalized words, number of negative words, number of positive words, etc. There are so many features used in SVM that it can be another research area to detect most affective feature for Sentiment Analysis. Some researchers have preferred to decide sentiment in two phases like objectivity detection and polarity detection instead of predicting sentiment in a single step [35-44].

2.4. ENSEMBLES AND COMBINATIONS

There are so many studies done on Sentiment Analysis with DL techniques that it has resulted in trying different ensembles & combinations of DL techniques. Especially the ones which participate in SemEval competition are very successful in this way. Top scorers on SemEval 2017 Subtask A [50] , that is to classify English tweets into three categories (positive, negative, and neutral), generally use Deep Learning techniques to decide sentiments in tweets.

2.4.1. CNN and LSTM

Mostly used ensemble in the literature is combination of CNN and LSTM. In [51], an implementation of CNN and LSTM networks was presented, showing the significant advantages of using together these two neural networks. One of the top scorers BB_twtr[52] team who had scored the same as DataStories has tried different deep learning methods which are CNN and LSTM. In addition they have done a combination of both. They are using zero-padding strategy to make all tweets have same matrix dimension. Since LSTM doesn't sufficiently take into account post word information, they use bidirectional LSTM to solve this problem. For combination, they have used an ensemble of 10 CNNs and 10 LSTMs together through soft voting, which means taking the average of probabilities for each class and deciding the sentiment according to average.

Some other researches have been concentrated on more organically combination of CNN & LSTM rather than an ensemble. 3rd top scorer LIA[53] team has used an ensemble of CNN and LSTM. One lexical embedding and three different sentiment embeddings are fed to four different DNN. Each DNN has a CNN and an LSTM layer sequentially and the resulting vectors are fed to a Multi-Layer Perceptron. In [54], a similar method is applied with [53] with three different embeddings and using Bidirectional Long Short-Term Memory Unit (BLSTM) instead of LSTM layer. Convolutional Recurrent Neural Network (CRNN) in the Figure 2.3 represents a combination of CNN and BLSTM layers as represented in Figure 2.4

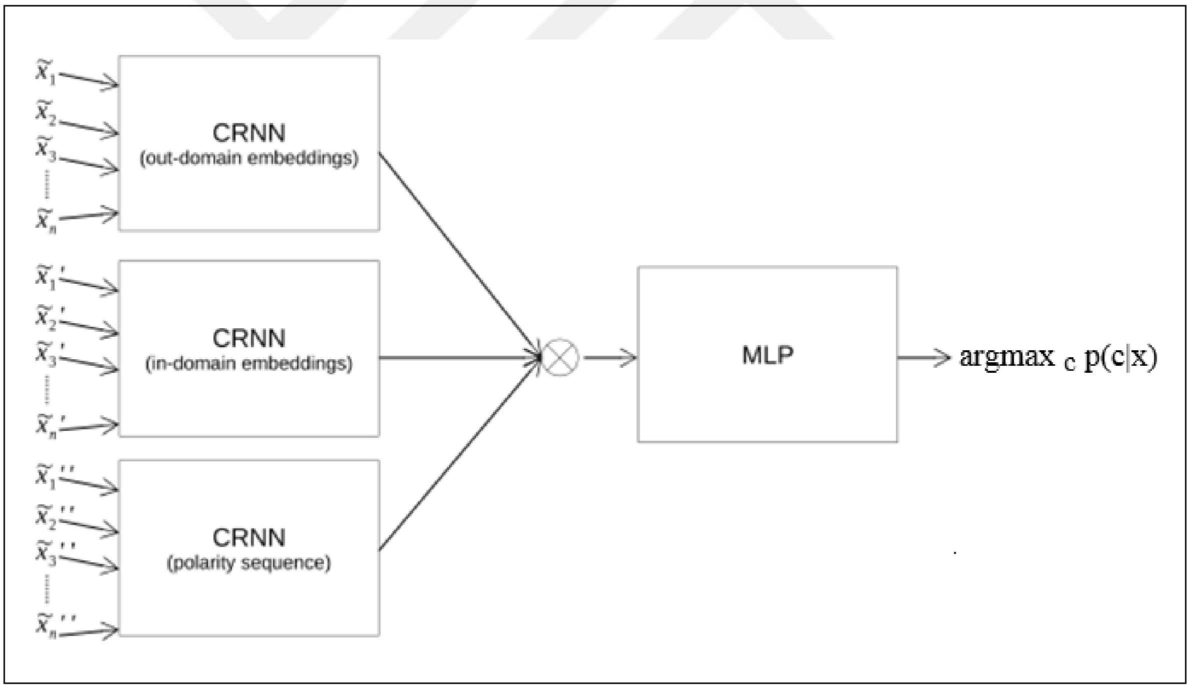


Figure 2.3. System architecture of the CNN-LSTM-MLP model [54]

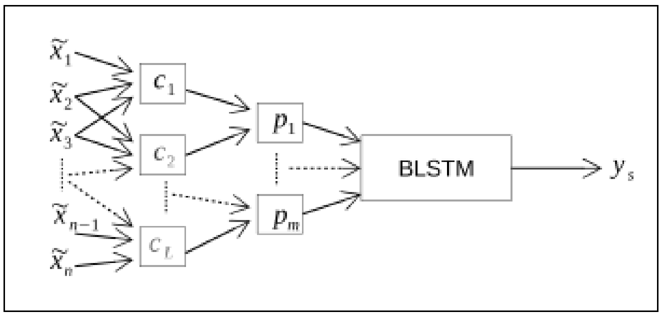


Figure 2.4. Implementation of the convolutional recurrent neural network. [54]

Another way to use both CNN and LSTM organically is to use varying length filters in CNN layers and to use outputs of CNNs in LSTM layers [55].

[1] divides tweets into regions, each region is fed to a CNN and resulting vectors are fed to an LSTM layer as shown in Figure 2.3. This model can capture both local information within sentences via CNN and long distance dependency across sentences with the help of LSTM.

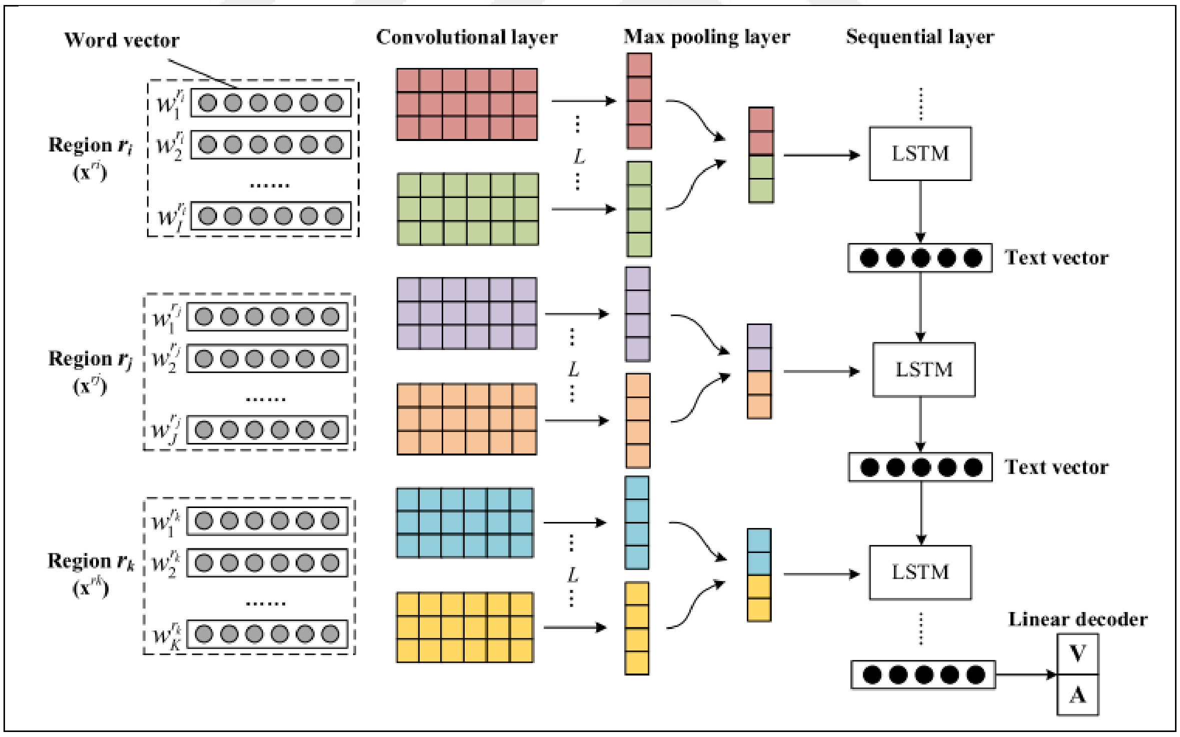


Figure 2.5. System architecture of the regional CNN-LSTM model [1]

There are other simple ideas to make use of both LSTM and CNN that is to use them separately and combine results in some alternative ways. [56] combines the results with an interpolation algorithm where weight of algorithm is decided with grid search. [57] combines the results through soft-voting. [58] uses two CNN models and a Gated Recurrent Unit (GRU) model separately and combines the results with another CNN model. They use Gated Recurrent Neural Network (GRNN) instead of LSTM because they have got better results on GRU with their data. On the other hand, [57] states that LSTM hits GRU in sentiment analysis task. GRU and LSTM can be used interchangeably in the literature, they have similar results. It depends on data and parameters of the models used, we have use LSTM in our research.

2.4.2. LSTM and SVM

As we have stated in the previous section, some researchers prefer to use GRU instead of LSTM because of better accuracy in their data and model. [49] is another supporter of GRU. A gated recurrent neural network is trained using pretrained word embeddings, then feature are extracted from GRU layer and these features are input to SVM for classification.

Some researchers have tried multiple deep learning algorithms for just comparison rather than using them in an ensemble or combination. BUSEM team[31] has done a research in this manner in SemEval 2016, they have tried SVM, RF, NB, and LSTM independently. Finally, they have resulted in SVM scores best among these alternatives in SA.

2.4.3. CNN and SVM

In the literature, CNN and SVM is used sequentially like the structure in CNN and LSTM. In [59], CNN is applied first and outputs of CNN are input to SVM together with additional linguistic information. Proposed model performs better than single CNN.

In [60], two CNN and LSTM are applied separately. SVM classifier features are based on a bag of words model, the CNN classifier is initialized with pre-trained word vectors. Finally, ensemble model counts votes from three classifiers and predicts the class which has the maximum number of votes from the three classes.

2.5. SEMANTIC EVALUATION (SemEval)

Increasing need and interest to sentiment analysis in computer science have resulted in science competitions for sentiment analysis. SemEval is one of the most popular competitions in this area, which is an international workshop on semantic evaluation. Sentiment analysis competition is repeated annually with small changes in tasks.

In SemEval 2016 [25] & SemEval 2017 [50], top scorer teams used deep learning for the task 4 of sentiment analysis, which is to group sentiments into three categories such as positive, negative, and neutral. They have mainly used CNN, LSTM, SVM, and their ensembles and combinations. In SemEval 2016[25], half of the ten top-ranked participating teams used convolutional neural networks; three teams were using recurrent neural networks, and seven teams used ensembles and combinations in their systems. Usage of classifiers such as support vector machines, which were dominant until recent years, seems to have lost its popularity because of its low efficiency compared to convolution and recurrent neural networks. In SemEval 2017[50], we again see an increasing interest towards deep learning rather than SVM. There were at least 20 teams who used deep learning and neural network methods such as CNN and LSTM networks.

3. METHODOLOGY

3.1. CONVOLUTIONAL NEURAL NETWORKS

Artificial intelligence and neural network deal with the machines' learning in the same way people learn. One of the most important neural networks, which is convolutional neural networks (CNN) exactly behaves in this manner. For instance, we are born with no knowledge about animals. Throughout our lives, we see a lot of animals like cats, dogs, birds, etc. in our living space. We learn some wild animals, sea animals from the documentaries displayed in television or over the internet, which is not possible for us to see in real life. While some of us can group animals into larger categories like animal, cat, dog, etc., some people may group them in more detail for example into different cat categories like lion, tiger, etc. The difference between people come from their earlier experiences, that means it depends on how much different samples of these animals are seen and grouped by people before. If we have seen more animals or dealed much more with animals, we can have a broader vision and have more accurate knowledge about them. We can do these grouping even for the ones which we have seen first time in our lives unintentionally from our previous knowledge. Before dealing with machine learning, I had never talked about how I am grouping animals, for instance cat, dog, or bird. Of course, it is very easy to discriminate the birds from dogs and cats because they have wings, beaks, they have two feet as another difference from dog and cat. They have lots of other filterable characteristics visually. What about the difference between cats and dogs? We can categorize cats and dogs from their eyes, noses, ears, tails, etc. We can even classify different categories of cats for a more detailed visual characteristics like eye colors, fur shapes, etc. though they are not as clear as more general classification attributes among cats and birds.

So far, I have mentioned about how people's learn different animals from their experiences, categorization of animals from larger to more detailed comparisons, visual characteristics we have extracted from our previous scenes. The idea behind the grouping of the CNN is exactly in the same way. Let us continue with the technical details of CNN while keeping these samples in mind.

CNN is a special type of ANN which uses supervised learning algorithm. Supervised learning algorithm means that algorithm learns from given labeled sample inputs. For each image we give in our training set we have a class. This class may be a general classification item label like bird, cat, etc. or it may be a more detailed categorization class of cat species. Learning of the algorithm to which extent depends on the given classes like our detailed or general learning example above. If we give samples of specific cat species in our training set, algorithm learns these details. If we give only cat, bird, and dog classes in the labelled training set, algorithm only learns about grouping these categories.

Let us take a deeper look into CNN and talk about how it makes classification. As we have mentioned above people can distinguish between different categories by means of visual characteristics of each class. It is the same in convolutional neural networks. Images are categorized in terms of filters. Convolution is the process of convolving filters over images. To find the probability of containing an object, or belonging to a special category, filters are applied pixel by pixel by sliding it through an image as shown in the Figure 3.2 and similarity of filter to convolved space is calculated. Applying filter means that element-wise matrix multiplication is applied and results are summed. Feature map is obtained by applying filters over the image and it shows the probability of containing a filter. In Figure 3.2, a 3x3 filter is applied to a 7x7 image and a 5x5 feature map is obtained as the output of convolution process. Width and height of the feature map are calculated with the below equations, where W and H refers to width and height of input image, Fw is filter width, Fh is filter height, P value is padding, Sw is stride width, and Sh is stride height.

$$output\ width = \frac{W - Fw + 2P}{Sw} + 1 \tag{3.1}$$

$$output\ height = \frac{H - Fh + 2P}{Sh} + 1 \tag{3.2}$$

Items in the above equations can be described as;

Filter is the convolved sub image over the input. It is also called as kernel. It may be bird’ s beak or tail, cat’ s ear or eye, dog’ s tongue or tail, etc. as shown in Figure 3.1 related to our dog, cat, and bird classification sample.

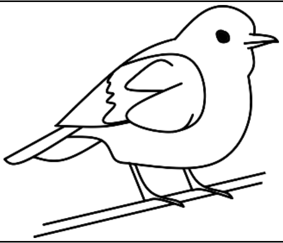
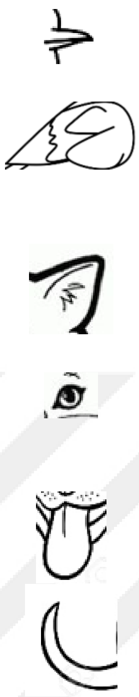
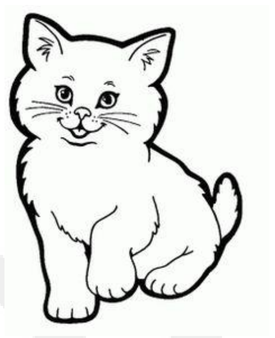
SAMPLE IMAGES	FILTERS	FEATURE MAPS
		Higher results for filters with bird' s beak and tail
		Higher results for filters with cat' s ear and eye
		Higher results for filters with dog' s tongue and tail

Figure 3.1. Images and convolution filters

Filter is applied all over the image by sliding it from left to right and from up to down. In Figure 3.2 convolution is applied by sliding filter one by one in both directions. It could be done by sliding filter with more than one in each iteration. Stride value refers to number of sliding cells.

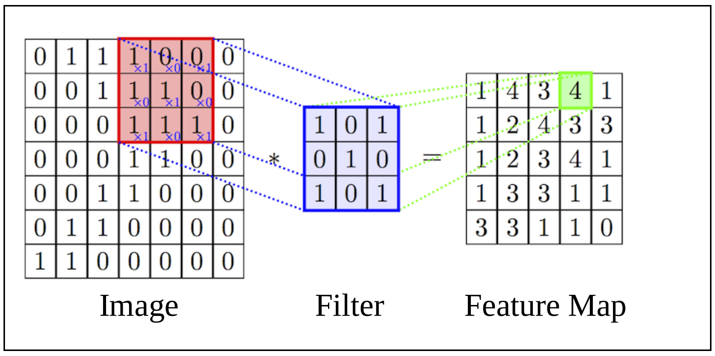


Figure 3.2. Convolution process

After applying filters, dimension decreases in the output of convolution. In some cases it is needed to have same dimensions after convolution, therefore, padding is applied. Padding means assuming that input image has larger dimensions and outer cells have zero values an filters are applied starting from appended cells and goes until the end of appended zeros as shown in Figure 3.3. Padding value is number of appended rows and columns.

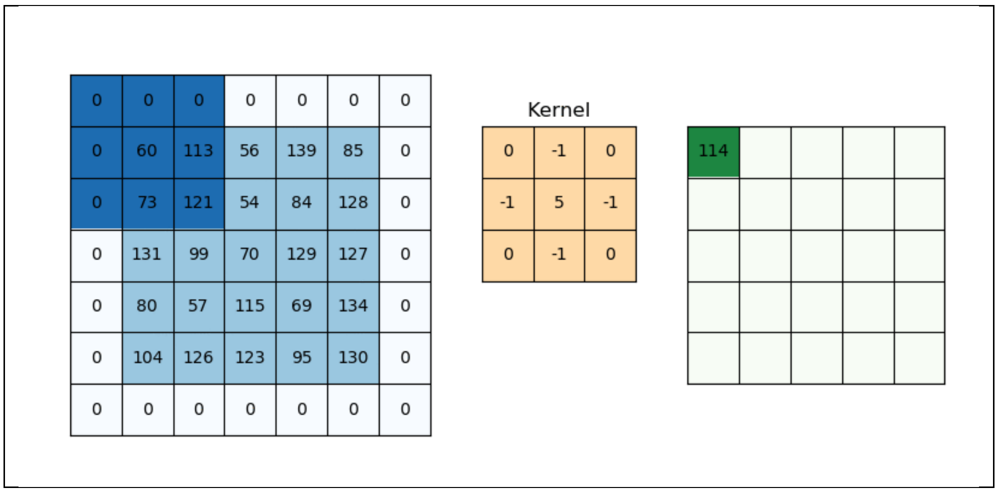


Figure 3.3. Convolution filter with padding

In convolutional neural networks, some pooling is applied after convolution process. The most commonly used pooling is max pooling, where maximum of the numbers in the pool is selected and projected to the output as shown in Figure 3.4. In this sample, max pooling with 2x2 windows is applied to a 4x4 input matrix and 2x2 output matrix is obtained. Another commonly used option is to take average of values instead of taking the highest value, it is called mean pooling. Pooling causes the input size to get smaller while keeping the values in bigger image. It is a process like squeezing the image in the feature map. This provides us with easier calculation processes in the later layers.

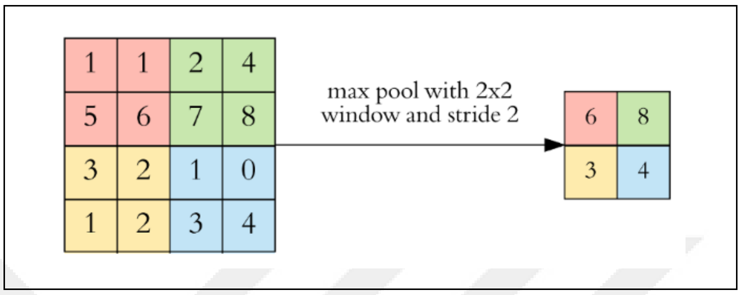


Figure 3.4. Max pooling

Next and the final layer after max pooling is fully connected layer to classify input image. One or multiple fully connected layers are used and a vector is output including the possibility values for each class. Structure of layers in a CNN is shown in Figure 3.5

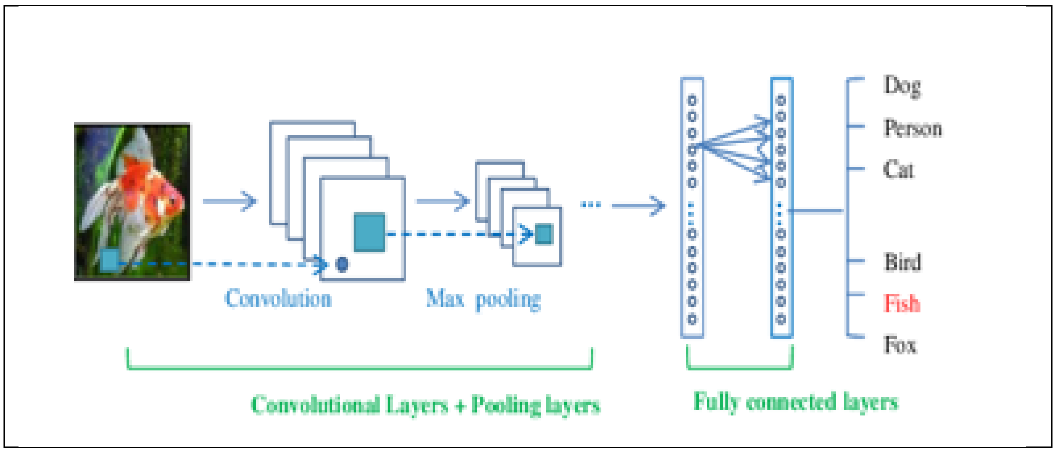


Figure 3.5. CNN layers [61]

Filters are determined by Back Propagation Through Time (BPTT) in the training phase of supervised learning. In CNN, after getting the filters we have a function to classify images. On the other hand, we need to find a function which classifies images with the least error given the input values and output values. In BPTT, derivative of the function is calculated and weights are updated at each iteration according to a selected loss function [62]. In training phase, labelled input data are fed to network in batches, batch size in convolution layer refers to how many input data are used at each step with back propagation. After feeding all the input data in batches to the network, a cycle is completed. Each cycle is called an epoch, epoch size is an important parameter for deep learning. It determines how many times all the input data is used for training.

CNN is mainly used for image processing tasks such as image classification, object recognition, etc. In recent years, CNN has been also used for NLP tasks like sentiment analysis and has shown state of the art results. In this study, CNN is one of the used DL algorithms for sentiment analysis.

As mentioned above and can be seen in the Figure 3.2, convolution needs numbers to process. On the other hand, sentiment analysis deals with sentences as a NLP task. So, first need before using CNN is to convert words to their corresponding vectoral representations which is called as word embedding. Details of word embeddings can be seen in Section 3.4. Later, word vectors can be concatenated to represent sentence (tweet) matrices and these matrices are fed to CNN as input data like images.

In a CNN model for sentiment analysis, there exist one or more convolution layers as the first layer. Then, output of convolution with different filters are sent to a pooling layer. In the literature, it has seen that max pooling best fits for pooling in sentiment analysis task. Another option is to use mean pooling rather than max pooling. After pooling operation, neural networks with some classifier are used for final classification (Figure 2.1).

3.2. LONG SHORT-TERM MEMORY NETWORKS

Long short-term memory networks (LSTM) is a special type of recurrent neural networks (RNN). Therefore, it is a worth to mention about RNN before going into LSTM details. RNN is a deep learning which is used for tasks in which an input with time series is given and next item is predicted. Most common usage areas of RNNs change in a variety of tasks from weather forecast to exchange rates prediction. At the same time, RNN has been one of the most important deep learning models used in natural language processing (NLP) tasks in recent years. It shows competitive results because of its structure dealing with sequential information. In a RNN, output of each node is input to the next item. RNN not only deals with last step it also takes into consideration information in a series of previous steps. RNNs are good at predicting next item, they can be even used for complicated art tasks such as writing poems, composing music, painting, etc. Figure 3.6 shows the standard RNN structure.

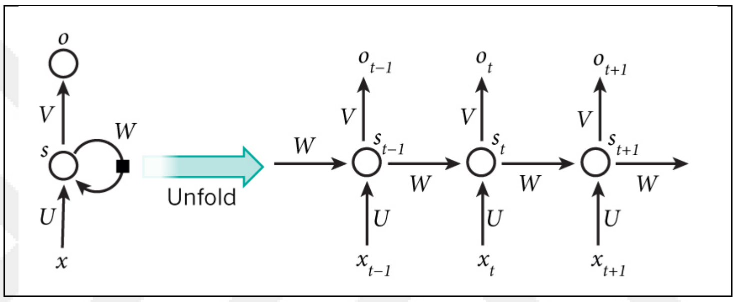


Figure 3.6. RNN

Recurrent neural networks learn with BPTT like convolutional neural networks. Therefore, it has an important problem named vanishing gradient problem. Vanishing gradient problem is a difficulty found in training with gradient based methods. In particular, this problem makes it really hard to learn and tune the parameters of the earlier layers in the network. Gradient based methods learn a parameter's value by understanding how a small change in the parameter's value will affect the network's output. If a change in the parameter's value causes very small change in the network's output, the network just can't learn the parameter effectively.

At this time, LSTM helps for a solution to this problem. It solves vanishing gradient problem through usage of gates which are forget gate, input gate, and output gate. Each cell in LSTM is responsible for keeping track of the dependencies between the elements in the input sequence. Forget gate decides which information should be kept or thrown away from the cell state. Forget gate uses a sigmoid function, information from the previous hidden state and information from the current input are input to this sigmoid function. Function outputs a value between zero and one, information with closer values to one are kept and values near zero are thrown away. Input gate is responsible for deciding what new information is going to be stored in the cell state. It consists of two functions to update the cell state. First, previous hidden state and current input are input to a sigmoid function that decides which values will be updated by transforming the values to be between zero and one where one refers to the importance of the input. Second function is a tanh function to squish values between minus one and one to help regulate the network. Then the tanh output with the sigmoid output are multiplied. The sigmoid output will decide which information is important to keep from the tanh output. Third and the last gate is output gate. Output gate decides the next hidden state. Hidden state contains information on previous inputs and it is used for predictions. This hidden is state is used as input to the next time step. LSTM is much better at capturing long-term dependencies with the help of gates. The memory in LSTMs are called cells and these cells take as input the previous state h_{t-1} and current input x_t. Internally these cells decide what to keep in memory. Finally, the previous state, the current memory, and the input are combined.

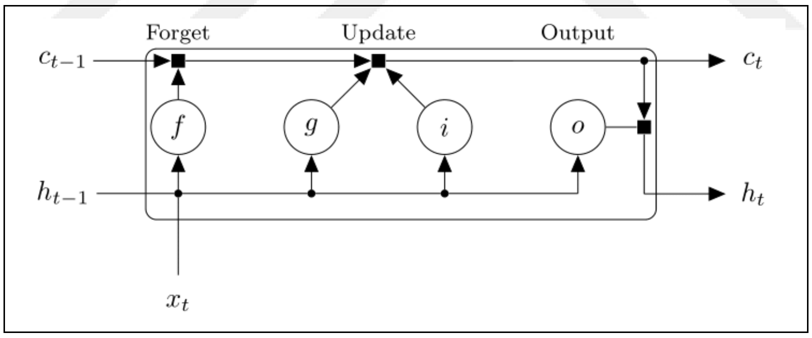


Figure 3.7. LSTM and gates

A special version of LSTM, Gated Recurrent Unit (GRU), is introduced in 2014 . It combines the forget and input gates into a single gate named update gate. It also merges the cell state and hidden state. It has no output gate. It is used instead of standard LSTM in some works.

3.3. SUPPORT VECTOR MACHINES

SVM is another supervised learning algorithm that has been used for classification and regression tasks. Since sentiment analysis task is also a classification task, SVM is used in the literature and shows competitive results. SVM is based on separating data into the groups by drawing a border among items in the plane. The idea is to find out the most distant boundary, called hyperplane, to all of the groups so that new coming items can be separated into one of the groups more easily (Figure 3.8). Hyperplane is a line in a two dimensional classification task. A plane is used as hyperplane in a three dimensional input space (Figure 3.9). SVM is not a deep learning model since it consist of a single layer, it does not contain hidden layers. Thus, SVM shows significant accuracy with less computing cost.

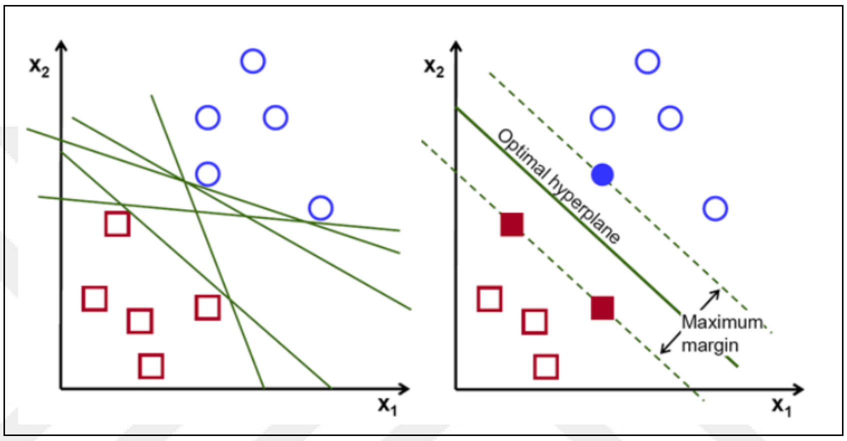


Figure 3.8. SVM and possible hyperplanes

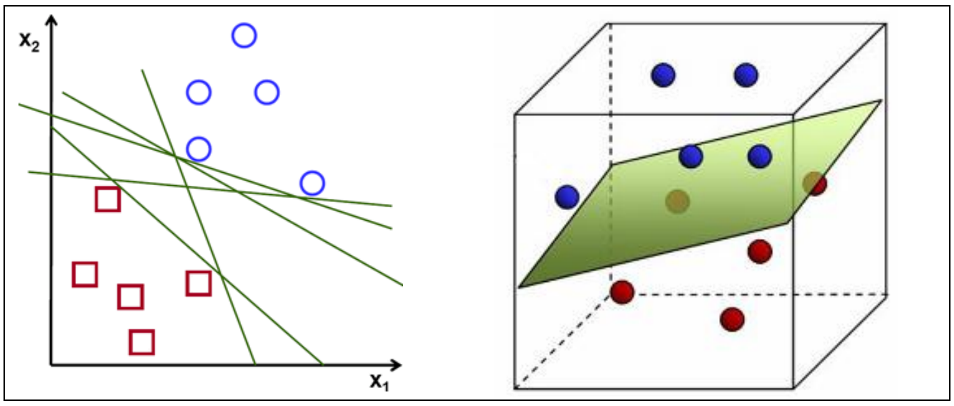


Figure 3.9. Hyperplanes related to input dimensions

First input to the SVM is word embeddings as expected. In addition to word embeddings, a lot of different parameters can be used in sentiment analysis with SVM. Some of the commonly used example inputs in the literature are; n-gram features, negation features, number of hashtags, number of emoticons, TF-IDF features, number of capitalized words, number of negative words, number of positive words, etc.

3.4. WORD EMBEDDINGS

In sentiment analysis task, we have tweets and their sentiments for classification. Tweets consist of text data, but we need matrices consisting of numbers for calculation in Deep Learning methods. So we need to convert these tweets somehow to their corresponding matrix representations. One of the best practices for this task is to use word embeddings, that is to detect vectoral representations of words. Words are represented in such a way that, for example, vector operations vector('Paris') - vector('France') + vector('Turkey') results in a vector that is very close to vector('Ankara'), and vector('king') - vector('man') + vector('woman') is close to vector('queen')(Figure 3.10).

Word embedding is an unsupervised learning model, very large amount of text corpus is used for training. Words are trained against other words that neighbour them in the input corpus. It is done in one of two ways, either using context to predict a target word (a method known as continuous bag of words, or CBOW), or using a word to predict a target context, which is called skip-gram.

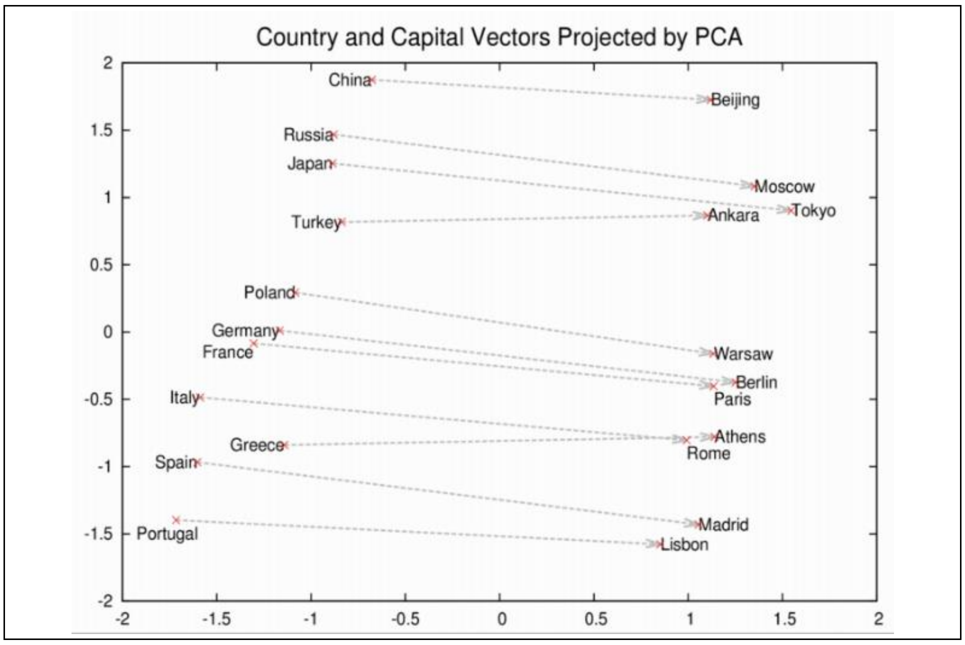


Figure 3.10. Country and capital vectors

3.4.1. Word2Vec

Word2Vec is maybe the most popular word embedding worldwide, developed by Google. Google provides a lot of pretrained word embedding data set with different number of dimensions, corpus size, vocabulary size, etc. (Table 3.1). In addition, there are a lot of libraries provided for different programming languages. You can easily create your own word embeddings by providing your own corpus as input text and setting parameters for training. It can use one of two algorithms for learning; Continuous Bag Of Words (CBOW) and Continuous Skip-gram. CBOW is generally used to to predict a target word using context. On the other hand, continuous skip-gram model is used to predict a target context using a word. Word2Vec.Net library is one of the libraries provided for .Net framework. It provides creation of word2vec vectors by setting the following training options;

- WithTrainFile: text data to train the model
- WithOutputFile: path to save the resulting word vectors / word clusters
- WithSize: size of word vectors; default is 100
- WithSaveVocubFile: file path to save the vocabulary file
- WithDebug: debug level (default is two, more info during training)
- WithBinary: whether to save the resulting vectors in binary mode; default is zero (off)
- WithCBow: whether to use the continuous bag of words model; default is one, zero is used for skip-gram model
- WithAlpha: starting learning rate; default is 0.025 for skip-gram and 0.05 for CBOW
- WithWindow: max skip length between words; default is 5
- WithSample: threshold for occurrence of words. Those that appear with higher frequency in the training data are randomly down-sampled; default is 1e-3
- WithHs: whether to use hierarchical softmax; default is zero (not used)
- WithNegative: number of negative examples; default is five, common values are between three and ten. (zero means not used)
- WithThreads: number of threads (default 12)
- WithIter: number of training iterations (default is five), number of epochs.
- WithMinCount: number to discard words that appear less than, default is five which means numbers occurring less than five times are ignored

- WithClasses: output word classes rather than word vectors; default number of classes is zero (vectors are written)

Table 3.1. Word2vec pretrained sample vectors

Model file	**Number of dimensions**	**Corpus (size)**	**Vocabulary size**	**Author**	**Architecture**
Google News	300	Google News (100B)	3M	Google	word2vec
Freebase IDs	1000	Gooogle News (100B)	1.4M	Google	word2vec, skip-gram
Freebase names	1000	Gooogle News (100B)	1.4M	Google	word2vec, skip-gram
Wikipedia dependency	300	Wikipedia (?)	174,015	Levy & Goldberg	word2vec modified
DBPedia vectors (wiki2vec)	1000	Wikipedia (?)	?	Idio	word2vec

3.4.2. GloVe

Global Vectors for Word Representation (GloVe) is another most widely used word embedding in sentiment analysis task. It is developed by Stanford NLP Group and they provide researchers with a variety of pretrained word vectors publicly distributed. Pretrained vectors for GloVe are shown in Table 3.2

3.4.3. SentiWordNet

SentiWordNet is a lexical resource for word vector representation. Unlike word2vec and GloVe, SentiWordNet does not have large number of dimensions for word vectors. It represents each word only with a two-dimensional vector which are positivity score and negativity. These scores have values between zero and one. In addition, objectivity score is calculated by subtracting the sum of positivity score and negativity score from one. Finally, words are represented by three-dimensional sentiment vectors.

Table 3.2. GloVe pretrained sample vectors

Model file	**Number of dimensions**	**Corpus (size)**	**Vocabulary size**	**Author**	**Architecture**
Wikipedia+ Gigaword 5	50	Wikipedia+ Gigaword 5 (6B)	400,000	GloVe	GloVe
Wikipedia+ Gigaword 5	100	Wikipedia+ Gigaword 5 (6B)	400,000	GloVe	GloVe
Wikipedia+ Gigaword 5	200	Wikipedia+ Gigaword 5 (6B)	400,000	GloVe	GloVe
Wikipedia+ Gigaword 5	300	Wikipedia+ Gigaword 5 (6B)	400,000	GloVe	GloVe
Common Crawl 42B	300	Common Crawl (42B)	1.9M	GloVe	GloVe
Common Crawl 840B	300	Common Crawl (840B)	2.2M	GloVe	GloVe
Twitter (2B Tweets)	25	Twitter (27B)	1.2M	GloVe	GloVe
Twitter (2B Tweets)	50	Twitter (27B)	?	GloVe	GloVe
Twitter (2B Tweets)	100	Twitter (27B)	1.2M	GloVe	GloVe
Twitter (2B Tweets)	200	Twitter (27B)	1.2M	GloVe	GloVe

4. ANALYSIS AND DESIGN

In this research, we have compared different word embeddings, different learning algorithms individually and with different combinations and ensembles. We have compared the results in terms of accuracy, training time, recall, precision, and f-measure. Overall system consist of downloading and pre-processing tweets, word embedding, and applying deep learning algorithms. We have applied two different word embeddings to eight different deep learning configurations. In addition, we have used two different data organization types which are standard concatenation of words in a tweet and applying regional structure. In total, we have 32 different results for comparison.

4.1. TWEETS

4.1.1. Creating the Tweet Corpus

Since learning methods we use, namely CNN, LSTM, and SVM, are supervised learning algorithms, we first need labeled data for training. SemEval committee provides labeled tweets for development, training, and testing. On the other hand, SemEval cannot directly publish these tweets because of Twitter's privacy laws. Twitter do not let distribution of deleted tweets in terms of privacy rules. Therefore, SemEval only provides status id of tweets and sentiment values for the given tweets. Later, those tweets are downloaded from Twitter servers via different APIs provided by Twitter company. In our thesis, we have created a .Net application with LinqToTwitter API provided by Twitter. Three different datasets were collected for training data which were SemEval2017 – Task 4 development data plus previous years' full data for the same task, namely SemEval2016 – Task 4 full data, SemEval2014 - Task 9 - Sub Task B full data are downloaded for training. We were able to download a total of 31007 tweets for training which consist of 662.000 total words with a vocabulary of around 10.000 words.

4.1.2. Data Cleaning

After downloading tweets, next step was to process the tweets in order to increase the system’s performance during training. A lot of tweets need to be cleaned for getting better performance in training. Therefore, we have done a few cleaning operations over downloaded tweets. Firstly, whole tweets are converted to lowercase letters to get rid of case sensitivity because it does not affect the sentiment of a tweet very much. Later, some characters such as single quotes (‘), double quotes (”), @ character, # character of hashtags are removed from text. Smileys are replaced, :) and :-) symbols are replaced with <smile> tag, :(and :-(symbols are replaced with <sadface> tag. Links are replaced with <url> tag because links do not have contribution in the sentiment.

4.2. WORD EMBEDDING

At the end of tweet cleaning operations, clean tweets are ready for training different learning models. One more operation is needed to be able to train models which is to get vectoral representations of words in tweets. In this stage, two different options for word embedding are used. First option is to create word vectors via tools like word2vec, GloVe, etc. Second option is to download pretrained word vectors (word2vec, GloVe, SentiWordNet). In this research, both alternatives are tried and reported for comparison.

4.2.1. Self-trained Word Vectors

For creating self-trained word vectors, word2vec model is used. Input data were all tweets provided for SemEval 2014 and SemEval 2016 competitions. In addition, SemEval 2017 development tweets were used for training word2vec. Word2vec model is configured for output 25-dimensional word vectors. 25-dimensional vectors rather than larger-dimensions are preferred because of performance issues for training in terms of time with a computing environment without GPU support. Other options were, setting output in text format rather than binary file, continuous bag of words model is used, max skip length between words is set to 5, with 100 training iterations, discarding words that appear less than 5 times.

4.2.2. Pre-trained Word Vectors

GloVe pre-trained word vectors were utilized as a second alternative way. Again, 25-dimensional word vectors are used for comparing both self-trained and pre-trained vectors under the same circumstances. GloVe vectors were downloaded from Stanford University NLP group web page. Word vectors are created from twitter data with a corpus size 27B which means word embeddings are created from data with 27 billion words. 2 billion tweets were used to create GloVe vectors.

Because of large corpus size and vocabulary, pre-trained GloVe vectors scored better than self-trained word2vec vectors up to seven percent although word2vec vectors were created from SemEval data.

4.2.3. Normalizing Word Vectors

After creating word vectors, each value in 25-dimensional vector is normalized to get a value between 0 and 1. Property value is calculated by equation:

$$propVal' = \frac{propVal - minVal}{maxVal - minVal} \tag{4.1}$$

where minVal is minimum value for selected property, maxVal is maximum value for selected property, and propVal is current item’s value for selected property. Some sample word vectors before and after normalization is shown in the below tables.

Table 4.1. Word vectors before normalization

Word	Vectoral Representation
yesterday	[-0.718 0.297 1.058 -0.736 -0.321 -0.413 1.271 -0.633 -1.044 0.197 -0.071 0.838 -4.207 0.608 1.010 -0.178 0.545 -0.439 -0.615 -0.554 -0.287 -0.242 0.220 0.217 0.177]
today	[-0.834 0.021 -0.056 -0.915 -0.218 -0.147 1.662 -0.577 -0.374 -0.180 -0.387 0.574 -5.073 0.881 0.892 -0.359 0.128 -0.836 -0.663 -0.400 -0.532 -0.576 0.022 0.452 0.167]
tomorrow	[-0.729 0.670 0.590 -1.258 -0.491 -0.515 1.896 -0.775 -0.355 -0.071 -0.928 0.116 -4.646 0.829 1.356 -0.494 0.091 -1.258 -0.964 -0.168 -0.798 -0.441 0.513 0.885 0.734]
man	[0.370 -0.396 -0.022 -0.630 -0.319 0.343 0.110 0.488 -0.487 0.368 -0.392 0.254 -4.928 0.068 0.371 0.368 1.166 0.092 -0.877 -0.746 0.409 1.567 -0.239 0.248 0.764]
woman	[-0.925 -0.339 -0.321 0.147 0.523 -0.153 1.217 -0.224 -0.087 0.285 0.185 0.428 -4.398 0.262 -0.430 -0.001 0.669 -0.081 0.563 -0.375 1.179 0.771 -0.528 -1.259 -0.451]
king	[-0.745 -0.120 0.373 0.368 -0.447 -0.229 0.701 0.829 0.395 -0.583 0.415 0.371 -3.691 -0.201 0.115 -0.347 0.362 0.096 -0.018 0.685 -0.049 0.540 -0.210 -0.654 0.646]
queen	[-1.127 -0.521 0.456 0.211 -0.051 -0.652 1.140 0.699 -0.206 -0.718 -0.028 0.110 -3.309 -0.493 -0.514 0.104 -0.118 -0.085 0.026 0.686 -0.292 0.459 -0.400 -0.404 0.318]
uncle	[-1.276 0.917 1.099 0.489 -0.700 -0.615 0.608 0.571 -0.220 -0.128 -0.227 0.698 -3.295 -0.665 0.783 -0.437 1.450 -0.611 -0.567 0.001 0.998 0.791 -0.643 -0.703 -0.544]
aunt	[-1.955 0.941 1.322 0.589 -0.483 -0.927 0.671 -0.268 -0.388 -0.476 -0.515 0.114 -3.305 -0.602 0.071 -0.394 1.474 -0.367 -0.487 0.217 1.302 0.280 -0.748 -0.097 -0.484]

Table 4.2. Word vectors after normalization

Word	Vectoral Representation
yesterday	[0.507 0.501 0.587 0.444 0.441 0.478 0.640 0.475 0.473 0.564 0.501 0.516 0.166 0.555 0.573 0.553 0.494 0.465 0.394 0.510 0.468 0.448 0.441 0.558 0.525]
today	[0.499 0.482 0.517 0.433 0.447 0.495 0.666 0.479 0.514 0.540 0.483 0.498 0.099 0.571 0.565 0.543 0.469 0.442 0.391 0.520 0.451 0.426 0.428 0.574 0.524]
tomorrow	[0.506 0.526 0.558 0.413 0.431 0.471 0.682 0.465 0.515 0.547 0.451 0.466 0.132 0.568 0.598 0.535 0.467 0.417 0.372 0.535 0.431 0.435 0.459 0.603 0.560]
man	[0.581 0.453 0.519 0.451 0.441 0.528 0.561 0.558 0.507 0.575 0.482 0.476 0.110 0.521 0.528 0.584 0.531 0.497 0.377 0.497 0.518 0.562 0.412 0.560 0.561]
woman	[0.493 0.457 0.500 0.497 0.492 0.495 0.636 0.505 0.532 0.570 0.516 0.488 0.151 0.533 0.471 0.563 0.502 0.486 0.470 0.522 0.574 0.512 0.394 0.457 0.486]
king	[0.505 0.472 0.544 0.511 0.433 0.490 0.601 0.583 0.561 0.514 0.530 0.484 0.207 0.505 0.510 0.544 0.483 0.497 0.433 0.591 0.485 0.497 0.414 0.499 0.554]
queen	[0.479 0.445 0.549 0.501 0.457 0.461 0.631 0.573 0.524 0.506 0.504 0.466 0.237 0.487 0.465 0.569 0.455 0.486 0.435 0.591 0.468 0.492 0.402 0.516 0.534]
uncle	[0.469 0.543 0.590 0.518 0.418 0.464 0.595 0.564 0.523 0.543 0.492 0.506 0.238 0.477 0.557 0.538 0.548 0.455 0.397 0.546 0.561 0.513 0.387 0.495 0.480]
aunt	[0.423 0.545 0.604 0.524 0.431 0.443 0.599 0.502 0.513 0.521 0.475 0.466 0.237 0.480 0.507 0.541 0.550 0.470 0.402 0.561 0.583 0.481 0.380 0.536 0.484]

4.2.4. Creating Sentence Vectors

Sentence vectors are created by concatenating word vectors belonging to the words in a tweet. Since convolution needs the same size input and all tweets do not have the same length, we need to equalize the size of the tweets. In literature, different alternatives are applied for getting equal size tweets. Most common way is to feed zeros to the end of short tweets until getting the same length. Another common option is to feed randomly generated values. We have tried a new way, which is to repeat the word vectors until we get the same length (Algorithm 4.1). We set max word length to 40. We decided this number according to the length of the tweets in our training dataset. If a tweet has more than 40 words, first 40 words are taken into consideration. For the tweets which have less than 40 words, we repeat the words in the same order they appear in text. For instance the tweet “my parents are going to the zac brown band concert tomorrow night and im so jealous.” has 16 words. Words in this tweet are repeated twice and first eight words are repeated three times. Final representation of tweet becomes “my parents are going to the zac brown band concert tomorrow night and ım so jealous. my parents are going to the zac brown band concert tomorrow night and ım so jealous. my parents are going to the zac brown” so it has finally 40 words.

After having all equal size tweets input data is created by concatenating vectoral representations of words. The words which do not exist in our vocabulary forms another problem. We use zero-padding for such words, that is, we add 25 zeros for these words.

Final vectors are saved to a file after adding sentiment value to each line for processing in deep learning methods. These vectors are 40 times 25 dimensional vectors. So, we have 1000-dimensional vectors for sentence representation plus one sentiment value for each line in the training file (Algorithm 4.1).

Algorithm 4.1. Creating sentence vectors with no region

```
maxNumberOfWords = 40
wordsArray = SplitTweetIntoWords()
while wordsCount < maxNumberOfWords do
    foreach word ∈ wordsArray do
        AddWordToWordList(wordList, word)
        wordsCount = wordsCount + 1
    end for
    foreach word ∈ wordList do
        wordVector = GetWordVectorFromDatabase(word)
        if wordVector is empty then
            for i from 1 to 25
                wordVector[i] = 0
            end for
        end
        sentenceVector = sentenceVector + wordVector
    end for
end
```

4.2.5. Creating Sentence Vectors with Regions

In addition to standard CNN, LSTM, and SVM, different combinations of these algorithms are used in sentiment analysis. One alternative is to use regional structure for combination as shown in Figure 2.1. In this structure, each sentence is split into regions. We use punctuation notations such as full stop, comma, colon, semi-colon, question mark, exclamation mark for splitting regions. Each region is set to a standard length of 10 words, we apply the same rules used in the sentence vectors with no regions for padding and non-existing words. In addition, each tweet is split into eight regions. If a tweet has less than eight regions, regions are repeated in the same way we apply for the word length. If a tweet has more than eight regions, first eight regions are used for training (Algorithm 4.2). Finally we have 2000-dimensional vectors by concatenating eight regions in a sentence, 10 words in a region, and 25 dimensional vectors for each word (Figure 4.1). For non-regional data organization, the same structure is applied by accepting each single word as a region (Figure 4.2). Additionally, sentiment value is appended to the end of training file’s each line.

Algorithm 4.2. Creating sentence vectors with regional structure

```
maxNumberOfRegions = 8
maxNumberOfWords = 10
regionsArray = SplitTweetIntoRegions()
while regionsCount < maxNumberOfRegions do
    foreach region ∈ regionsArray do
        AddRegionToRegionList(regionList, region)
        regionsCount = regionsCount + 1
    end for
    foreach region ∈ regionList do
        wordsArray = SplitRegionIntoWords()
        while wordsCount < maxNumberOfWords do
            foreach word ∈ wordsArray do
                AddWordToWordList(wordList, word)
                wordsCount = wordsCount + 1
            end for
            foreach word ∈ wordList do
                wordVector = GetWordVectorFromDatabase(word)
                if wordVector is empty then
                    for i from 1 to 25
                        wordVector[i] = 0
                    end for
                end
                sentenceVector = sentenceVector + wordVector
            end for
        end
    end for
end
```

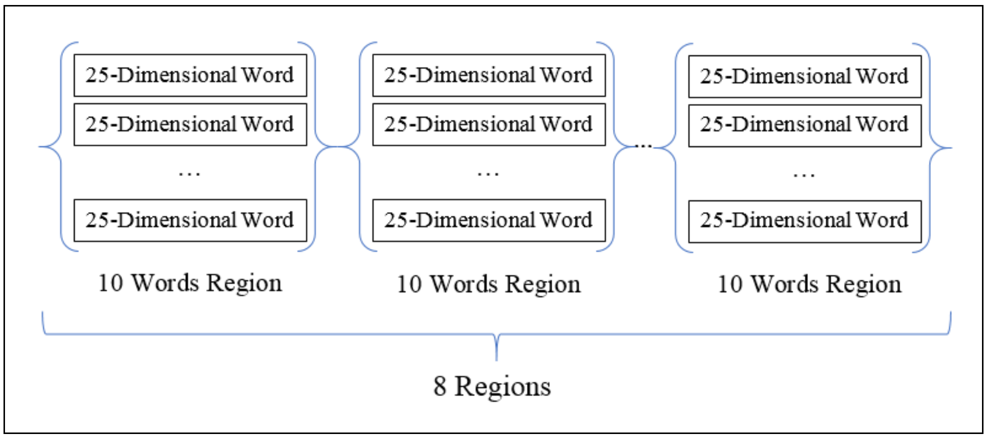


Figure 4.1. Regional structure for one tweet with regional data

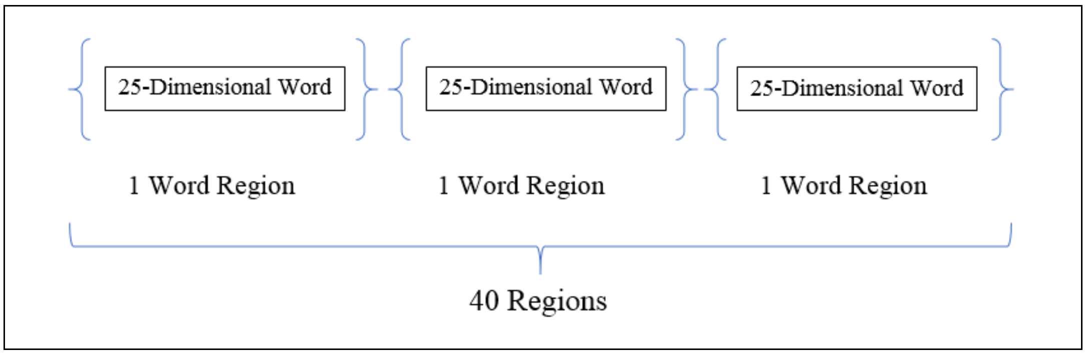


Figure 4.2. Regional structure for one tweet with non-regional data

4.3. NEURAL NETWORKS

Sentence vectors created in the previous step are used as input for training and testing issues. SemEval2016 Task4 full data, SemEval2014 Task 9 Sub Task B full data, and SemEval2017 development data are used for training. SemEval2017 test data are used for testing which have 12284 tweets in total. In this research, widely used learning models such as CNN and LSTM are compared. In addition, some alternative ensembles and combinations of these models are used for comparison. Moreover, SVM is applied as an alternative solution in one combination.

4.3.1. Single CNN Network

Single CNN model is a deep learning model consisting of a convolution layer, a max pooling layer, a flatten layer, and a fully connected (dense) layer with a sigmoid activation function. Convolution layer takes input as 1000 dimensional vector which is a concatenation of 40 words, each word is represented by 25 dimensional vectors. It has 12 filters with a kernel size of 50. Pool size is set to three in max pooling layer. Flatten layer is applied for making the output of max pooling layer usable for fully connected layer. Since we have 12 filters, output of the max pooling for each tweet is 317 x 12 matrix. Flatten layer concatenates these values to a single vector. Finally, fully connected layer outputs three dimensional vector which represent the probabilities of each class. Sigmoid activation function and mean square error (mse) loss function is used for training. The same model is applied for regional word embeddings, only dimensions change. Implementation details of configuration and parameters for CNN are shown in Figure 4.3 and Table 4.3. The reason behind regional model having less dimension after first convolution is that stride is set to 25 in the first convolutional layer for being able to deal with large dimensional input.

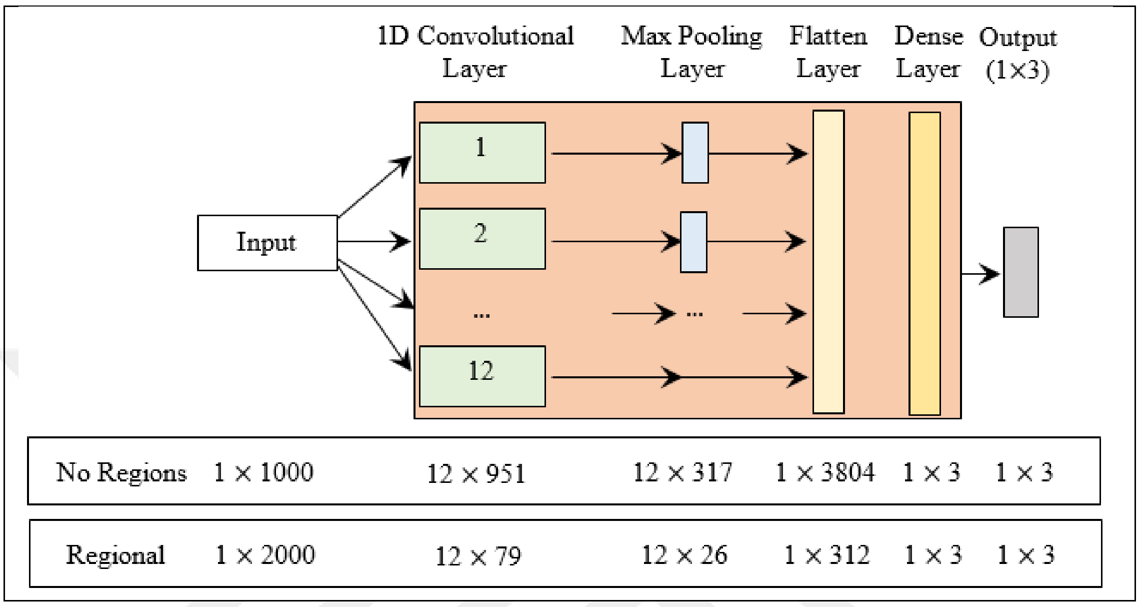

1D Convolutional Layer
Max Pooling Layer
Flatten Layer
Dense Layer
Output (1×3)
Input
1
2
...
12
No Regions 1 × 1000 12 × 951 12 × 317 1 × 3804 1 × 3 1 × 3
Regional 1 × 2000 12 × 79 12 × 26 1 × 312 1 × 3 1 × 3

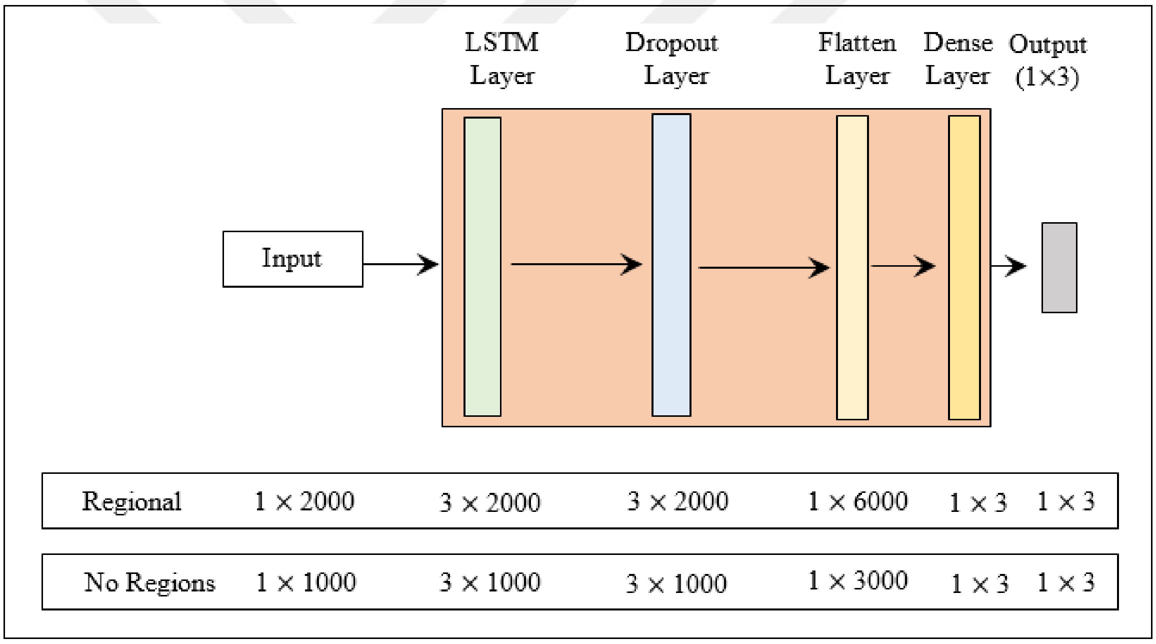
Figure 4.3. Single CNN network

Table 4.3. CNN parameters for single CNN network

PARAMETER	VALUE (NO REGIONS)	VALUE (REGIONAL)
Number of Layers	1 Convolution 1 Max pooling 1 Fully connected	1 Convolution 1 Max pooling 1 Fully connected
Batch size	100	100
Epochs	300	300
Number of tweets	31007	31007
Word count	40	80
Word Vector Dimensions	25	25
Number of filters	12	12
Kernel size	50	50
Stride number	1	25
Pool size	3	3
Activation function	sigmoid	sigmoid
Loss function	MSE	MSE
Optimizer	rmsprop	rmsprop
Metrics	accuracy	accuracy
Validation split	0.05	0.05

4.3.2. Single LSTM Network

LSTM model uses the same input with the CNN model and some parameters such as activation and loss functions are the same. But the layer outputs are totally different from the one in CNN model described in previous section because of the differences between CNN and LSTM structure. LSTM model has an LSTM layer, a dropout layer, a flatten layer, and a fully connected layer. In LSTM layer, output dimension is set to three. A dropout of 20 percent is applied. Flatten layer and fully connected layer are used in the same way with CNN model.



Regional	1 × 2000	3 × 2000	3 × 2000	1 × 6000	1 × 3	1 × 3
No Regions	1 × 1000	3 × 1000	3 × 1000	1 × 3000	1 × 3	1 × 3

Figure 4.4. Single LSTM network

Table 4.4. LSTM parameters for single LSTM network

PARAMETER	**VALUE (NO REGIONS)**	**VALUE (REGIONAL)**
Number of Layers	1 LSTM 1 Dropout 1 Fully connected	1 LSTM 1 Dropout 1 Fully connected
Batch size	100	100
Epochs	300	300
Number of tweets	31007	31007
Word count	40	80
Word Vector Dimensions	25	25
Dropout	0.2	0.2
Activation function	sigmoid	sigmoid
Loss function	MSE	MSE
Optimizer	rmsprop	rmsprop
Metrics	accuracy	accuracy
Validation Split	0.05	0.05

4.3.3. Individual CNN and LSTM Networks with a SVM Classifier

SVM model used in this research is not a single SVM layer. In this configuration, single CNN network and single LSTM network described in the previous sections are used with the parameters described in Table 4.3, Table 4.4, Figure 4.3, and Figure 4.4.Outputs of the fully connected layers of CNN model and LSTM model are concatenated and input to the SVM for training. Therefore, SVM model structure is an ensemble of CNN model and LSTM model described in the previous sections. SVM model has an input of six dimensional vector and it outputs three dimensional vector. Aim of this configuration was to see whether SVM could be used as a more intelligent algorithm rather than just voting results of LSTM and CNN layers. Therefore, SVM model is applied with default values described in sklearn [63] library. The default values of SVM described in sklearn library are; penalty parameter C of the error term is one, kernel type to be used in the algorithm is rbf, kernel coefficient for algorithm is set to auto which uses one / number of features, shrinking heuristic is used, probability estimates are set to false, verbose output option is set to false.

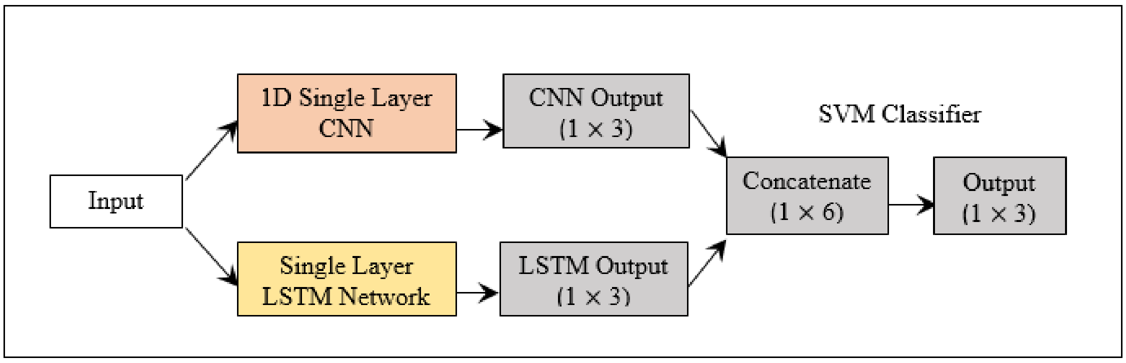


Figure 4.5. Individual CNN and LSTM networks with a SVM classifier

4.3.4. Individual CNN and LSTM Networks

Input to this model is provided exactly in the same way with SVM model described in previous section. The only difference is that instead of feeding LSTM and SVM results to an SVM layer, a soft-voting is used for final prediction that means average of each class are calculated to decide corresponding sentiment value.

Although models mentioned so far do not use regional structure, it is also used with data provided for regional structure. When we use regional data, some parameters in the model change for best accuracy. In addition, layer outputs have different shapes.

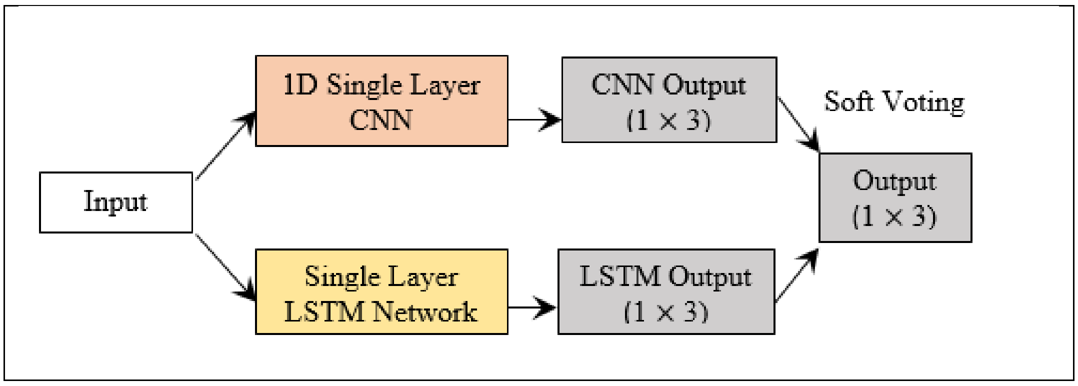


Figure 4.6. Individual CNN and LSTM networks

4.3.5. Multiple CNNs and LSTM Networks

In this structure (Figure 2.5), tweets are divided into regions. Each region is a sub sentence in a tweet which is divided by punctuation marks such as full stop, comma, colon, semi-colon, question mark, exclamation mark. Each tweet is divided into eight regions and each region is set to a maximum number of 10 words.

Firstly, a CNN model is applied to each of eight regions. CNN model has a convolution layer, a max pooling layer, a flatten layer, and a fully connected (dense) layer with a sigmoid activation function. Convolution layer takes input as 250 dimensional vector which is a concatenation of 10 words, each word is represented by 25 dimensional vectors. It has 12 filters with a kernel size of three. Pool size is set to three in max pooling layer. Since we have 12 filters, output of the max pooling for each region is 2 x 12 matrix. Flatten layer concatenates these values to a single vector. Finally, fully connected layer outputs three dimensional vector which represent the probabilities of each class. Sigmoid activation function and mean square error (mse) loss function are used for training.

Second step is to feed CNN model outputs to LSTM model and get the final predictions. Since we have eight regions, LSTM model input is eight times three dimensional vectors. LSTM model has an LSTM layer, a dropout layer, a flatten layer, and a fully connected layer. In LSTM layer, output dimension is set to three. A dropout of 30 percent is applied. Flatten layer and fully connected layer are used in the same way with CNN model.

Although this model is mainly used for regional structure, it is also used with data provided without regions. In this scenario, each word in the tweet is accepted as a region for simplicity. Therefore, region size in a tweet becomes 40 and each region consists of 25 dimensional word vector. Network structure is shown in Figure 4.7 for both scenarios. CNN parameters and LSTM parameters are shown in detail in Table 4.5 and Table 4.6.

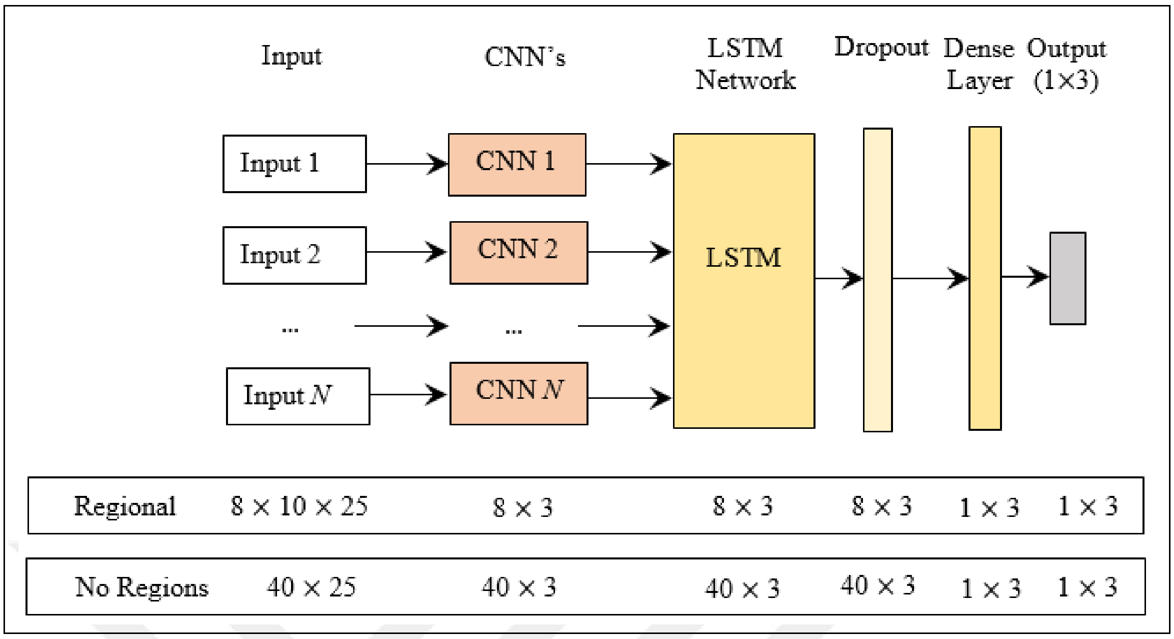


Figure 4.7. Multiple CNNs and LSTM networks

Table 4.5. CNN parameters for multiple CNNs and LSTM networks

PARAMETER	VALUE (NO REGIONS)	VALUE (REGIONAL)
Number of CNNs	40	8
Number of Layers	1 Convolution 1 Max pooling 1 Fully connected	1 Convolution 1 Max pooling 1 Fully connected
Batch size	100	100
Epochs	300	300
Number of tweets	31007	31007
Input Dimensions	1 x 25	10 x 25
Number of filters	12	12
Kernel size	3	3
Stride number	1	1
Pool size	3	3
Activation function	sigmoid	sigmoid
Loss function	MSE	MSE
Optimizer	rmsprop	rmsprop
Metrics	accuracy	accuracy
Validation split	0.05	0.05

Table 4.6. LSTM parameters for multiple CNNs and LSTM networks

PARAMETER	VALUE (NO REGIONS)	VALUE (REGIONAL)
Number of Layers	1 LSTM 1 Dropout 1 Fully connected	1 LSTM 1 Dropout 1 Fully connected
Batch size	100	100
Epochs	300	300
Number of tweets	31007	31007
Input Dimensions	40 x 3	8 x 3
Dropout	0.3	0.3
Activation function	sigmoid	sigmoid
Loss function	MSE	MSE
Optimizer	rmsprop	rmsprop
Metrics	accuracy	accuracy
Validation Split	0.05	0.05

4.3.6. Multiple CNNs and Bidirectional LSTM Networks

This model is the same as the one in the previous section. Difference is the usage of bidirectional LSTM layer instead of LSTM layer. When bidirectional LSTM is used, output of LSTM layer differs because LSTMs in each direction yields three dimensional vectors and a total of six dimensional vector is created. Difference can be seen in Figure 4.8. Aim of this configuration is to compare the impact of using bidirectional LSTM instead of single directional LSTM because some works in the literature have shown competitive results with bidirectional LSTM. Parameters are the same as stated in Table 4.5 and Table 4.6.

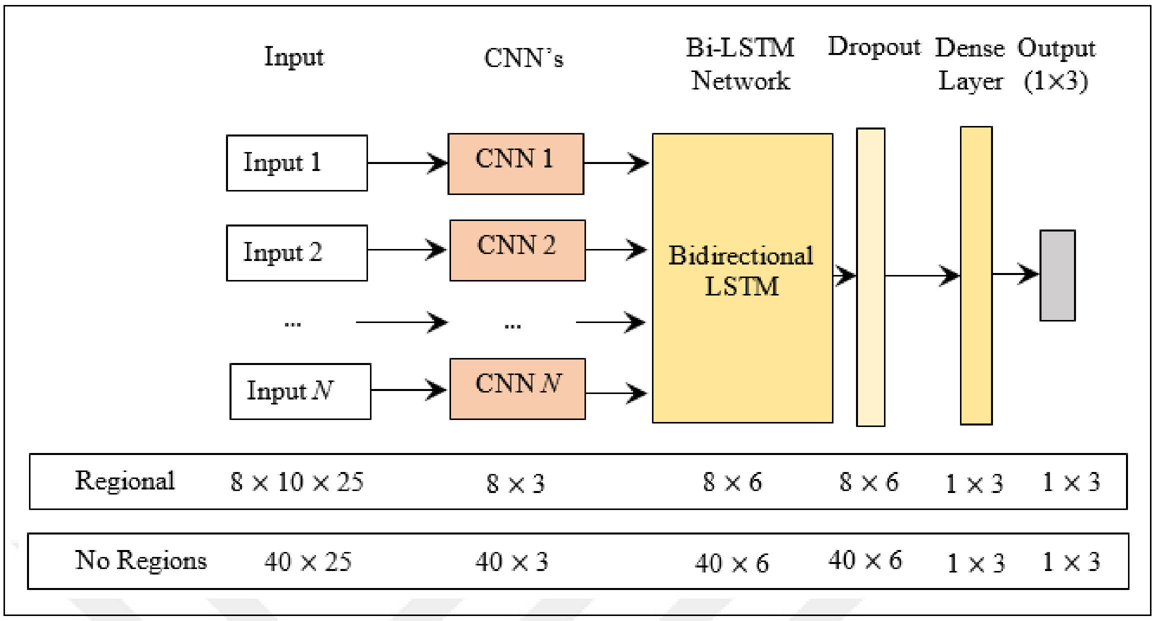


Figure 4.8. Multiple CNNs and bidirectional LSTM networks

4.3.7. Single Three-layer CNN and LSTM Networks

This model consists of three convolution layers, each convolution is followed by a max pooling layer with a pool size of three. After three CNN and max pooling layers, LSTM, dropout, and fully connected layers are applied to decide final prediction. Configuration is displayed in

Figure 4.9 where the input is directed to a three-layer CNN. The input has a size of 1000 if it is based on words (non-regional) or a size of 2000 if it is based on regions (regional). CNN parameters are shown in Table 4.7 and LSTM parameters are the same as described in Table 4.4.

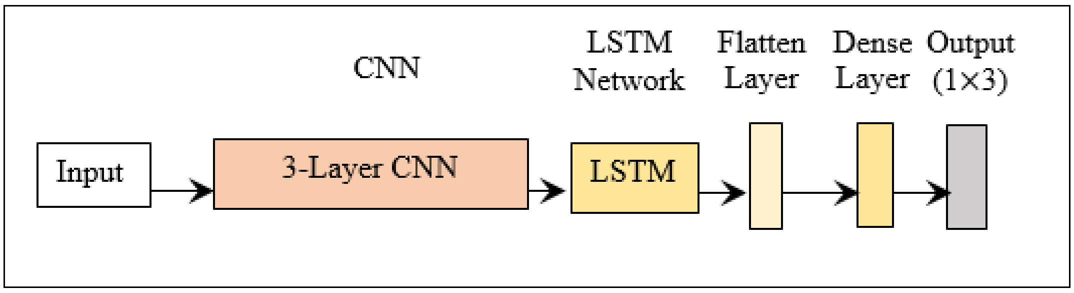


Figure 4.9. Single three-layer CNN and LSTM networks

Table 4.7. CNN parameters for single three-layer CNN and LSTM networks

PARAMETER	VALUE (NO REGIONS)	VALUE (REGIONAL)
Number of Layers	3 Convolution 3 Max pooling	3 Convolution 3 Max pooling
Batch size	100	100
Epochs	300	300
Number of tweets	31007	31007
Word count	40	80
Word Vector Dimensions	25	25
Number of filters	12	12
Kernel size	50	50
Stride number	1	1
Pool size	3	3
Activation function	sigmoid	sigmoid
Loss function	MSE	MSE
Optimizer	rmsprop	rmsprop
Metrics	accuracy	accuracy
Validation split	0.05	0.05

4.3.8. Single Three-layer CNN and Bidirectional LSTM Networks

This model is the same as the model described in the previous section. Only difference is the application of bi-directional LSTM layer instead of one directional LSTM. Motivation behind this configuration is to compare the impact of using bidirectional LSTM instead of single directional LSTM because some works in the literature have shown competitive results with bidirectional LSTM. CNN parameters and LSTM parameters are the same as described in Table 4.7 and Table 4.4.

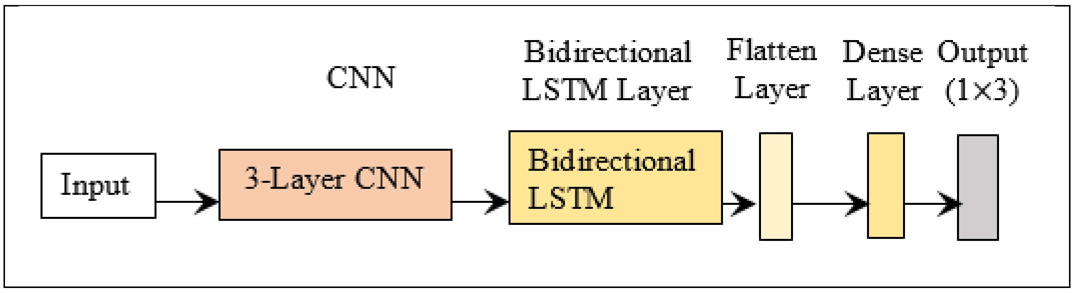
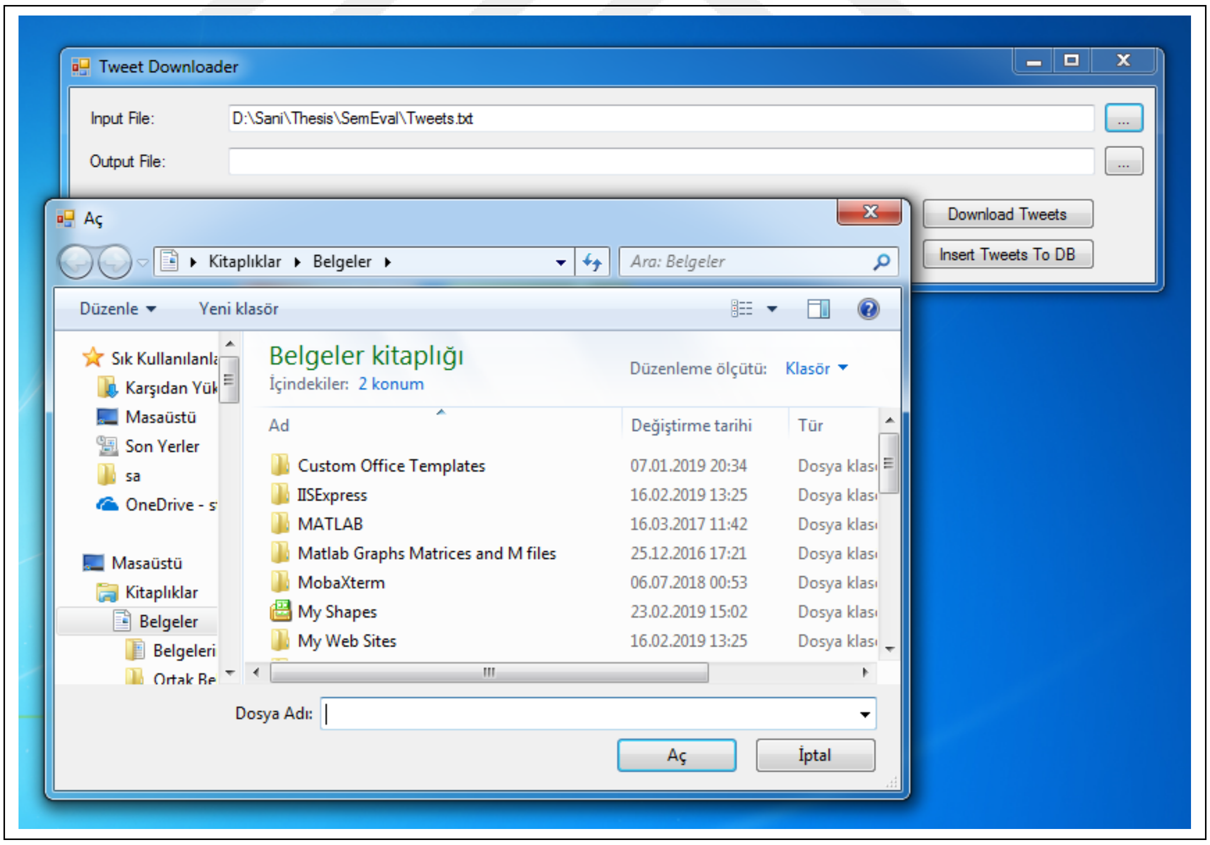


Figure 4.10. Single three-layer CNN and bidirectional LSTM networks

5. IMPLEMENTATION

This research can be split into two steps. First step is about arranging input data and second step is deep learning itself. In each step, different tools and technologies are used for implementation.

First step of sentiment analysis is to download data and convert tweets into numerical values. We need to download tweets for converting them to numerical values. SemEval committee does not directly distribute tweets because of Twitter privacy laws that forbids sharing of removed tweets. They provide us with only status ids of tweets with a text file. Tweets are downloaded from Twitter databases via C# LinqToTwitter.net library, which is a publicly available library for tweet operations like reading, sharing, editing tweets. Downloaded tweets are inserted to an Sql Server database, and tweets are cleaned by removing links, replacing smileys, etc mentioned in Section 4.2.



Input File: D:\Sani\Thesis\SemEval\Tweets.txt

Output File:

Ad	Değiştirme tarihi	Tür
Custom Office Templates	07.01.2019 20:34	Dosya klas
IISExpress	16.02.2019 13:25	Dosya klas
MATLAB	16.03.2017 11:42	Dosya klas
Matlab Graphs Matrices and M files	25.12.2016 17:21	Dosya klas
MobaXterm	06.07.2018 00:53	Dosya klas
My Shapes	23.02.2019 15:02	Dosya klas
My Web Sites	16.02.2019 13:25	Dosya klas

Dosya Adı:

Figure 5.1. Tweet download application

Word vectors are needed to represent whole tweet in numerical values. We have created a windows application with .Net framework and C# programming language to create word2vec word vectors. Word vectors are created from SemEval data with open source C# word2vec library named word2vec.net. We have configured library to create 25 dimensional word vectors using continuous bag of words model, discarding words those occur less than 5 times, and repeating 100 iterations for training. The file containing SemEval 2014 tweets, SemEval 2016 tweets, and SemEval 2017 development tweets is set as training file for word2vec model and word vectors are inserted to the database (Figure 5.2) after unsupervised training with the selected file. Inserted word vectors are normalized with another .Net application written with C# programming language.

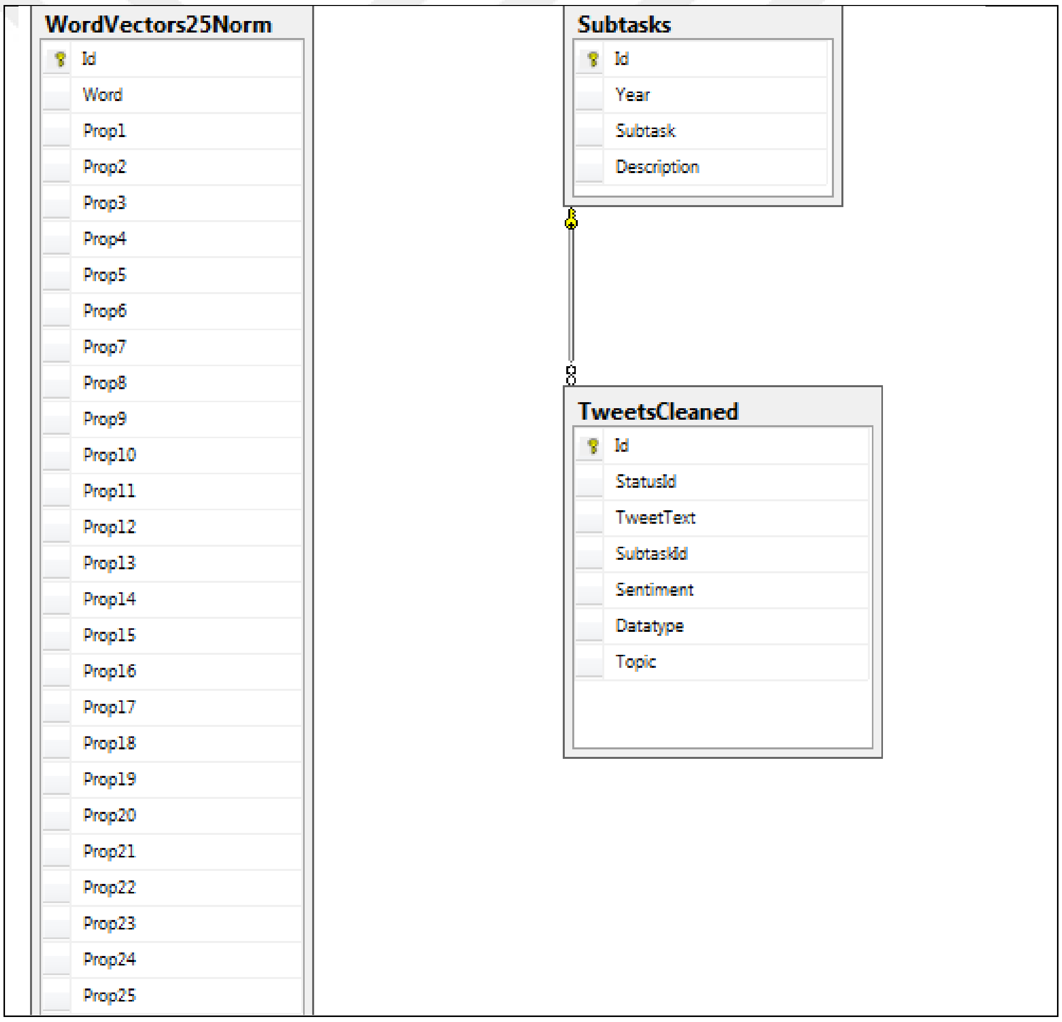


Figure 5.2. Relational database diagram for tweets and normalized word vectors

Finally, sentence vectors are created from word vectors. In single models, word vectors are simply concatenated to create sentence vectors. In regional structure, tweets are splitted into regions, later word vectors in each region are concatenated and later regions are concatenated to form sentence vectors. This application is also written with C# and .Net framework. This application exports sentence vectors to text file with System.IO library in .Net Framework.

Second step is the development and training of deep learning algorithms. In this phase, formatted text data files exported from .Net application are used as input to learning models. Learning models are created with Python. Pandas library is used for reading input files. Tensorflow [62] framework and Keras [64] deep learning libraries are used for designing deep learning models in Python. In addition, sklearn [63] library is used for SVM model.

6. TEST AND EVALUATION

This chapter compares the differences among different learning models such as CNN, LSTM, SVM, and their combinations in both classic way or regional structure. In addition, performance of pretrained GloVe vectors and self-trained Word2Vec vectors from SemEval data are compared.

6.1. COMPARISON OF DESCRIBED DEEP LEARNING CONFIGURATIONS

To begin with the testing and evaluation process, first mention about the data used for training and testing. Tweets published on Twitter have been changing continuously. For example, if you want to get tweets containing any hashtag, word, or topic, you can get different set of tweets each time you try because new tweets may be published, tweets may be updated, or previously published tweets may be removed. Of course, this change is proportional to popularity of the subject and keywords you search. Therefore, we have decided to work with a publicly distributed data set provided by SemEval committee. Even though we have worked with a static data set, we couldn't prevent deletion of some tweets. SemEval 2017 was dealing with two languages, Arabic and English with different subtasks. We have used data for Task 4 Subtask A which is about considering English tweets with positive, negative, and neutral sentiment values. Most of the top scoring teams use deep learning models such as CNN, LSTM, SVM, and their combinations in SemEval 2017 [25].

The overall accuracy for the evaluation of testing set and the computational efficiency is calculated by the equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

Other metrics used for test and evaluation are precision, recall, and f-measure values.

Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actually positive. Precision is ratio of truly predicted positive to all positive predictions. (Equation 6.2)

Recall actually calculates how many of the actual positives our model capture through labelling it as positive (Equation 6.3)

F-measure (F-score or F1 score) is the balance between precision and recall values. It is calculated by harmonic average of the precision and recall with the Equation 6.4.

For each class precision, recall, and F-measure are calculated by below equations where:

TP: instance is classified as a, belongs to a

FP: instance is classified as a, belongs to others

TN: instance is classified as others, belongs to others

FN: instance is classified as others, belongs to a

$$Precision = TP = \frac{TP}{TP + FP} \tag{6.2}$$

$$Recall = TP\ Rate = \frac{TP}{TP + FN} \tag{6.3}$$

$$F\_Measure = \frac{2 * Recall * Precision}{Recall + PRecision} \tag{6.4}$$

Let a, b, and c represent positive, negative, and neutral classes respectively. After calculating recall, precision, and F-measure for each class individually, we take average of recall, precision, and F-measure respectively to find values for the entire model. Results are shown in Table 6.2 and Table 6.3.

According to our evaluation criteria, we started with comparing different word vectors.

First, we tried different deep learning models with word2vec vectors. Word vectors are created from previous years' SemEval data tweets plus SemEval 2017 training data. 31007 tweets are used for training word vectors. Tweets consist of 662K words and it has a vocabulary of size 10K.

Second option was to use GloVe vectors. GloVe vectors are created by Stanford NLP Group and published freely. Word embeddings are created from two billion tweets with a corpus size of 27 billion words and 1.2 million words in vocabulary. Comparison of word vectors is shown at (Table 6.1).

With word2vec model, maximum accuracy of 52 percent was reached with two similar models which are combination of CNN and single LSTM or bi-directional LSTM with regional and non-regional structure. On the other hand, models with GloVe vectors scored to a maximum of 59 percent in case of using CNN and LSTM organically in multiple CNN and LSTM configuration described in section 4.3.5 and 4.3.6. The reason behind it lies to the fact that with Word2Vec model is trained with a relatively small training dataset, around 32.000 tweets are too little compared to the pretrained GloVe word vectors' training dataset with a larger corpus.

Table 6.1. Comparison of word embeddings

Model file	**Number of dimensions**	**Corpus (size)**	**Vocabulary size**	**Author**	**Architecture**
Twitter (2B Tweets)	25	Twitter (27B)	1.2M	GloVe	GloVe
Twitter (33K Tweets)	25	Twitter (662K)	10K	Generated via word2vec.Net library	Word2vec

After comparing different word vectors, our main research of interest is comparison of different learning models. According to our test results, best scorers in terms of accuracy were the ones in which CNN and LSTM models are used. Usage of multiple CNNs with LSTM networks increases the performance between three and six percent regardless of used word embedding system. This system almost always scores the best among all the versions with both one directional and bidirectional LSTMs. Another observation is that, arranging data in a regional structure does not always improve the performance, especially in the tests done with GloVe vectors. Soft voting after individual CNN and LSTM layers scores better than using an SVM layer for these results. Final observation is that bidirectional LSTM networks does not provide much better results than one directional standard LSTM models, which can be because of the structure of words in a sentence.

Table 6.2. Test results with word2vec self-trained vectors

Embedding Word System: Word2Vec					
Network Model	**Type**	**Recall**	**Prec.**	**F1**	**Acc.**
1. Single CNN network	N-R[a]	0.33	0.35	0.33	0.49
	R	0.32	0.34	0.33	0.51
2. Single LSTM network	N-R	0.43	0.51	0.39	0.51
	R	0.44	0.49	0.39	0.50
3. Individual CNN and LSTM Networks	N-R	0.43	0.47	0.37	0.50
	R	0.46	0.52	0.42	0.52
4. Individual CNN and LSTM Networks with SVM classifier	N-R	0.45	0.46	0.43	0.49
	R	0.42	0.54	0.38	0.51
5. Single 3-Layer CNN and LSTM Networks	N-R	0.41	0.52	0.40	0.46
	R	0.40	0.46	0.35	0.48
6. Multiple CNNs and LSTM Networks	N-R	0.43	0.47	0.37	0.50
	R	0.46	0.52	0.43	0.52
7. Single 3-Layer CNN and bi-LSTM Networks	N-R	0.42	0.45	0.39	0.48
	R	0.42	0.47	0.36	0.48
8. Multiple CNNs and bi-LSTM Networks	N-R	0.43	0.50	0.38	0.51
	R	0.46	0.51	0.44	0.52

[a] N-R: Non-Regional, R: Regional word embedding

Table 6.3. Test results with GloVe pretrained vectors

Embedding Word System: GloVe					
Network Model	**Type**	**Recall**	**Prec.**	**F1**	**Acc.**
1. Single CNN network	N-R[a]	0.44	0.41	0.4	0.54
	R	0.35	0.31	0.31	0.48
2. Single LSTM network	N-R	0.5	0.58	0.48	0.55
	R	0.51	0.55	0.51	0.55
3. Individual CNN and LSTM Networks	N-R	0.53	0.6	0.53	0.58
	R	0.55	0.6	0.55	0.56
4. Individual CNN and LSTM Networks with SVM classifier	N-R	0.52	0.55	0.53	0.56
	R	0.49	0.6	0.5	0.56
5. Single 3-Layer CNN and LSTM Networks	N-R	0.5	0.5	0.5	0.52
	R	0.43	0.61	0.39	0.53
6. Multiple CNNs and LSTM Network	N-R	0.53	0.60	0.53	0.58
	R	0.55	0.6	0.56	0.59
7. Single 3-Layer CNN and bi-LSTM Network	N-R	0.52	0.59	0.53	0.57
	R	0.50	0.57	0.50	0.55
8. Multiple CNNs and bi-LSTM Network	N-R	0.54	0.60	0.55	0.59
	R	0.55	0.6	0.56	0.59

[a] N-R: Non-Regional, R: Regional word embedding

6.2. COMPARISON OF BEST SCORING COMBINATION WITH THE STATE OF THE ART METHODS

In the second comparison, we compare our best scoring network with the state of the art models. Results are displayed in Table 6.4. It is obviously seen our study has similar but a bit lower, up to six percent, scores. This difference stem from usage of different datasets and specialized methods used in other studies for reshaping dataset or tuning the neural network. Moreover, the scope of this study was not focused on achieving the best performance in comparison with other studies, but rather to evaluate and compare different deep neural networks and word embedding systems on a single testing environment with an accepted dataset. At this point, it is worth to mention that the best performance in the literature in terms of accuracy (~65 percent) it is still not satisfactory, thus revealing that on sentiment analysis deep learning methods are still far from guaranteeing a performance comparable to other fields where the same networks are used with higher success rate (e.g. deep learning networks for object recognition in images).

Table 6.4. Comparison of the state of the art methods with the best results of current study

Study	**Network System**	**Word Embedding**	**Dataset (labeled Tweets)**	**Acc.**
Baziotis et al. [30]	bi-LSTM	GloVe	~50.000	0.65
Cliche [52]	CNN+LSTM	GloVe, FastText,Word2Vec	~50.000	0.65
Deriu et al. [16]	CNN	GloVe, Word2Vec	~300.000	0.65
Rouvier and Favre [24]	CNN	Lexical, POS, Sentiment	~20.000	0.61
Wange et al. [1]	CNN+LSTM	Regional Word2Vec	~8.500	1.341[a]
Current study	CNN+LSTM	Regional GloVe	~31.000	0.59

[a] RMSE

Cliche [52] has tried different deep learning methods which are CNN and LSTM. This model is similar to the one we have proposed in Section 4.3.4. They perform better because they have done a lot of improvements starting from the number of CNN and LSTM models. We have only used one CNN and one LSTM for soft-voting. In CNN, they use 200 dimensional word vectors. They set the tweet size to have 80 words. 200 filters are applied with three different kernel sizes. So, a total of 600 filters are applied. After that a hidden layer is used with a size of 30. To reduce overfitting 50 percent dropout is applied after both max pooling and fully connected layers. They use cross-entropy as the loss function, loss is minimized using the Adam optimizer. They use bi-directional LSTM. In terms of data improvements, 100 million unlabeled tweets are downloaded in addition to 50000 labeled tweets, and these data are labeled according to smileys, tweets with smiling face ":)" are accepted positive, ones with sad face ":(" are accepted negative. They train Word2vec, GloVe, and FastText vectors with default settings from 100 million unlabeled tweets. They ensemble 10 CNNs and 10 LSTMs together through soft voting. The models ensembled have different random weight initializations, different number of epochs (from 4 to 20 in total), different set of filter sizes (either [1, 2, 3], [3, 4, 5] or [5, 6, 7]) and different embeddings. Since this configuration is similar to our individual CNN and LSTM model which has scored lower than our multiple CNNs and LSTM configuration, data collection and pre-processing, different number of filters, and soft voting mechanism can be applied to our best scoring configuration to get better results for the future work.

6.3. CLASS BASED PREDICTION COMPARISON

In Table 6.5 and Table 6.6 accuracies for each class labels are shown according to Equation 6.1. From the results, we can easily state that positive classes can be predicted most easily in our experiments regardless of word embedding technique and neural network configuration. Second best predicted classes are negative classes. Our learning algorithms have difficulties in predicting neutral sentiment values. This may be because of unequal distribution of classes in our data set. As it is shown in Table 6.7, almost half of the tweets have neutral sentiment in both training and test data. In addition, training data have a short number of negative tweets and test data have very little positive tweet set.

Table 6.5. Test results for class accuracies with word2vec self-trained vectors

Embedding Word System: Word2Vec				
Network Model	**Type**	**Positive Acc.**	**Negative Acc.**	**Neutral Acc.**
1. Single CNN network	N-R[a]	0.83	0.47	0.39
	R	0.85	0.46	0.38
2. Single LSTM network	N-R	0.79	0.68	0.54
	R	0.78	0.68	0.54
3. Individual CNN and LSTM Networks	N-R	0.85	0.68	0.61
	R	0.86	0.68	0.61
4. Individual CNN and LSTM Networks with SVM classifier	N-R	0.75	0.67	0.55
	R	0.82	0.68	0.53
5. Single 3-Layer CNN and LSTM Networks	N-R	0.74	0.66	0.52
	R	0.83	0.69	0.53
6. Multiple CNNs and LSTM Networks	N-R	0.79	0.68	0.53
	R	0.8	0.68	0.56
7. Single 3-Layer CNN and bi-LSTM Networks	N-R	0.77	0.67	0.53
	R	0.76	0.68	0.53
8. Multiple CNNs and bi-LSTM Networks	N-R	0.81	0.68	0.53
	R	0.8	0.68	0.56

[a] N-R: Non-Regional, R: Regional word embedding

Table 6.6. Test results for class accuracies with GloVe self-trained vectors

Embedding Word System: GloVe				
Network Model	**Type**	**Positive Acc.**	**Negative Acc.**	**Neutral Acc.**
1. Single CNN network	N-R[a]	0.84	0.51	0.42
	R	0.82	0.46	0.37
2. Single LSTM network	N-R	0.82	0.7	0.57
	R	0.8	0.71	0.58
3. Individual CNN and LSTM Networks	N-R	0.89	0.71	0.67
	R	0.88	0.7	0.67
4. Individual CNN and LSTM Networks with SVM classifier	N-R	0.82	0.71	0.59
	R	0.83	0.71	0.57
5. Single 3-Layer CNN and LSTM Networks	N-R	0.76	0.71	0.57
	R	0.83	0.69	0.53
6. Multiple CNNs and LSTM Networks	N-R	0.84	0.72	0.6
	R	0.83	0.73	0.61
7. Single 3-Layer CNN and bi-LSTM Networks	N-R	0.83	0.72	0.59
	R	0.82	0.71	0.57
8. Multiple CNNs and bi-LSTM Networks	N-R	0.83	0.73	0.61
	R	0.83	0.73	0.61

[a] N-R: Non-Regional, R: Regional word embedding

Table 6.7. Sentiment distribution in tweet data set

Data Type	**Positive Tweets**	**Negative Tweets**	**Neutral Tweets**	**Total**
Train	11469	4966	14572	31007
Test	2375	3972	5937	12284

If we take into consideration our best scoring model for further analysis we prefer to continue with multiple CNNs and LSTM network with GloVe vectors, and we select non regional data organization because it has the same score with regional data set, but it requires less computing power because it has smaller input data dimensions. According to confusion matrix (Table 6.8), we can see that most errors are done by predicting positive and negative as neutral. In addition least errors are confusion of positive values with negative values. Precision, recall, and f-measure values for each class are reported in Table 6.9.

Table 6.8. Confusion matrix for multiple CNN and LSTM network

Confusion Matrix		**Actual Classes**		
		Positive	**Negative**	**Neutral**
Predicted Classes	Positive	1100	139	594
	Negative	89	1228	583
	Neutral	1186	2605	4760

Table 6.9. Comparison of class scores for multiple CNN and LSTM network

	Recall	**Precision**	**F-Measure**
Positive	0.46	0.60	0.52
Negative	0.31	0.65	0.42
Neutral	0.80	0.56	0.66

6.4. CROSS VALIDATION RESULTS COMPARISON

Since our data is tiny compared to state of the art models, we have tried our selected deep learning configuration with k-fold cross validation. Training and test data described in the previous sections, with a total of 43290 records, are concatenated and used together. In terms of k we applied two different values, 10-fold cross validation and 5-fold cross validation because these two k values are most commonly used k values in the literature [65]. Our selected model was multiple CNNs and LSTM networks with no region. Cross validation takes long time because we are training the model five or ten times according to k value, moreover, we have 40 CNNs and one LSTM model in each iteration. In order to reduce training time, we have set number of CNN epoches to 100 instead of 300 because our aim was to compare cross validation results with the system's performance using test data rather than increasing the accuracy. For equality of training parameters we have also tried our model with 100 CNN epoches and we have observed only one percent decrease so it is not so much important compared to time we earn by decreasing epoch number. In this experiment, we have observed that cross validation score with bot 10-fold and 5-fold validation are similar to results we have obtained for training and testing data independently. In addition, we have tried with two different batch sizes, 100 and 1000. We have observed that increasing batch size to 1000 causes a five to eight percent decrease in accuracy. Another interesting observation is that, 10-fold cross validation performs better in 5-fold with 1000 batch size. On the other hand, 5-fold cross validation outperforms 10-fold validation with 100 batch size. Therefore, it is not possible to say 10-fold cross validation is better than 5-fold validation, or vice versa although we use the same data set, it depends on the parameters used.

Table 6.10. Cross validation results comparison

	Accuracies	**Average Accuracy**	**Configuration**
Multiple CNNs and LSTM Networks With No Region	0.58	0.58	CNN Epoches:300 LSTM Epoches: 300 Batch size: 100
Multiple CNNs and LSTM Networks With No Region	0.57	0.57	CNN Epoches:100 LSTM Epoches: 300 Batch size: 100
5-Fold Cross Validation	[0.50 0.59 0.59 0.57 0.55]	0.56	CNN Epoches:100 LSTM Epoches: 300 Batch size: 100
10-Fold Cross Validation	[0.54 0.51 0.56 0.59 0.59 0.57 0.55 0.60 0.57 0.57]	0.57	CNN Epoches:100 LSTM Epoches: 300 Batch size: 100
5-Fold Cross Validation	[0.48 0.54 0.54 0.49 0.48]	0.51	CNN Epoches:100 LSTM Epoches: 300 Batch size: 1000
10-Fold Cross Validation	[0.45 0.46 0.56 0.51 0.46 0.51 0.50 0.52 0.49 0.48]	0.49	CNN Epoches:100 LSTM Epoches: 300 Batch size: 1000

6.5. COMPARISON WITH LARGER DIMENSIONAL WORD VECTORS

In these experiment sets, we have applied our selected network configuration with different size of word embeddings such as 100 dimensional word vectors and 200 dimensional word vectors. Pre-trained GloVe word embeddings from Stanford university are downloaded for this task (Table **3.2**). All the embeddings are created from the same dataset with a vocabulary size of 1.2 million words and 27 billion corpus size. So, all vectors are compared under the same circumstances.

First, we have tested the network with the same parameters we had previously tried with our best scoring model as described in Table 6.11. Within these implementations, we have observed that 100 dimensional vectors scored the best accuracy with 61 percent and this is the maximum accuracy we have reached so far. Since 200 dimensional vectors score less than 100 dimensional vectors, we cannot conclude that increasing the vector size is positively related with accuracy. The aim should be finding optimum vector size for best accuracy.

Table 6.11. Larger dimensional vectors with same parameters

PARAMETER	**25D VECTORS**	**100D VECTORS**	**200D VECTORS**
Number of CNNs	40	40	40
Batch size	100	100	100
Epochs	300	300	300
Number of tweets	31007	31007	31007
Input Dimensions	1 x 25	1 x 100	1 x 200
Number of filters	12	12	12
Kernel size	3	3	3
Stride number	1	1	1
Pool size	3	3	3
Activation function	sigmoid	sigmoid	sigmoid
Loss function	MSE	MSE	MSE
Optimizer	rmsprop	rmsprop	rmsprop
Metrics	accuracy	accuracy	accuracy
Validation split	0.05	0.05	0.05
Accuracy	0.58	0.61	0.60

In the second experiment set, we have tried 100 dimensional and 200 dimensional word vectors with different parameters such number of filters, kernel size, and pooling size. In these experiments, we have again observed superiority of 100 dimensional vectors over 200 dimensional vectors in all parameter sets. Another observation is that, our best scoring parameters which were decided before with 25 dimensional vectors have again scored best results with both 100 and 200 dimensional vectors. In addition, increasing or decreasing any of these parameters is not directly related with accuracy. We can only see their effect with trial and error in accordance with other parameters, as observed for number of dimensions in word embeddings.

Table 6.12. Larger dimensional vectors with different parameters

PARAMETER	**100D VECTORS**	**200D VECTORS**	**100D VECTORS**	**200D VECTORS**	**100D VECTORS**
Number of CNNs	40	40	40	40	40
Batch size	100	100	100	100	100
Epochs	300	300	300	300	300
Number of tweets	31007	31007	31007	31007	31007
Input Dimensions	1 x 100	1 x 200	1 x 100	1 x 200	1 x 100
Number of filters	48	48	48	48	12
Kernel size	12	12	12	12	3
Stride number	1	1	1	1	1
Pool size	12	12	3	3	12
Activation function	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid
Loss function	MSE	MSE	MSE	MSE	MSE
Optimizer	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop
Metrics	accuracy	accuracy	accuracy	accuracy	accuracy
Validation split	0.05	0.05	0.05	0.05	0.05
Accuracy	0.60	0.57	0.59	0.46	0.57

6.6. LSTM AND GRU COMPARISON

Another comparison was about the GRU and LSTM models. We have implemented GRU models alone and compared with LSTM only model. The same parameters described in Table 4.4 are used for these implementations. In addition, we have applied GRU instead of LSTM in multiple CNNs and LSTM models. Parameter details are shown in Table 4.5 and

Table 4.6. They have output similar results in our tests (Table 6.13), but GRU only model performed one percent better than LSTM only model with both Glove and word2vec embeddings. On the other hand, they have shown the same accuracy when used with multiple CNN model. Therefore, it is not easy to say one is more preferable than the other according to our test results.

Table 6.13. LSTM and GRU comparison

	Pretrained GloVe vectors				**Self-trained word2vec vectors**			
Model	LSTM Only	GRU Only	CNN LSTM	CNN GRU	LSTM Only	GRU Only	CNN LSTM	CNN GRU
Accuracy	0.55	0.56	0.59	0.59	0.51	0.52	0.52	0.52
Recall	0.5	0.54	0.55	0.55	0.43	0.43	0.46	0.46
Precision	0.58	0.56	0.6	0.6	0.51	0.51	0.52	0.52
F-Measure	0.48	0.54	0.56	0.56	0.39	0.41	0.43	0.43

7. CONCLUSION

In this thesis, different configurations of deep learning methods with CNN, LSTM, SVM, and their combinations and ensembles are tested for sentiment analysis with Twitter data provided by SemEval organization. These tests proved the importance of data for deep learning tasks by showing better results with pretrained word vectors from a larger corpus. Test results showed similar results with state of the art methods but with lower values. With the help of convolution's effective dimensionality reduction process and long and short term dependency detection of LSTM networks, combination of these two models scored better than their individual scores. All in all, performance contribution of different dataset has contributed much more than changing learning models used in the networks. We have observed this different in both using different word embeddings and comparing our best networks with state of the art models. Therefore, it is more valuable than concentrating on getting better datasets for future work. In order to having a better training and test datasets, larger dimensional word vectors such as having 100, 300 dimensions are used and we have seen best results with 100 dimensional word vectors. SentiWordNet datasets can be used in another neural network and results can be combined with word embedding results, thus making use of different word embeddings. More CNN, LSTM models with different parameters can be added to networks throughout Grid Search.

To summarize, the contribution of this thesis is that it allowed to evaluate different deep neural network configurations and principal word embedding systems under a single dataset and evaluation framework allowing to shed more light on their advantages and limitations.

REFERENCES

1. Wang J, Yu LC, Lai KR, Zhang X. Dimensional sentiment analysis using a regional CNN-LSTM model. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*; 2016.

2. Linq provider for the twitter api (twitter library) [cited 2018 16 July]. Available from: https://github.com/JoeMayo/LinqToTwitter.

3. Lee L, Pang B. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval.* 2008;1(2):1–135.

4. Fukushima K. Neocognition: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*. 1980;202(36):193–202.

5. Lecun Y, Haffner P, Bottou L, Bengio Y. Object recognition with gradient-based learning. *Shape, Contour and Grouping in Computer Vision*. 1999;1:319-345.

6. Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in The Microstructure of Cognition.* 1985;1:318-362.

7. Understanding LSTM networks [cited 2019 24 February]. Available from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

8. Kim Y. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*; 2014.

9. Dos Santos CN, Gatti M. Deep convolutional neural networks for sentiment analysis of short texts. *Proceedings of COLING 2014, the 25th International Conference on*

Computational Linguistics: Technical Papers; 2014.

10. Palogiannidi E. Tweester at SemEval-2016 task 4: Sentiment analysis in twitter using semantic-affective model adaptation. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

11. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*; 2013.

12. Zhang L, Wang S, Liu B. Deep learning for sentiment analysis: A Survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. 2018;8(4): 253-287.

13. Pennington J, Socher R, Manning C. Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*; 2014.

14. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*. 2016;5(1):135–146.

15. Hamdan H. Senti17 at SemEval-2017 task 4: Ten convolutional neural network voters for tweet polarity classification. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

16. Deriu J, Gonzenbach M, Uzdilli F, Lucchi A, De Luca V, Jaggi M. SwissCheese at SemEval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

17. Yin Y, Yangqiu S, Zhang M. NNEMBs at SemEval-2017 task 4: Neural twitter sentiment classification: A simple ensemble method with different embeddings. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

18. Kim Y. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*; 2014.

19. Lei T. Semi-supervised question retrieval with gated convolutions. *Proceedings of NAACL-HLT 2016*; 2016.

20. Yoon J, Lyu K, Kim H. Adullam at SemEval-2017 task 4: Sentiment analyzer using lexicon integrated convolutional neural networks with attention. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

21. Yin R, Li P, Wang B. Sentiment lexical-augmented convolutional neural networks for sentiment analysis. *Proceedings - 2017 IEEE 2nd International Conference on Data Science in Cyberspace*; 2017: IEEE.

22. Hao Y, Lan Y, Li Y, Li C. XJSA at SemEval-2017 task 4: A deep system for sentiment classification in twitter. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

23. Maoquan W, Shiyun C, Yufei X, Lu Z. EICA at SemEval-2017 task 4: A simple convolutional neural network for topic-based sentiment classification. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

24. Rouvier M, Favre B. SENSEI-LIF at SemEval-2016 task 4: Polarity embedding fusion for robust sentiment analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

25. Briones G, Amarasinghe K, McInnes BT. SemEval-2016 task 4: Sentiment analysis in twitter. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

26. Gao H, Oates T. MDSENT at at SemEval-2016 Task 4: A supervised system for message polarity classification. *Proceedings of the 10th International Workshop on Semantic*

Evaluation (SemEval-2016); 2016.

27. You Q, Luo J, Jin H, Yang J. Robust image sentiment analysis using progressively trained and domain transferred deep networks. *Image and Vision Computing*. 2015;65(1):3–14.

28. Recurrent neural networks (RNN) – the vanishing gradient problem [cited 2019 20 March]. Available from: https://www.superdatascience.com/recurrent-neural-networks-rnn-the-vanishing-gradient-problem/.

29. Lai S, Xu L, Liu K, Zhao J. Recurrent convolutional neural networks for text classification. *Twenty-ninth AAAI Conference on Artificial Intelligence*; 2015.

30. Baziotis C, Pelekis N, Doulkeridis C. DataStories at SemEval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

31. Ayata D, Saraclar M, Ozgur A. BUSEM at SemEval-2017 task 4: A sentiment analysis with word embedding and long short term memory RNN approaches. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

32. Cho K. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*; 2014.

33. Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Deep Learning Workshop at NIPS2014*; 2014.

34. Khan FH, Qamar U, Bashir S. A semi-supervised approach to sentiment analysis using revised sentiment strength based on SentiWordNet. *Knowledge and Information Systems*. 2016;51(3):851–872.

35. Giorgis S, Rousas A, Pavlopoulos J, Malakasiotis P, Androutsopoulos I.

aueb.twitter.sentiment at SemEval-2016 task 4: A weighted ensemble of SVMs for twitter sentiment analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

36. Gomez-Adorno H, Vilariño D, Sidorov G, Pinto D. CICBUAPnlp at SemEval-2016 task 4-A : Discovering twitter polarity using enhanced embeddings. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

37. Sarker A, Gonzalez G. DiegoLab16 at SemEval-2016 task 4 : Sentiment analysis in twitter using centroids, clusters, and sentiment lexicons. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

38. Martínez V, Pla F, Hurtado LF. DSIC-ELIRF at SemEval-2016 task 4: Message polarity classification in twitter using a support vector machine approach. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

39. Zhou Y, Zhang Z, Lan M. ECNU at SemEval-2016 task 4: An empirical investigation of traditional NLP features and word embedding features for sentence-level and topic-level sentiment analysis in twitter. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

40. Sarker A, Gonzalez G. HLP@UPenn at SemEval-2017 task 4A: A simple, self-optimizing text classification system combining dense and sparse vectors. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

41. Esuli A, Faedo IA, Moruzzi G. ISTI-CNR at SemEval-2016 task 4 : Quantification on an ordinal scale. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

42. Cozza V, Petrocchi M. MIB at SemEval-2016 task 4a: Exploiting lexicon based features for sentiment analysis in twitter. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

43. Corrêa EA, Marinho VQ, Borges L, Santos D. NILC-USP at SemEval-2017 task 4: A multi-view ensemble for twitter sentiment analysis. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

44. Nandi V, Agrawal S. Twitter sentiment analysis using hybrid approach. *International Research Journal of Engineering and Technology*. 2016;3(6):2887–2890.

45. Jabreel M, Moreno A. SiTAKA at SemEval-2017 task 4: Sentiment analysis in twitter based on a rich set of features. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

46. Lozi D, Šari D, Toki I, Medi Z, Šnajder J. TakeLab at SemEval-2017 task 4: Recent deaths and the power of nostalgia in sentiment analysis in twitter. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

47. Balikas G, Amini M. TwiSE at SemEval-2016 task 4: Twitter sentiment classification. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

48. Abreu J, Castro I, Martínez C, Oliva S, Gutiérrez Y. UCSC-NLP at SemEval-2017 task 4: Sense n-grams for sentiment analysis in twitter. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

49. Wang M, Chu B, Liu Q, Zhou X. YNUDLG at SemEval-2017 task 4: A GRU-SVM model for sentiment classification and quantification in twitter. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

50. Rosenthal S, Farra N, Nakov P. SemEval-2017 task 4: Sentiment analysis in twitter. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

51. Tang D, Qin B, Liu T. Document modeling with gated recurrent neural network for sentiment classification. *Proceedings of the 2015 Conference on Empirical Methods in*

Natural Language Processing; 2015.

52. Cliché M. BB twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

53. Rouvier M. LIA at SemEval-2017 task 4: An ensemble of neural networks for sentiment classification. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

54. Angel González J, Pla F, Hurtado LF. ELiRF-UPV at SemEval-2017 task 4: Sentiment analysis using deep learning. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

55. Zhang H, Wang J, Zhang J, Zhang X. YNU-HPCC at SemEval 2017 task 4: Using a multi-channel CNN-LSTM model for sentiment classification. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

56. Yang TH, Tseng TH, Chen CP. System implementation for SemEval-2017 task 4 subtask a based on interpolated deep neural networks. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

57. Xu S, Liang H, Baldwin T. UNIMELB at SemEval-2016 Tasks 4A and 4B: An ensemble of neural networks and a word2vec based model for sentiment classification. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

58. Deshmane AA, Friedrichs J. TSA-INF at SemEval-2017 task 4: An ensemble of deep learning architectures including lexicon features for twitter sentiment analysis. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*; 2017.

59. Vilaresa D, Doval Y, Alonso MA, Rodriguez CG. LyS at SemEval-2016 task 4:

Exploiting neural activation values for twitter sentiment classification and quantification. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

60. Khalil T, El-Beltagy SR. NileTMRG: Deep convolutional neural networks for aspect category and sentiment extraction in SemEval-2016 task 5. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*; 2016.

61. Sudhakar S. Convolution neural network [cited 2019 04 June]. Available from: https://towardsdatascience.com/convolution-neural-network-e9b864ac1e6c.

62. An end-to-end open source machine learning platform [cited 2019 04 June]. Available from: https://www.tensorflow.org/.

63. Scikit-learn machine learning in python [cited 2019 01 June]. Available from: https://scikit-learn.org/.

64. Keras: The python deep learning library [cited 2018 01 December]. Available from: https://keras.io/.

65. Brownlee J. A gentle introduction to k-fold cross-validation [cited 2019 01 June]. Available from: https://machinelearningmastery.com/k-fold-cross-validation/.