TEXT CATEGORIZATION USING SYLLABLES AND RECURRENT NEURAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Ersin Yar
July 2017

TEXT CATEGORIZATION USING SYLLABLES AND RECURRENT NEURAL NETWORKS

By Ersin Yar

July 2017

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Süleyman Serdar Kozat(Advisor)

Cem Tekin

Burcu Can Buğlalılar

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

TEXT CATEGORIZATION USING SYLLABLES AND RECURRENT NEURAL NETWORKS

Ersin Yar
M.S. in Electrical and Electronics Engineering
Advisor: Süleyman Serdar Kozat
July 2017

We investigate multi class categorization of short texts. To this end, in the third chapter, we introduce highly efficient dimensionality reduction techniques suitable for online processing of high dimensional feature vectors generated from freely-worded text. Although text processing and classification are highly important due to many applications such as emotion recognition, advertisement selection, etc., online classification and regression algorithms over text are limited due to need for high dimensional vectors to represent natural text inputs. We overcome such limitations by showing that randomized projections and piecewise linear models can be efficiently leveraged to significantly reduce the computational cost for feature vector extraction from the tweets. We demonstrate our results over tweets collected from a real life case study where the tweets are freely-worded and unstructured. We implement several well-known machine learning algorithms as well as novel regression methods and demonstrate that we can significantly reduce the computational complexity with insignificant change in the classification and regression performance.

Furthermore, in the fourth chapter, we introduce a simple and novel technique for short text classification based on LSTM neural networks. Our algorithm obtains two distributed representations for a short text to be used in classification task. We derive one representation by processing vector embeddings corresponding to words consecutively in LSTM structure and taking average of the produced outputs at each time step of the network. We also take average of distributed representations of the words in the short text to obtain the other representation. For classification, weighted combination of both representations are calculated. Moreover, for the first time in literature we propose to use syllables to exploit the sequential nature of the data in a better way. We derive distributed representations of the syllables and feed them to an LSTM network to obtain the distributed representation for the short text. Softmax layer is used

to calculate categorical distribution at the end. Classification performance is evaluated in terms of AUC measure. Experiments show that utilizing two distributed representations improves classification performance by $\approx 2\%$. Furthermore, we demonstrate that using distributed representations of syllables in short text categorization also provides performance improvements.

Keywords: Sentiment analysis, text categorization, distributed representation, long short term memory, fully connected layer.

ÖZET

TEKRARLAMALI SİNİR AĞLARI VE HECELERİ KULLANARAK METİN SINIFLANDIRMA

Ersin Yar
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Süleyman Serdar Kozat
Temmuz 2017

Kısa metinlerin çok sınıflı sınıflandırmasını incelemekteyiz. Bu amaçla, üçüncü bölümde serbestçe kelimelere dökülmüş metinden üretilen yüksek boyutlu öznitelik vektörlerinin çevrimiçi işlenmesine uygun son derece etkin boyut azaltıcı teknikler sunarız. Metin işleme ve sınıflandırma duygu tanıması, reklam seçimi vb. gibi birçok uygulamada yüksek derecede önemli olmasına rağmen çevrimiçi metin sınıflandırma ve regresyon algoritmaları doğal metin girdilerini gösterimlemek için yüksek boyutlu vektörlere olan ihtiyaçtan dolayı sınırlıdır. Bu gibi kısıtlamaların üstesinden öznitelik vektörü özütlemesi için hesaplama maliyetini ciddi ölçüde azaltan rasgeleleştirilmiş izdüşümler ve parçalı doğrusal modellerin etkin bir biçimde kullanıldığını göstererek gelmekteyiz. Bu sayede, gerçek zamanlı çok sınıflı tweet sınıflandırması ve regresyonu yapılabilmekteyiz. Sonuçlarımızı gerçek bir hayat çalışmasından toplanan serbestçe yazılmış ve düzensiz tweetler üzerinden göstermekteyiz. Özgün regresyon yöntemleri ile iyi bilinen makine öğrenimi algoritmaları uygulamakta ve sınıflandırma ve regresyon performansında önemli değişiklik olmadan hesaplama karmaşıklığının önemli ölçüde azaltıldığını göstermekteyiz.

Dahası, dördüncü bölümde kısa metin sınıflandırması için LSTM sinir ağlarına dayalı basit ve özgün bir teknik tanıtmaktayız. Algoritmamız sınıflandırmada kullanılmak üzere kısa bir metin için iki dağıtılmış gösterim elde eder. Bir gösterimi kelimelere karşılık gelen vektör gösterimlerini LSTM yapısında ardışık olarak işleyip ağda her bir zamanda üretilen çıktıların ortalamasını alarak üretmekteyiz. Diğer gösterimi üretmek için de kısa metindeki kelimelerin dağıtılmış gösterimlerinin ortalamasını alırız. Sınıflandırma için her iki gösterimin ağırlıklı birleşimi hesaplanır. Bundan başka, literatürde ilk defa, verinin ardışık doğasından daha iyi yararlanmak için heceleri kullanmayı önermekteyiz. Hecelerin dağıtılmış gösterimlerini elde ederiz ve kısa metnin dağıtılmış gösterimini çıkarmak için LSTM ağına veririz. En sonda sınıfsal dağılımı hesaplamak için softmax katmanı kullanılır.

Deneyler iki dağıtılmış gösterimden yararlanmanın sınıflandırma performansını 2% artırdığını gösterir. Ayrıca, kısa metin sınıflandırmasında hecelerin dağıtılmış gösterimlerini kullanmanın da performans iyileşmesi sağladığını göstermekteyiz.

Anahtar sözcükler: Duygu analizi, metin sınıflandırma, dağıtılmış gösterim, uzun kısa zaman bellek, tamamen bağlı katman.

Acknowledgement

I would like to express my gratitude to my advisor, Assoc. Prof. Süleyman Serdar Kozat, for his guidance throughout my M.S. study.

I would like to thank Asst. Prof. Cem Tekin and Asst. Prof. Burcu Can Buğlalılar as my examining committee members.

I would like to thank my friends from Middle East Technical University, especially Serdar Hanoğlu and İhsan Utlu who were with me in this great journey and my friends from Bilkent University.

Finally, I would like to thank my family for supporting me throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to recent developments in Internet technologies, the amount of accessible text information has significantly increased with the contribution of forums, columns, blogs, and social media. Clearly, processing of this big data, extracting information, performing classification and regression can significantly contribute to commercial products or to social sciences. However, text-based analysis proves to be very challenging due the variability and irregularity of media for text shares, the rapid variation of user sharing habits, and the large volume of data to be processed. Although text processing and classification are highly important due to many applications such as emotion recognition, advertisement selection, etc., online classification and regression algorithms over text are limited due to need for high dimensional vectors to represent natural text inputs. Especially, the state of the art representations such as the N-grams that are widely used as feature vectors require millions of components for accurate results, which deem them impractical for real time processing for text data such as real time emotion classification or sentiment analysis. This problem is especially exacerbated for agglutinative morphological structured languages such as Turkish, Finnish and Hungarian. These special languages derive words using extensive suffixes usually from a single word. Hence, the dimension of the word space exponentially increase unlike the Anglo-Saxon vocabularies. Because of the fundamental differences of agglutinative languages i.e. extreme usage of suffixes, making NLP research based on those languages is much more difficult.

To this end, in Chapter 3, we introduce highly novel and computationally efficient feature extraction methods that can be even used for agglutinative languages. We emphasize that our methods directly apply to English, however, we choose the Turkish language as the real life case study to demonstrate the versality of our approach. We construct online and offline algorithms for multi class classification of tweets, where we introduce highly efficient dimensionality reduction techniques suitable for online processing of high dimensional feature vectors generated from freely-worded text. Since we work on a real life application and the tweets are freely worded, we introduce a preprocessing pipeline with text correction, normalization and root finding components. Note that these components are also essential for other languages. We then introduce methods to derive feature vectors corresponding to tweets, which can be efficiently processed by the subsequent machine learning algorithms. We accomplish this by showing that randomized projections and piecewise linear models can be efficiently leveraged to significantly reduce the computational cost for feature vector extraction from the tweets. Hence, we can perform multi class tweet classification and regression in real time. We demonstrate that our methods increase the speed of text classification 10^2 times over the state-of-art methods such as PCA [11].

Moreover, in many applications from a wide variety of fields, the data to be processed can have a sequential structure, e.g., frames in video sequences [12], utterances in speech signals [13], DNA sequence in gene analysis [14]. Sequential data can be distinguished from the more-typical data in that the order of tokens within a sequence carries meaning and that the length of sequences within a dataset can vary. This kind of structured data exhibits sequential correlations, i.e., nearby points are likely to be related. Sequential patterns are important since they can be exploited to improve the performance.

A variety of techniques have been proposed for processing of sequential information [13, 15, 16]. Recurrent neural networks might be the most popular among these methods. The creation of internal state with self connections allow them to exhibit temporal behavior and to process inputs with arbitrary lengths. Although recurrent neural networks use the state information, which may potentially boost the performance in sequential tasks such as emotion recognition, the vanilla structure suffers from vanishing gradient problems [17, 18]. Exponential shrinking of gradient values to 0 due to

multiple matrix multiplications during training of RNNs limits their learning capability to a few time steps. Gradient contributions from far away steps become zero easily and states at those steps do not contribute to learning of long range dependencies. To remedy such considerations and at the same time still use the recurrent structure, Long Short Term Memory (LSTM) structures [19] are introduced. LSTMs are successful at capturing long term dependencies. Special cell structure presented in LSTMs allows the gradient flow through a number of steps back.

In addition, research on obtaining vector embeddings of words preserving semantic and syntactic patterns [20–22] have attracted a great amount of attention due to its simplicity and effectiveness. Word embedding is a mathematical representation of a word in k dimensional space. There are also studies obtaining unexpectedly successful results using word embeddings without considering the sequential knowledge embedded in text data [23]. Sequential processing combined with such a technique may yield better performance in classification tasks. Besides, the success of using smaller units of organization for a text sequence [24] and huge vocabularies of common languages promote to use smaller units to represent text data. The problem of large vocabulary is even exacerbated for agglutinatively morphological languages such as Turkish, Finnish and Hungarian.

To this end, in chapter 4, we propose a simple and novel framework for short text categorization. Underlying idea of our approach is to obtain two distributed representations for the given short text so that we use both representations to achieve better performance in classification tasks such as sentiment analysis. One representation results from processing of vector embeddings corresponding to words consecutively in LSTM structure for the given short text to efficiently exploit the sequential nature of the data. We take average of the produced outputs at each time step of the LSTM network to obtain a meaningful representation of the short text. The other representation results from directly averaging distributed representations of the words for the given short text. After both representations are obtained, we calculate weighted combination of them to predict the corresponding class of the given short text. We use softmax classification function to calculate class probabilities at the end. In addition, we propose to operate on syllables to exploit the sequential structure of the data in a better way. We process the short texts such that words are broken into their constituting syllables.

For the first time in literature, we derive distributed representations of the syllables and feed them to an LSTM network sequentially to obtain the distributed representation for the given short text. We demonstrate that using distributed representations of syllables yields a performance gain of $\approx 2\%$ in terms of AUC measure. We also show that weighted combination of two representations improves performance of basic LSTM layer.

The usage of syllables becomes prominent for agglutinative morphologically structured languages such as Turkish, Finnish and Hungarian. These special languages derive words using extensive suffixes usually from a single word. Hence, the dimension of the word space exponentially increase unlike the Anglo-Saxon vocabularies. Because of the fundamental differences of agglutinative languages i.e. extreme usage of suffixes, making NLP research based on those languages is much more difficult.

1.1 Natural Language Processing

In this section, we motivate the sentiment analysis problem considered in the subsequent chapters of this thesis. We start the discussion with Natural Language Processing (NLP) framework and then give different applications considered in NLP research. Afterwards, as the concentration of this thesis, we consider sentiment analysis as an important application of NLP. We point out initial studies from sentiment analysis literature. For the sake of completeness, we also give some earlier works related to sentiment analysis. Although this thesis focuses on supervised learning, other existing techniques in sentiment analysis research are also mentioned. Furthermore, since this thesis can also be considered under text categorization, we reserve a section and give related references in this part.

NLP is an active research area which investigates the methods to interpret natural text and speech using computers. NLP aims to discover how humans make use of natural languages. Hence, it develops suitable techniques to comprehend and manipulate natural languages. NLP is an area composed of contributions from a number of fields such as linguistics and computer science.

The history of NLP dates back to 1950s. Most of the NLP systems had been using complex set of hand written rules. The introduction of machine learning algorithms during the late 1980s has revolutionized the methods for language processing. Also, the contribution of increase in computational power to this development is considerable. Early successes in NLP occurred in machine translation. However, recent studies have focused on many different areas. The reason for this trend is to ability to obtain enormous amount of data.

NLP has many applications. We can group them under 4 main categories. These are syntactic, semantic, discourse and speech. Figure 1.1 shows each application under the corresponding group. Since NLP is a continuously developing field there might be more applications that might be also known with other names.

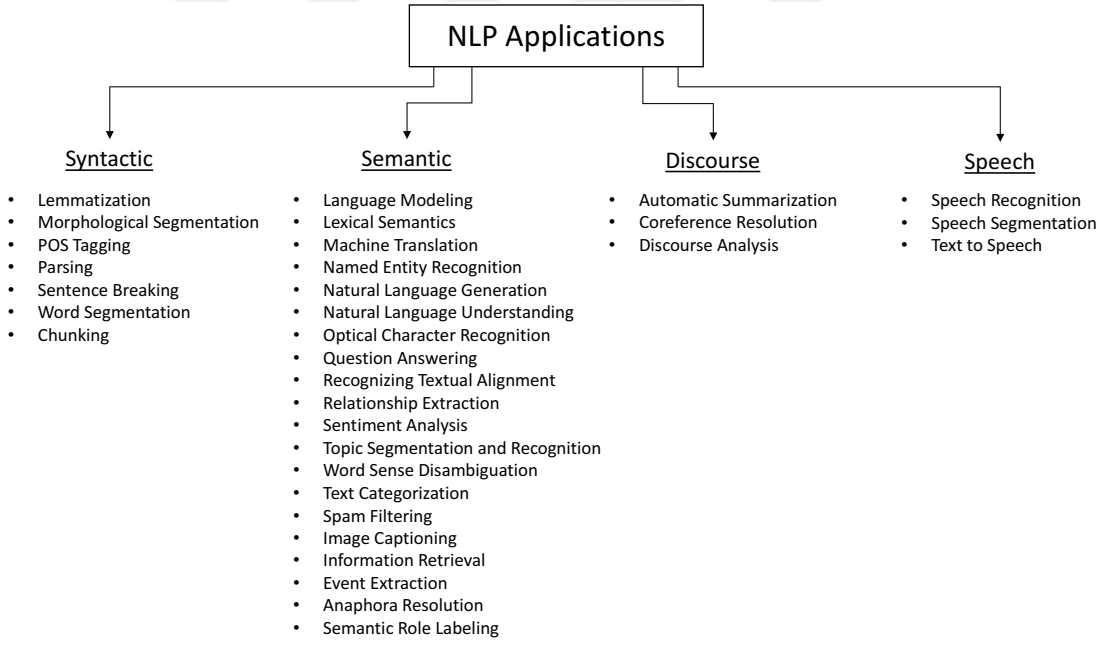

Figure 1.1: Different applications in NLP research grouped under 4 main categories, which are syntactic, semantic, discourse and speech.

1.1.1 Sentiment Analysis

Since the Internet usage has become widespread, online medium turned into a rich source of data. In addition, with the development of social media (e.g., blogs, forums), contents of this environment can presumably be used for various purposes. For instance, organizations may use available data on the Web in order to reach public opinion on a specific topic or a product. Similarly, as the interest in e-commerce grows customers mostly rely on reviews posted by existing customers. These developments open up many research directions in NLP such as opinion mining and emotion detection. There are also many names and slightly different tasks, e.g., sentiment analysis, opinion extraction, sentiment mining, emotion analysis, etc. These basically represent the same field. Sentiment analysis is the focus of this thesis.

Sentiment analysis, which is also called opinion mining, is the field of study that analyzes people's opinions, sentiments and emotions towards products, services, organizations, individuals and events. It aims to extract the emotion, whose existence is assumed, expressed in utterances of speech or in pieces of text, i.e., its goal is to determine overall attitude of a speaker or writer. Sentiment analysis mainly focuses on opinions which express or imply either positive or negative sentiments.

Sentiment analysis research can be divided into 2 main categories, learning based and machine learning based approach. This thesis investigates supervised learning methods under machine learning based approach. All available techniques are shown in Figure 1.2.

There are various studies conducted in sentiment analysis. These studies vary with respect to method, task and dataset. [9] is a comprehensive survey which investigates applications, common challenges and major tasks in sentiment analysis and opinion mining research. In addition, different possible tasks in sentiment analysis are presented in [10].

Although NLP has a long history, the number of studies conducted about people's opinions is limited before 2000. Since then, the field has become a very active research area. Although the sentiment analysis research mainly started at the beginning of 2000,

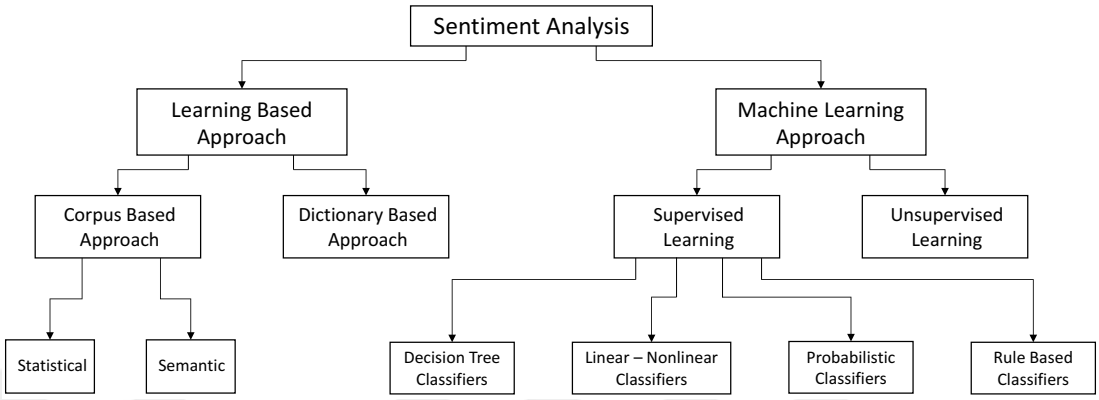

Figure 1.2: Methods used in sentiment analysis research.

there are some earlier works on sentiment adjectives and view points.

In [1] a method is presented to compare semantic orientation of joined adjectives. They propose to use conjunctions between adjectives as indirect information. This method is combined with supplementary morphology rules and it predicts whether joined adjectives are of same or different orientation. Moreover, the author of [3] proposes an approach to identify the person whose perspective exists in narratives. This method detects whether this character is a previously mentioned or new in the sentence by exploring regularities in how a character's perspective is reflected in the text. Furthermore, [4] presents a study to evaluate and develop coder reliability in discourse tagging using statistical techniques. This work shows a policy to choose a single best tag in case of disagreements on discourse tagging. It is applicable to any tagging task in which the coders show some symmetric disagreement resulting from bias. They formulate bias-corrected tags and produce an automatic classifier.

Although these studies considered sentiment analysis related tasks, one earliest study using sentiment analysis term first is [5]. Instead of making a classification of whole document as positive or negative, this paper presents an approach on extracting sentiments with associated positive or negative polarity for specific subjects in a document. They apply semantic analysis using a syntactic parser with sentiment lexicon and achieve high precision in finding sentiments. Also, the term opinion mining first appeared in [6]. This work introduces a method to differentiate positive and negative reviews. The procedure explained first trains a classifier on a corpus of self-tagged

reviews. This classifier is then improved using same corpus before applying it to sentences.

The research on sentiments and opinions appeared earlier even though sentiment analysis and opinion mining terms were not mentioned explicitly. One such study is given in [7]. This study presents a simple unsupervised learning algorithm for classifying reviews. The classification of a review is done by calculating the average semantic orientation of the phrases in the review. If average semantic orientation of the phrases in a given review is positive then it is classified as recommended. In addition, [8] specifies the existence of subjectivity using a method for clustering words with respect to distributional similarity. These features are further improved with the addition of lexical semantic features of adjectives. This study ensures high precision when features based on both similarity clusters and the lexical semantic features are employed.

1.1.2 Text Categorization

Text categorization is another important application of NLP. It is the task of assigning textual data into one of predefined categories. Text categorization or text classification may find useful applications in real world. One such application is spam filtering where e-mails are classified either as spam or not spam. Other well-known applications are organization of news stories by subject categories and classification of academic publications by domain of interest.

Text categorization has found application areas after the rapid growth of online information on the Internet. Since the online medium contains lots of data in the form of text, the processing and handling of this data in an organized way are very important. The problem of manually labeling documents, which can be very time consuming, can be taken care of by automatic categorization. To ease this problem, text classification approaches try to determine the category of the text. Categorization methods can be either manual or automatic. Manual methods are typically rule based techniques and automatic methods make use of machine learning techniques.

Although the focus of this thesis is given as sentiment analysis in the previous section, this thesis can also be considered under text categorization. The history of text categorization dates back to 1960. Most methods had used manually defined rules until the beginning of 90s. Starting from 90s, machine learning based approaches have gained popularity for text categorization. [11] gives a detailed survey of text categorization.

In this thesis, we focus on sequential data in the form of text. However, there are other applications of sequential data such as genomic research [25]. HMM are widely used in speech recognition [26]. Sequence classification is also important in information retrieval to categorize text and documents. The broadly used methods for document classification include Naive Bayes [27] and SVM [28]. Text classification has various extensions such as multi-label text classification [29], hierarchical text classification [30] and semi-supervised text classification [31]. [32] provides a more detailed survey on text classification.

Short text categorization is an important task in many areas of natural language processing including sentiment analysis [33], question answering [34] and machine translation [35]. Several different approaches have been used for short text classification such as using Support Vector Machines [36] and Naive Bayes in combination with SVMs [37] in the machine learning literature. Recently, the introduction of different neural network structures has brought a whole new perspective to machine classification problems. Several recent studies also focus on using convolutional neural networks [24] for this important task. Recurrent neural networks introduce memory to the network and can be used in many applications such as handwriting recognition [38] and generation [39], speech recognition [15, 16]. They are also used for language translation [35] and modeling [40]. Short texts generally appear to be composed of sequential components such as words forming sentences or utterances in speech. The processing of this sequential information may improve classification performance. Recent works on sequential short text classification using distributed representations of the words are considerable [41] and [42] employs convolutional and recurrent neural network structures together.

1.2 Thesis Contributions

The main contributions of this thesis are as follows:

- Although PCA and randomized projections are well studied techniques for dimensionality reduction, the idea of using them to process high dimensional feature vectors representing natural text inputs efficiently is rather new.
- The algorithm introduced in [49] constructs all piecewise linear regressors corresponding to different partitions of the regressor space and then calculates adaptive linear combination of the outputs of these regressors while we show that classification performance can still be improved using perfect partitions with increasing depth.
- To the best of our knowledge, this is the first study that uses syllables to process text data in a sequential manner. This is important for agglutinative languages such as Turkish, Finnish and Hungarian since vocabulary sizes are huge due to extensive use of suffixes and prefixes for these languages.
- We propose a novel idea based on the weighted combination of two distributed representations of a text input. One of these is obtained by processing distributed representations corresponding to words in LSTM structure and taking average of LSTM outputs after last time step. The other representation is obtained by taking average of distributed representations of words in the text directly.
- We demonstrate the significant performance gains achieved by our algorithm over numerical examples and real data sets.

1.3 Thesis Outline

There are five chapters in this thesis. In the second chapter, we explain how distributed representations are obtained. We consider continuous bag of words and skip gram approaches. Detailed network structures and equations are provided.

In the third chapter, we introduce highly efficient dimensionality reduction techniques and piecewise linear models for online classification of high dimensional feature vectors. We apply several data preprocessing techniques on our collected data. We then present a vector space model to construct feature vectors from text and perform classification using several machine learning algorithms. We illustrate the performance of the introduced algorithms via various simulations.

In the fourth chapter, we investigate short text classification using LSTM neural networks and syllables. We provide a novel method on combining two distributed representations of text inputs for classification. In addition, for the first time in literature, we obtain distributed representations of syllables and use them for sequential short text classification. Performances of the introduced methods are illustrated via extensive simulations over Turkish and Arabic languages.

Finally, in the fifth chapter, we conclude the thesis.

Chapter 2

Distributed Representations

2.1 Word2Vec

Word2vec model has attracted great amount of attention in recent years. It learns the distributed representations of words using a neural network model. The vector representations of words learned by word2vec models have been shown to carry semantic and syntactic relationships. They are useful in many NLP tasks. There are two methods that word2vec uses to obtain word embeddings, which are continuous bag of words (CBOW) and skip gram (SG). The former derives a representation of a word given its context while the latter obtains the representations using the word to predict the context. We used both methods. Therefore, we explain them in detail.

2.1.1 Continuous Bag of Words

2.1.1.1 One Word Context

This is the most simple form of CBOW model. In this model, we consider a word to be predicted by only a word as the context. In other words, the model will predict one target word for given one context word. In Figure 2.1 we give the schematic of neural

network used. We consider the vocabulary size as V, and the vector size as N. The layers are fully connected. The input and output is one-hot encoded vectors. It means that for a vector of size V only one of the dimensions is equal to 1 and all other units are zero. The index of only nonzero entry is the index of the corresponding word in the vocabulary set.

The weights between input and hidden layer is represented by a $V \times N$ matrix $\mathbf{W}^i$. Each row of $\mathbf{W}^i$ is the N dimensional vector representation $\mathbf{v}_w$ of the corresponding word. In other words, $\mathbf{W}^i$ is the embedding matrix whose rows correspond to distributed representations of specific words. Given a context word, we have

$$\mathbf{h} = \mathbf{W}^{i^T}\mathbf{x} = \mathbf{v}_{w_I}^T.$$

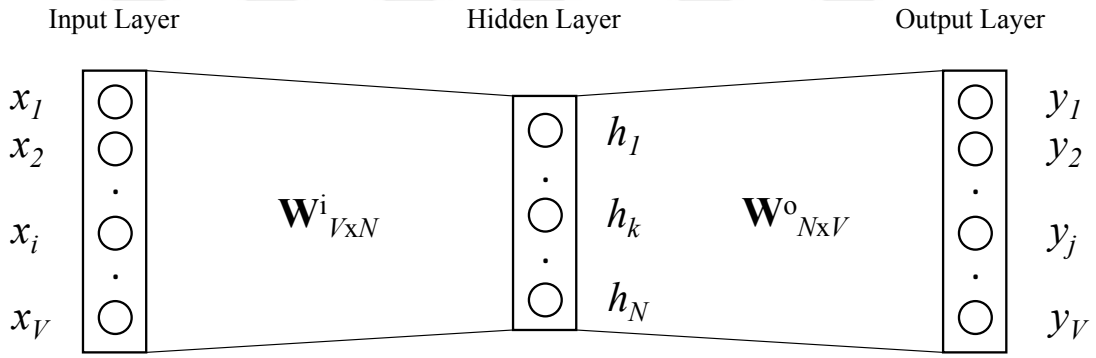

Figure 2.1: Network structure used to obtain distributed representations when context consists of only one word for continuous bag of words model.

This operation is basically copying the row of $\mathbf{W}_i$ corresponding to context word to $\mathbf{h}$. $\mathbf{v}_{w_I}$ is the vector representation of the input word w_I. Note that contrary to ordinary neural networks, the activation function of the neurons in the hidden layer is linear. Therefore, from input layer to hidden layer we just choose the associated row of word embedding matrix $\mathbf{W}^i$.

From hidden layer to output layer, we have W^o that maps the context word to output word. The dimension of this matrix is $N \times V$. Using these weights, we obtain a score

for each word in the vocabulary

$$u_j = \mathbf{W}^{o^T}_{[:,j]}\mathbf{h},$$

where $\mathbf{W}^{o}_{[:,j]}$ denotes the j^{th} column of the matrix $\mathbf{W}^{o}$. We then can use softmax function to obtain the posterior distribution of words

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j' \neq j}^{V} \exp(u_{j'})},$$

where y_j is the output of the j^{th} unit in the output layer. Substituting $\mathbf{h}$ and u_j into the last equation we obtain

$$p(w_j|w_I) = y_j = \frac{\exp(\mathbf{W}^{o^T}_{[:,j]}\mathbf{h})}{\sum_{j' \neq j}^{V} \exp(\mathbf{W}^{o^T}_{[:,j']}\mathbf{h})}.$$

Update Equation for Output Layer

Our training objective is to maximize the conditional probability of observing the actual output word w_O (let's denote its index in the output layer with j^*) given the input context word w_I with regard to the weights. We define the loss function as the logarithm of the conditional probability

$$\begin{aligned}\log p(w_O|w_I) &= \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) = -E,\end{aligned}$$

where $E = -\log p(w_O|w_I)$ is our loss function. Our goal is to minimize E. We note that loss function is actually the cross entropy function. Let L denote cross entropy measure. It is defined as

$$L = \sum_{j=1}^{V} -t_j \log(y_{j^*}),$$

where $t_j = \mathbb{1}(j = j^*)$ is the indicator function, i.e., t_j will only be 1 when the j^{th} unit is the actual output word, otherwise $t_j = 0$. Therefore, cross entropy measure boils down to our error function.

To derive the update equation of the weights of the output layer we need to take the derivative of error function with respect to each element of output matrix

$$\frac{\partial E}{\partial w_{ij}^o} = \frac{\partial E}{\partial u_j}\frac{\partial u_j}{\partial w_{ij}^o} = e_j h_i.$$

Using stochastic gradient descent we obtain the weight update equation for output weight matrix as

$$\mathbf{W}_{[:,j]}^o = \mathbf{W}_{[:,j]}^o - \eta e_j \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V,$$

where $\eta > 0$ is the learning rate.

Update Equation for Input Layer

After stating update equation for W^o we can derive update equation for W^i. We can take the derivative of E with respect to W^i. For this purpose, we can write

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i}\frac{\partial h_i}{\partial w_{ki}},$$

where $h_i = \sum_{k=1}^{V} x_k w_{ki}$. Now we need to take derivative of E with respect to h_i

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j}\frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j w_{ij}^o = \mathbf{e}\mathbf{w}_i^o,$$

since $\frac{\partial h_i}{\partial w_{ki}} = x_k$

$$\frac{\partial E}{\partial \mathbf{W}} = \mathbf{x}\mathbf{e}\mathbf{w}^{\mathsf{T}}.$$

Note that only one component of $\mathbf{x}$ is non zero, so only one row of $\frac{\partial E}{\partial W}$ is non zero. The update equation for W^i is as follows

$$\mathbf{v}_{w_I} = \mathbf{v}_{w_I} - \eta \mathbf{e}\mathbf{w}^{\mathsf{T}},$$

where $\mathbf{v}_{w_I}$ is the row of $\mathbf{W}^i$ corresponding to the context word. All other rows of $\mathbf{W}^i$ remain unchanged since their derivatives are zero.

2.1.1.2 Multi-Word Context

Figure 2.2 shows the CBOW model with multi-word context. To produce hidden layer vector, this time instead of copying only the vector corresponding to context word we take the average of vectors of context words

$$\begin{aligned}\mathbf{h} &= \frac{1}{C}\mathbf{W}^{i^{\mathsf{T}}}(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) \\ &= \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C}),\end{aligned}$$

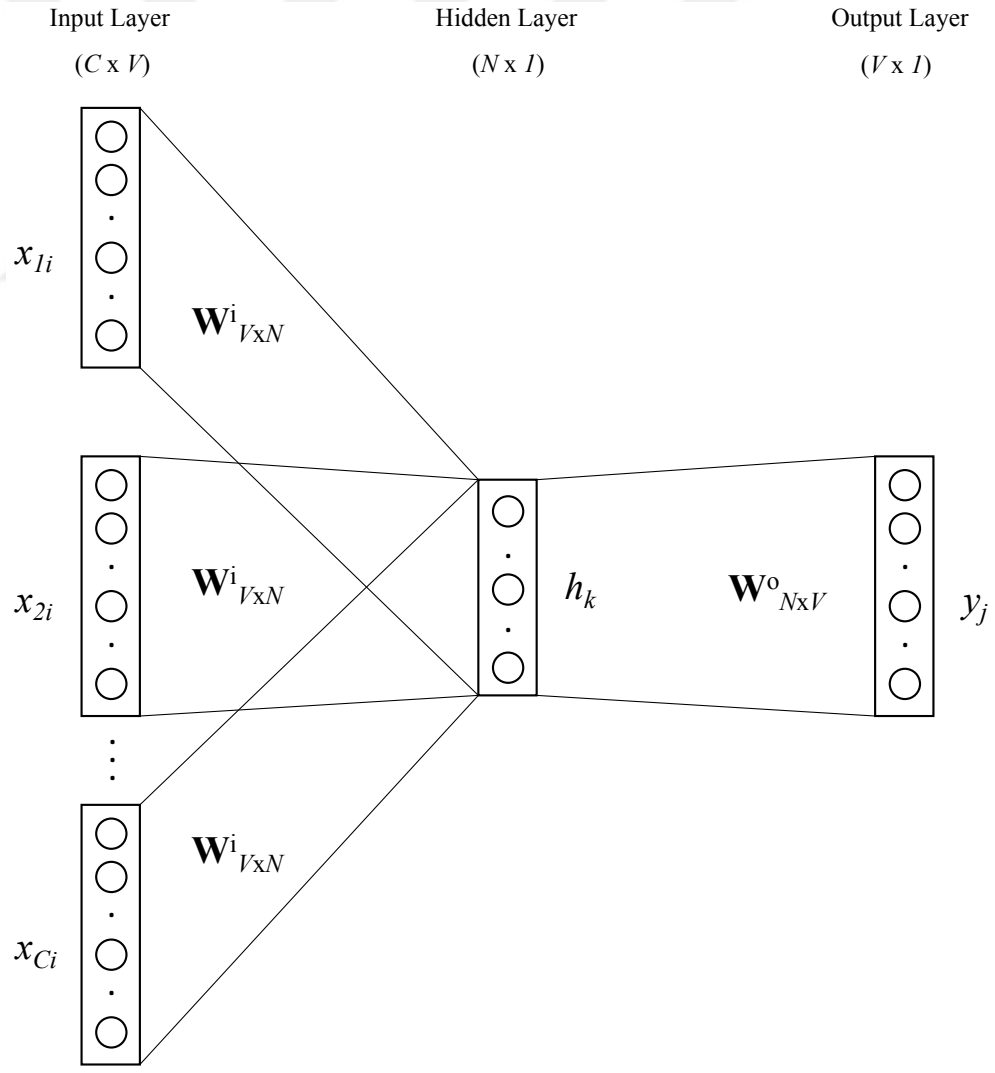

Figure 2.2: Network structure used to obtain distributed representations when context consists of more than one word for continuous bag of words model.

where C is the number of words in the context and $w_1, \cdots, w_C$ are the context words. $\mathbf{v}_w$ is vector representation of the word w. The loss function is

$$\begin{aligned} E &= -\log p(w_O|w_{I,1}, \cdots, w_{I,C}) \\ &= -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \\ &= -\mathbf{v}_{w_O}{}^{\mathsf{T}}\mathbf{h} + \log \sum_{j'=1}^{V} \exp(\mathbf{v}_{w_{j^*}}{}^{\mathsf{T}}\mathbf{h}). \end{aligned}$$

The update equation for the output layer is same as before

$$\mathbf{W}^o_{[:,j]} = \mathbf{W}^o_{[:,j]} - \eta e_j \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

The update equation for input layer is also same as before only with a minor difference

$$\mathbf{v}_{w_{I_c}} = \mathbf{v}_{w_{I_c}} - \frac{1}{C}\eta \mathbf{e}\mathbf{w}^{\mathsf{T}} \qquad \text{for } c = 1, 2, \cdots, C,$$

where $\mathbf{v}_{w_{I_c}}$ is the vector representation of the c^{th} word in the input context.

2.1.2 Skip Gram

Figure 2.3 shows skip gram model. The idea in CBOW model is to predict a word given its context to derive vector representations of words. In skip gram model, words are used to predict related context. In this manner, it is the opposite of CBOW model.

Similar to CBOW, the operation performed in input layer is to copy the row of input weight matrix corresponding to given word to hidden layer. The definition of hidden layer's content is same

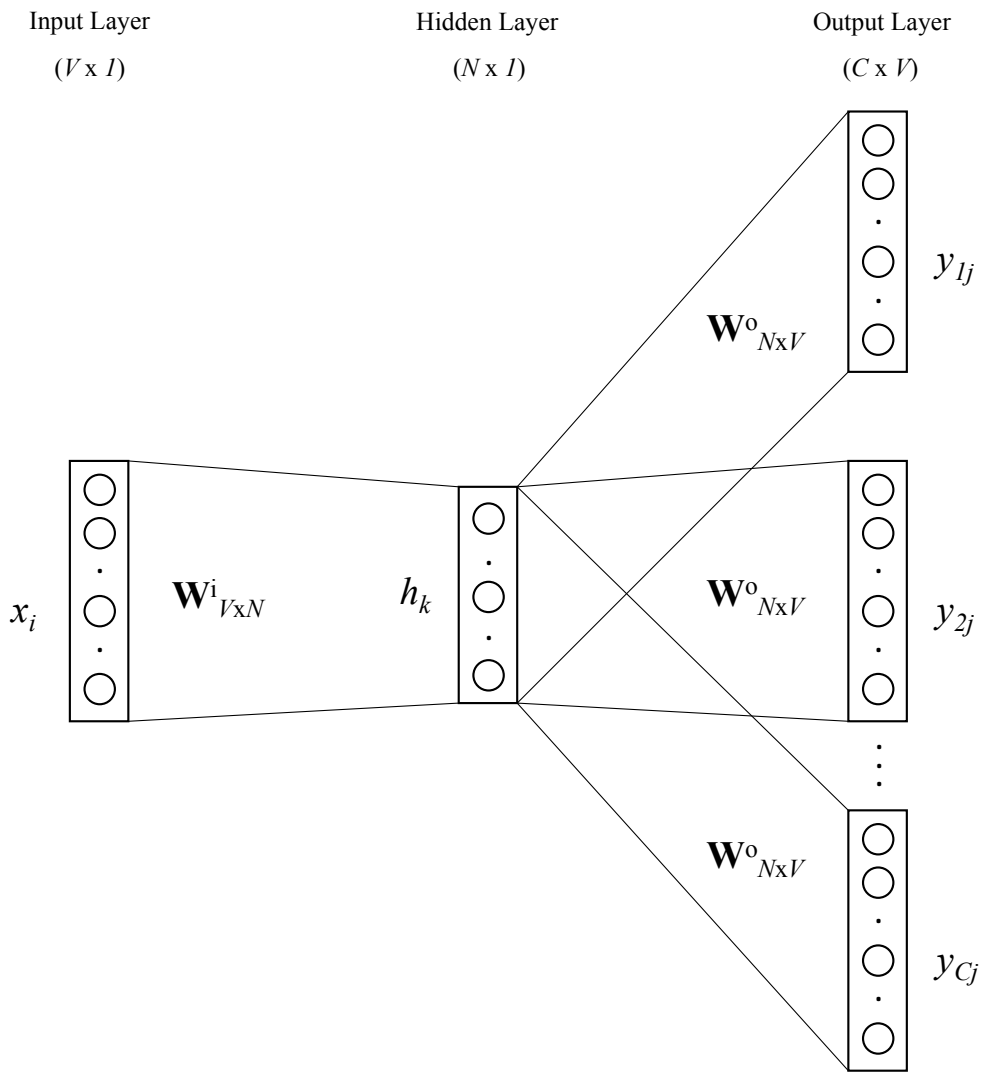

Figure 2.3: Network structure used to obtain distributed representations using skip gram model.

$$\mathbf{h} = \mathbf{W}^{iT}\mathbf{x} = \mathbf{v}^T_{w_I}.$$

For output layer, instead of generating one multinomial distribution, C multinomial distributions are calculated. Same output layer matrix is used to compute each output

$$p(w_{c_j} = w_{O_c}|w_I) = y_{c_j} = \frac{\exp(u_{c_j})}{\sum_{j'=1}^{V}\exp(u_{j'})},$$

where w_{c_j} is the j^{th} word on the c^{th} panel of the output layer, w_{O_j} is the actual c^{th} word in the output context words and w_I is the input word. Since output layer shares same output weight matrix for all words in the given context the following relation could be reached

$$u_{c_j} = \mathbf{v}_{w_j}^{o\ \mathsf{T}} \mathbf{h} \qquad \text{for } c = 1, 2, \cdots, C,$$

where $\mathbf{v}_{w_j}^o$ is the column of the output matrix corresponding to word w_j. The derivation of parameter update equations is very similar to CBOW model. The loss function is defined as

$$\begin{aligned} E &= -\log p(w_{O_1}, w_{O_2}, \cdots, w_{O_C} | w_I) \\ &= -\log \prod_{c=1}^{C} \frac{\exp(u_{c_{j_c^*}})}{\sum_{j'=1}^{V} \exp(u_{j'})} \\ &= -\sum_{c=1}^{C} u_{j_c^*} + C \log \sum_{j1=1}^{V} \exp(u_{j'}), \end{aligned}$$

where j_c^* is the index of the actual c^{th} output context word in the vocabulary. We take the derivative of error with respect to output matrix $\mathbf{W}^o$

$$\frac{\partial E}{\partial w_{ij}^o} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c_j}} \frac{\partial u_{c_j}}{\partial w_{ij}^o} = \mathbf{EI}_j h_i,$$

where we define $\frac{\partial E}{\partial u_{c_j}} = e_{c_j}$ and $\mathbf{EI}_j = \sum_{c=1}^{C} e_{c_j}$. Thus, we obtain update equation for output layer as

$$\mathbf{W}_{[:,j]}^o = \mathbf{W}_{[:,j]}^o - \eta \mathbf{EI}_j \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

This update is same as before with a minor difference, which is the summation of errors for all context words. The derivation of update equation for input layer is same as before considering that prediction error e_j is replaced with $\mathbf{EI}_j$. It is given below

$$\mathbf{v}_{w_I} = \mathbf{v}_{w_I} - \eta \mathbf{e}\mathbf{w}^{\mathsf{T}},$$

where $(ew)_i = \sum_{j=1}^{V} \mathbf{EI}_j w_{ij}^o$.

Chapter 3

Computationally Highly Efficient Online Text Classification and Regression for Real Life Tweet Analysis

In this chapter, we study multi class classification of tweets. For tweet analysis we introduce preprocessing techniques due to unstructured and freely worded tweets. We then obtain feature vectors by representing them in our vector space model. We introduce highly efficient dimensionality reduction techniques suitable for online processing of high dimensional feature vectors generated from tweets.

3.1 Regression and Classification on Freely Worded Tweets

In this section, we first present our case study and data collection procedure. We then introduce our data preprocessing steps since the tweets are freely worded. After

preprocessing, we construct feature vectors using these tweets and then introduce our classification and regression methods in a real life case scenario.

3.1.1 Data Collection

The tweets in our database are gathered through a case study where 1440 tweets written in Turkish are collected from 168 different users between April 10th, 2013 and May 28th, 2013. These users are selected among people studying at Koç University. There are at most 10 tweets from a single user. The tweets, whose contents can be related to anything, are freely worded and unstructured. There are 3 classes, i.e., a tweet fall into one of three categories, which are "No Statement(0)", "Specific(1)" and "General(2)". These categories reflect the level of statement about other people in a tweet.

Tweets are manually labeled by human experts. For this purpose, three human coders are employed. We choose Krippendorff's α to measure inter-coder agreement or inter-rater reliability. Human coders manually labeled tweets at a reliability of Krippendorffs α= 0.7.

3.1.2 Data Preprocessing

Agglutinative morphological structure of languages such as Turkish, Finnish and Hungarian enables one to derive numerous words using derivational suffixes even from a single root [43]. Thus, dimension of the word space constructed as a collection of distinct words can be considerably large. Moreover, we observe that tweets are freely worded, unstructured and they are not typed correctly all the time. Same word can emerge in significantly different forms due to aforementioned issues. Therefore, we apply a number of data preprocessing techniques to interpret the tweets properly. Our methods are generic such that they can be applied to any languages.

To this end, we removed urls, links and location information as well as mentions in tweets. We also discarded retweets. There are some words encountered frequently

in most of the sentences that do not carry importance in terms of providing thematic content. Hence, we used a list of common words to eliminate them. We also eliminate numbers and the words having sizes smaller than 3. We then apply text correction to correct first the unwanted characters and then to correct the words that are misspelled. As we mention earlier, in agglutinatively morphological languages words having similar meanings can have the same roots. To be able to represent these words in one form, we apply stemming to obtain the roots. After these operations, final form of the tweets are obtained. The process pipeline is explained in Figure 3.1.

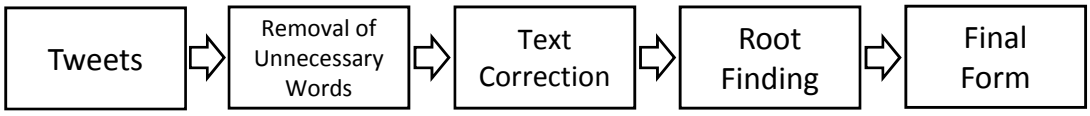

Figure 3.1: Processing pipeline of tweets.

3.1.3 Vector Space Model

We use a vector space model to represent tweets in our corpus. In tweet classification we define our vocabulary as the union of all distinct words used in the whole dataset and equate the dimension of our vector space to the size of the vocabulary. We represent the tweets in terms of N-grams [44], which is a representation technique consisting of N consecutive words. In this study, we use unigrams and bigrams to represent the tweets. An example of N-grams representation is given in Figure 3.2.

In vector space model, we express each tweet as a vector where each component is related to a distinct word and assign a weight to that component. We use "TF-IDF" measure to calculate this weight [45]. "TF" means term frequency and we take it as the relative frequency of a word in a tweet. "IDF" means inverse document frequency and emphasizes how uncommon of a word is between other tweets .If a word does not appear in many tweets we increase its emphasis according to "IDF" measure. "TF" and "IDF" measures are found by

$$TF(f,t) = \frac{f}{t},$$

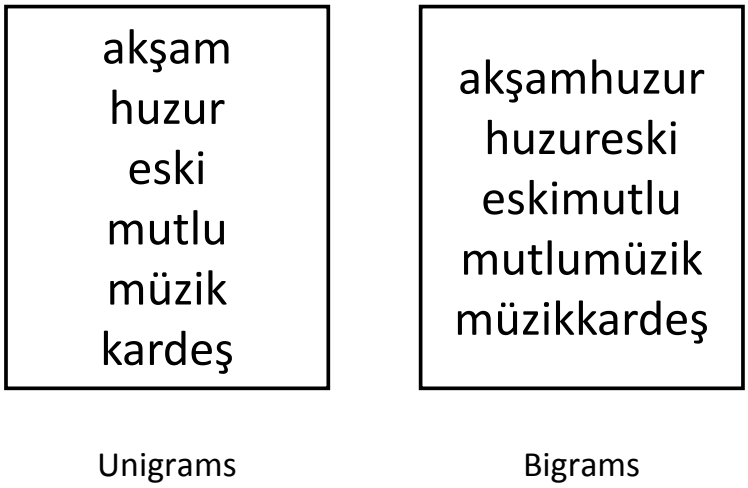

Figure 3.2: Unigram and bigram representation of a tweet.

$$IDF(f, d_t) = 1 + \frac{\log(|d_t|)}{|t|},$$

where f is the current word, t is the corresponding tweet and d_t denotes tweet corpus. In our vector space model we use the multiplication of both term as the weight for a word in a tweet

$$TF - IDF_{(f,t,d_t)} = TF(f, t) * IDF(f, d_t).$$

At the end of these operations for each tweet t_t in tweet space $T = \{t_1, t_2, ..., t_n\}$ we derive a d dimensional feature vector $t_t = [w_1, w_2, ..., w_d]^T$. Since text inputs are represented in high dimensional vectors we introduce two methods to represent them in low dimensional vectors to process efficiently, namely random projection and principal component analysis.

To this end, we present random projection as a simple and computationally efficient way to reduce the dimensionality of the data [46]. We project the original d dimensional vector to k-dimensional space by multiplying it with a random $k \times d$ dimensional matrix R. We construct this random matrix R chosing its entries randomly from the set $\{-1, 1\}$ or as samples from standard normal distribution.

Principal component analysis is another dimensionality reduction technique we employ. We map the high dimensional feature vectors to a lower dimensional space by

multiplying them with a $k \times d$ transformation matrix whose rows are the eigenvectors corresponding to the k largest eigenvalues of the covariance matrix of data [46].

We verify the validity of the transformations of feature spaces from high dimension to low dimension using following lemma.

Johnson Lindenstrauss lemma: *For any* $0 < \epsilon < 1$ *and any integer n, let k be a positive integer such that*

$$k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln(n).$$

Then for any set V of points in $\mathbb{R}^{\mathrm{d}}$*, there is a map* $f : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}^{\mathrm{k}}$ *such that* $\forall u, v$

$$(1 - \epsilon)\|u - v\|_2 \leq \|u - v\|_2 \leq (1 + \epsilon)\|u - v\|_2 .$$

Using the result of Johnson Lindenstrauss lemma [47] we show that we can transform points from a high-dimensional space to a lower dimensional space in such a way that the distances between the points remain approximately same [48].

We are interested in preserving the information as much as possible while applying dimensionality reduction techniques. It is desirable to keep the pairwise distances same between data points for a projection in low dimensional space, which can be important for the application of algorithms such as nearest neighbors. The goal of random projection is to maintain pairwise distances between data points. In that manner, the lemma states when points in high dimensional space are randomly projected to low dimensions, the pairwise squared distances between the points change by a factor of no more than $1 \pm \epsilon$ with large probability. In other words, the lemma guarantees that random projections do not distort the distances between points with a certain probability.

3.1.4 Classification

We define automatic tweet classification as the process of identifying the class which a tweet belongs to. There is a space containing tweets $T = \{t_1, t_2, ..., t_n\}$, where each

tweet t_t is represented by a d dimensional vector $t_t = [w_1, w_2, ..., w_d]^T$, where each w_k is the weight of term k in tweet t_t and there is a fixed set of classes $C = \{c_1, c_2, ..., c_C\}$. Our goal is to build a classification function matching tweets to their classes.

We carry out classification in two parts. In the first part we perform offline classification where we use all the data available. In the second part we introduce online classification of tweets by using them sequentially.

3.1.4.1 Offline Classification

There are many types of algorithms used in text categorization [11]. In this study, we use the following classifiers

- Support Vector Machines
- K-Nearest Neighbors
- Decision Trees
- Logistic Regression

In this part, we employ classification algorithms given above along with two different dimensionality reduction techniques, namely random projection and principal component analysis. We give the results in simulations section.

3.1.4.2 Online Classification

In this part, we use a piecewise linear model [49] to represent the relationship between features vectors and class labels. We construct this piecewise linear model combining seperate linear models trained in disjoint regions that are generated by partitioning d dimensional feature space using seperator functions. Our approach is adaptive in the sense that at each instance both model parameters and seperator function parameters are updated. In other words, we adaptively train model parameters and seperator

function parameters to minimize the final regression error. We point out that as we sequentially classify tweets both model and seperator function parameters are adjusted such that space partitioning characterizes the structure of the data better and piecewise linear model predicts the corresponding class more accurately. In order to obtain satisfactory results parameter tuning should be done carefully. In Figure 3.3 we indicate a sample partitioning of two dimensional feature space into 4 disjoint regions.

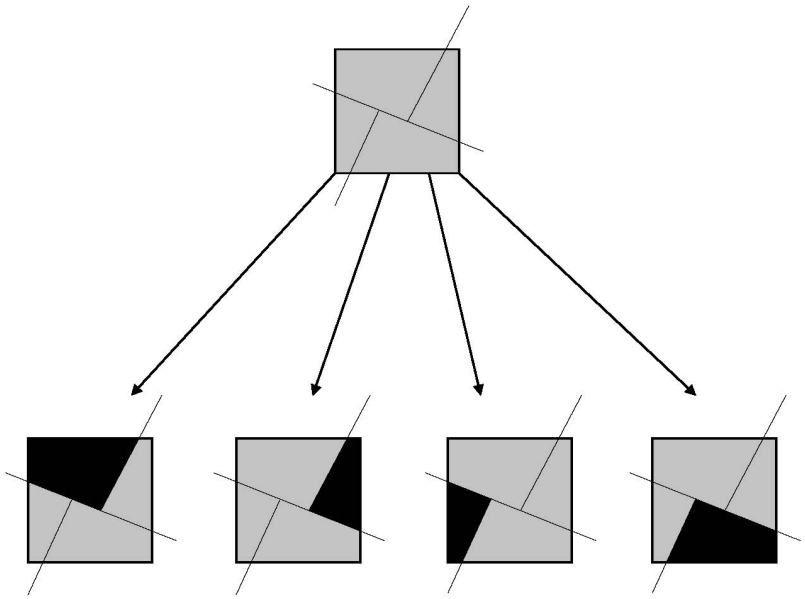

Figure 3.3: Sample partitioning of a two dimensional feature space into 4 disjoint regions.

3.2 Simulations

In this section, we demonstrate the performances of the algorithms. The dimensions of the feature vectors for unigrams and bigrams are 2511 and 6139, respectively. We reduce each of these dimensions to 125 and 250 applying different dimensionality reduction techniques. We obtain AUC values as performance measure for classification algorithms by optimizing their parameters over grid search using 10-fold cross validation.

We point out that the AUC values obtained using low dimensional feature vectors

are comparably smaller than the AUC values obtained without applying dimensionality reduction. This small loss comes with gain in computational complexity. For instance, logistic regression classifier utilizing random projection executes at least 100 times faster than the standard logistic regression classifier.

The results are given in Table 3.1 and in Table 3.2 for unigrams and bigrams, respectively. In Table 3.3 computational complexities of classification algorithms are given [46, 50, 51].

Table 3.1: AUC scores obtained using classification algorithms with different dimensionality reduction techniques for unigrams.

Classifier / Dim. Reduction	SVM	KNN	DT	Log. Reg.
No Reduction	0.8173	0.7915	0.7306	0.8089
PCA_{125}	0.8083	0.7934	0.6650	0.8022
PCA_{250}	0.8107	0.7980	0.6824	0.8053
$\text{RP}_{-1,1_{125}}$	0.7802	0.7743	0.6192	0.6894
$\text{RP}_{-1,1_{250}}$	0.7904	0.7766	0.6402	0.7082
$\text{RP}_{\text{Gaussian}_{125}}$	0.7849	0.7776	0.6382	0.6966
$\text{RP}_{\text{Gaussian}_{250}}$	0.7944	0.7822	0.6403	0.7279

Table 3.2: AUC scores obtained using classification algorithms with different dimensionality reduction techniques for bigrams.

Dim. Reduction \ Classifier	SVM	KNN	DT	Log. Reg.
No Reduction	0.7806	0.7678	0.7330	0.7828
PCA_{125}	0.7619	0.7499	0.6161	0.7596
PCA_{250}	0.7679	0.7552	0.6084	0.7679
$\text{RP}_{-1,1_{125}}$	0.7760	0.7622	0.6330	0.6659
$\text{RP}_{-1,1_{250}}$	0.7756	0.7769	0.6397	0.6814
$\text{RP}_{\text{Gaussian}_{125}}$	0.7696	0.7549	0.6550	0.6653
$\text{RP}_{\text{Gaussian}_{250}}$	0.7727	0.7757	0.6287	0.6723

Our goal is not to obtain high AUC values in these experiments. We want to show that we can still obtain classification performance close to original case in exchange for gain in computational complexity. We expect decrease in performance since we ignore some information when dimensionality reduction is applied. Experimental results verify our expectations also. The best performance is obtained when no dimensionality reduction is applied. The values we obtain for other cases using different classifiers are lower than the value obtained in no dimensionality reduction case. Moreover, when we increase the reduced dimension from 125 to 250 the classification performance improves, which demonstrates that taking into account more information results in better performance.

Table 3.3: Comparison of the computational complexities of classification algorithms. In the table, n represents the number of training instances, d represents regular dimension, k represents reduced dimension.

Algorithm	**Computational Complexity**
SVM	$O\left(n^3\right)$
SVM with PCA	$O\left(n^3\right)$
SVM with RP	$O\left(n^3\right)$
KNN	$O\left(nd\right)$
KNN with PCA	$O\left(nd\right)$
KNN with RP	$O\left(nk\right)$
DT	$O\left(dn^2\log(n)\right)$
DT with PCA	$O\left(kn^2\log(n)\right)$
DT with RP	$O\left(dn^2\log(n)\right)$
Log. Reg.	$O\left(nd^2\right)$
Log. Reg. with PCA	$O\left(nk^2\right) + O\left(nd\right)$
Log. Reg. with RP	$O\left(nk^2\right)$

For online classification, we illustrate the performance of our algorithm having 1, 2 and 4 disjoint regions with respect to the truncated Volterra filter [52]. In Figure 3.4 we provide the time accumulated regression errors for each of them averaged over 10 trials. We emphasize that as the number of regions increase error value decreases and the performance of algorithm with 4 regions is comparable to the performance of Volterra filter.

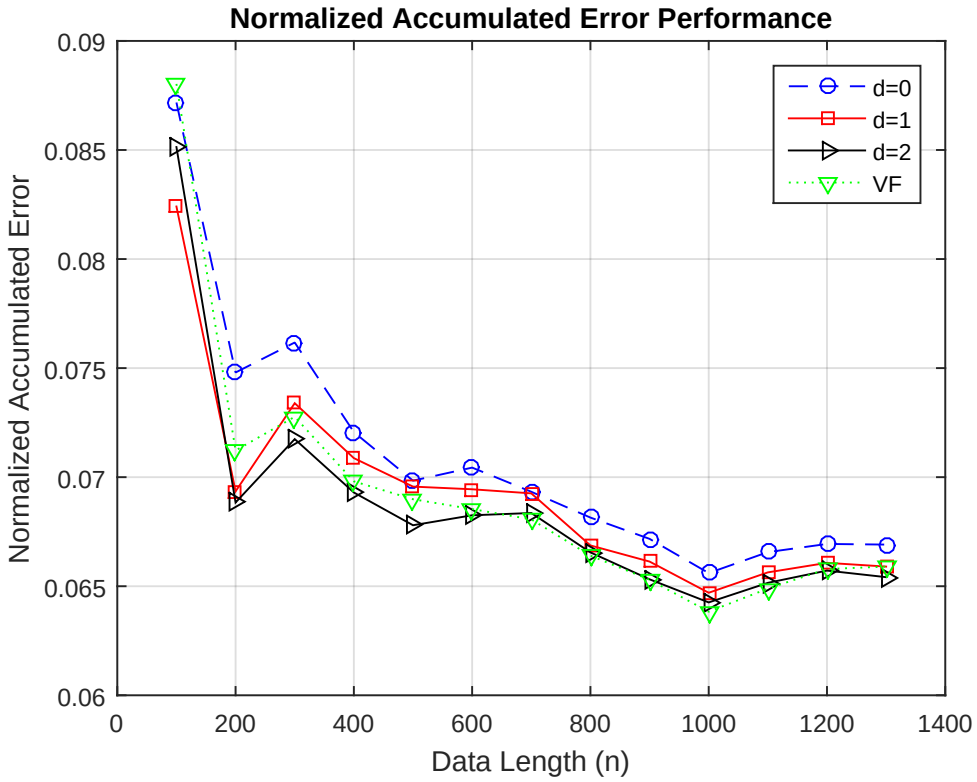

Figure 3.4: Normalized accumulated error performance.

Chapter 4

Short Text Categorization Using Syllables and LSTM Networks

We denote scalars with italic lowercases (e.g. x), vectors with bold lowercases (e.g. $\mathbf{x}$) and matrices with bold uppercases (e.g. $\mathbf{W}$). We use notation $\mathbf{x}^{i:j}$ to denote the sequence of vectors $(\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_j)$.

We study sequential short text classification. Each short text is composed of words or syllables, each of which is represented by a distributed vector $\mathbf{x}^l \in \mathbb{R}^k$. Hence, for i^{th} short text whose length is l we have a sequence of vectors $\{\mathbf{x}^{1:l}\}^i$. Moreover, each short text is associated with a label determining the class that the short text belongs to. We represent this label by a vector $\mathbf{d}^i \in \mathbb{R}^{|C|}$ whose only nonzero entry that corresponds to the given class is 1. Here, C is the set denoting possible classes and $|C|$ is the cardinality of the set C. Our goal is to sequentially estimate $\mathbf{d}^i$ by

$$\mathbf{z}^i = f(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^l),$$

where $f(\cdot)$ is a classification function. For each short text i, classification error is given by the categorical cross entropy function, i.e.,

$$\begin{aligned} E_i &= -\sum_{j=1}^{C} d_j^i \cdot \log(z_j^i) \\ &= -d_j^i \log(z_j^i), \end{aligned}$$

where d_j^i and z_j^i correspond to true label and estimate for the j^{th} class of i^{th} short text, respectively. Note that only d_j^i is nonzero for i^{th} short text from j^{th} class.

4.1 Data Preprocessing

To obtain distributed representations of the words and syllables in an unsupervised manner, we collected more than 500M tweets in Turkish and around 100M tweets in Arabic using Twitter's API [1]. These tweets are cleaned to keep only meaningful units. This process includes removal of links, urls, user mentions and hashtags since these do not contribute to the meaning of text. We use Zemberek [2] Turkish NLP tool for separation of words into syllables. We also insert special character '&' to denote spaces.

4.2 LSTM and Our Model

For a given short text of length *l*, we have a set of *k* dimensional vectors $\mathbf{x}^{1:l}$, which are inputs to the LSTM model to produce the *m* dimensional short text representation **s**. For the t^{th} unit in the short text, an LSTM layer takes $\mathbf{x}^t$, $\mathbf{h}^{t-1}$ and $\mathbf{c}^{t-1}$ and produces $\mathbf{h}^t$, $\mathbf{c}^t$ based on the following formulas:

$$
\begin{aligned}
\mathbf{i}^t &= \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{h}^{t-1} + \mathbf{b}_i) \\
\tilde{\mathbf{c}}^t &= \tanh(\mathbf{W}_c\mathbf{x}^t + \mathbf{R}_c\mathbf{h}^{t-1} + \mathbf{b}_c) \\
\mathbf{f}^t &= \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{h}^{t-1} + \mathbf{b}_f) \\
\mathbf{c}^t &= \mathbf{f}^t \odot \mathbf{c}^{t-1} + \mathbf{i}^t \odot \tilde{\mathbf{c}}^t \\
\mathbf{o}^t &= \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{h}^{t-1} + \mathbf{b}_o) \\
\mathbf{h}^t &= \mathbf{o}^t \odot \tanh(\mathbf{c}^t)
\end{aligned}
$$

where we do not use peephole connections since we do not need to learn precise timings. $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_c, \mathbf{W}_o \in \mathbb{R}^{m\times k}$, $\mathbf{U}_i, \mathbf{U}_f, \mathbf{U}_c, \mathbf{U}_o \in \mathbb{R}^{m\times m}$ are weight matrices and

[1] https://dev.twitter.com/streaming/public
[2] https://github.com/ahmetaa/zemberek-nlp

$\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_c, \mathbf{b}_o \in \mathbb{R}^m$ are bias vectors. The symbols $\sigma(\cdot)$ and $\tanh(\cdot)$ refer to the elements wise sigmoid and hyperbolic tangent functions, and $\odot$ is the element wise multiplication. $\mathbf{h}^0 = \mathbf{c}^0 = 0$.

In the pooling layer, the sequence of output vectors $\mathbf{h}^{1:l}$ from the LSTM layer are combined into a single vector $\mathbf{s} \in \mathbb{R}^m$ that represents the short text using mean pooling. Mean pooling averages all vectors, i.e., $\mathbf{s} = \frac{1}{l}\sum_{t=1}^{l} \mathbf{h}^t$.

Our model introduces a novel contribution to LSTM neural network. In our model, the sequence of vectors $\mathbf{x}^{1:l}$ input to the LSTM layer are also combined into a single vector $\mathbf{y} \in \mathbb{R}^k$ that also represents the short text using the mean pooling. Similarly, mean pooling averages all vectors, i.e., $\mathbf{y} = \frac{1}{l}\sum_{t=1}^{l} \mathbf{x}^t$. These representations, $\mathbf{s}$ and $\mathbf{y}$, are passed through separate fully connected layers and then summed. Finally, a softmax layer is used to obtain probabilities corresponding to each class

$$\mathbf{z} = \text{softmax}(\mathbf{W}_s\mathbf{s} + \mathbf{W}_y\mathbf{y} + \mathbf{b}_z),$$

where softmax$(\mathbf{z})$ is defined as $\frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for $j = 1, \ldots, K$.

The final output z_i represents the probability distribution over the set of C classes for the i^{th} short text and the j^{th} element of z_i corresponds to the probability that the i^{th} short text belongs to the j^{th} class as shown in Fig. 4.1.

4.3 A Novel Way to Represent Short Texts

As a further extension, we also use syllables to represent short texts. Syllables are the phonological building blocks of the words. We claim that syllables could also compose sequential components in short texts constructing words and thus sentences. The processing of this sequential information can improve classification performance.

Word2vec algorithm produces distributed representations of words. Although it is applied to a corpora of continuous text of words originally, it can also be applied

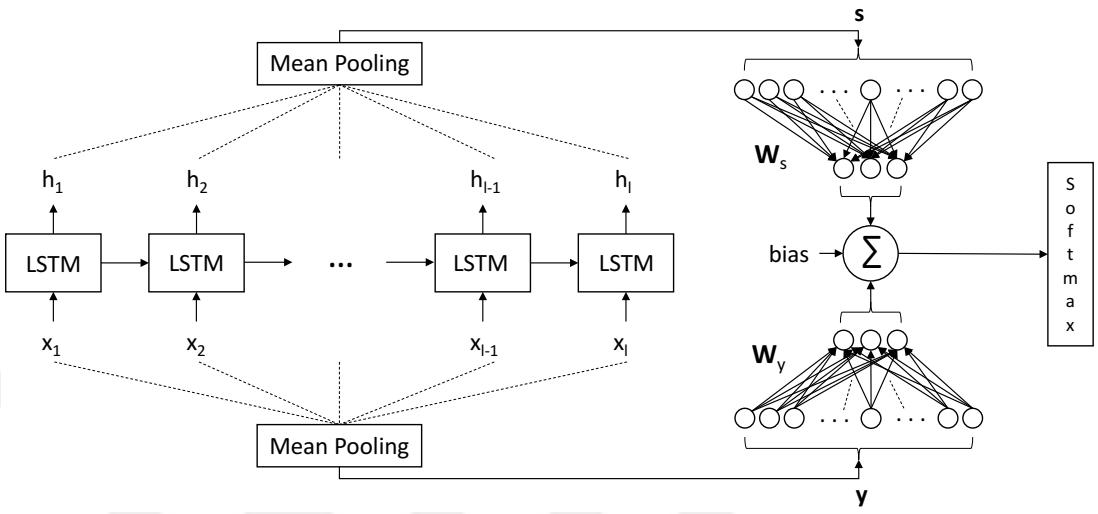

Figure 4.1: Classification model taking weighted combination of two distributed representations of a short text based on LSTM layer. One representation, **s**, is obtained after mean pooling of LSTM outputs while the other, **y**, is the average of inputs of LSTM layer.

to anything that has neighboring structure. Therefore, we use it in order to obtain distributed representations of syllables.

Distributed representations capture semantic and syntactic relationships between words. However, this does not apply for syllables since most of the syllables do not have a meaning by themselves. Only a small part of them that constitutes a word by itself have this property. Therefore, it is not conclusive to try to find semantic relationships between syllables. However, there exists syntactic relationships that can be inferred using distributed representations to some extent. For example, the link between question morphemes, $\{mı, mi, mu, etc\}$, in Turkish which are syllables by themselves can easily be inferred using distributed representations of syllables.

Syllables are important since they construct words gathering together although most of them are not meaningful by themselves. There is a complex relationship expressing how they combine to form words. Distributed representation of syllables can exploit this information.

4.4 Experiments

In this section, we present the results of the experiments we perform to compare the proposed novel methods with state of the art approaches. We use gensim *word2vec* [53] package in Python to obtain distributed representations of words and syllables. For the sake of completeness, we provide performance results of state of the art methods, which are random forests and support vector machines.

4.4.1 Dataset Descriptions

The dataset that we use for supervised classification task is composed of 6000 tweets in Turkish language [54]. They are labeled by human coders according to their sentiment polarity and each tweet is categorized into one of 3 classes, which are negative, positive and neutral. There are 3000 negative, 1552 positive and 1448 neutral tweets. Similar to unsupervised case, these tweets are also cleaned. Cleaning process contains removal of links, urls, user mentions and hashtags. Also, we use Zemberek Turkish NLP tool to separate words into syllables.

We also use several datasets in Arabic language for sentiment analysis [55, 56]. Datasets cover various domains.

- **Product Reviews (PROD)**: Products domain has a dataset of 4K reviews from the Souq website. The dataset includes reviews from Egypt, Saudi Arabia, and the United Arab Emirates.
- **Restaurant Reviews (RES)**: Restaurants dataset contains 2.6K reviews collected from TripAdvisor.
- **BBN Dataset**: It contains a random subset of 1200 Levantine dialectal sentences chosen from the BBN Arabic - Dialect / English Parallel Text. The sentences are extracted social media posts.
- **Syrian Tweets**: This is a dataset of 2000 tweets originating from Syria (Levantine dialectal Arabic is commonly spoken). These tweets were collected in May

2014 by polling the Twitter API.

First two datasets provide the text of the review as well as a rating entered by the reviewer for each sample. The rating reflects the overall sentiment of the reviewer towards the entity. Hence, for each review, the rating was extracted and normalized into one of three categories: positive, negative or neutral using the same approach adopted by [33, 57]. In addition to these, the manual sentiment annotations were performed for BBN dataset and Syrian tweets. Each post is annotated by at least 10 annotators and the majority sentiment label was chosen. Class distributions are given in Table 4.1.

Table 4.1: The number of data instances in each class for datasets used in experiments.

Classes / Datasets	Negative	Neutral	Positive	Total
PROD	863	308	3101	4272
RES	268	265	2109	2642
BBN Dataset	575	126	498	1200
Syrian Tweets	1323	392	285	2000
Turkish Tweets	3000	1448	1552	6000

4.4.2 Training

The models are trained to minimize the negative log-likelihood of predicting the correct class of the sentences in the training set, using stochastic gradient descent with the Adam update rule. At each gradient descent step, the weight matrices, bias vectors and word vectors are updated. For regularization, dropout [58] is applied to weight matrices. We limit parameter grid for epoch number with 10 to prevent overfitting.

Table 4.2: Experimental range for all hyperparameters.

Hyperparameter	Experiment Range
RF estimator num.	50, 100, 150, 250, 500
RF max. depth	5, 10, 15, 25, 50
RF split eval. criterion	gini, entropy
SVM C	2e-3, 2e-2, 0.2, 2, 5, 10, 20
SVM gamma	2e-3, 2e-2, 0.2, 2, 5, 10
SVM kernel	linear, rbf
LSTM block size	50, 100, 200, 250, 300, 400, 500
LSTM epoch num.	3, 4, 5, 10
LSTM batch size	8, 16, 32, 64
LSTM optimizer	adam, rmsprop
LSTM initialisation	lecun uniform
LSTM dropout rate	0, 0.1, 0.2
Vec. size for words	200, 300, 400
Vocab. size for words	5K, 10K, 50K
Context win. for words	10, 15, 20
Vec. size for syllables	100, 150, 200
Vocab. size for syllables	5K, 10K, 50K
Context win. for syllables	15, 20, 25

4.4.3 Simulations

We use Keras [59] framework to perform simulations. We separate 25% of the labeled dataset randomly as a test set and never use it neither for training nor validation data during experiments. The remaining 75% of the dataset is used for training. Parameter optimization is performed via 5-fold cross validation. In particular, we search over the number of trees in the forest, maximum depth of the tree and criterion to evaluate split quality for random forest classifier while we scan C, gamma and kernel for support vector machines.

Furthermore, our parameter grid contains block size, epoch number, dropout rate, batch size, optimiser choice and initialisation method of network weights for LSTM. Moreover, we also try different parameters while obtaining distributed representations of words and syllables to see their effects on performance results. These are dimensions of feature vectors, width of the context window and vocabulary size. We sort and get the most frequent words in order to keep meaningful ones. Hence, vocabulary size is limited to 50K. The whole parameter set we search over is given in Table 4.2.

We use Area Under the ROC Curve (AUC) measure to compare performances of aforementioned methods. ROC curve is a graphical plot that illustrates the performance of a binary classifier as its discrimination threshold is varied. The curve is produced by plotting true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The AUC is defined as the area under this curve. The optimum operating point for ROC curve is at TPR = 1 and FPR = 0, therefore, higher AUC means better performance. Classification accuracy can be very sensitive to unbalanced class distributions [60]. However, the area under ROC curve is insensitive to this lack of balance in the dataset [61]. Although AUC is defined for binary classification originally, we use micro averaging method which calculates AUC globally to use it in multi-class setting, i.e., we binarize the multi-class problem and consider all binary classifiers as a single binary classifier.

We also calculate coverage rates that are defined as how much percent of the dataset is covered by the words or syllables that are in the vocabulary since this closely relates to classification performance closely. The number of syllables in a language is limited. On the contrary, the number of words in a language is much higher than syllables. The situation is even exacerbated for agglutinative languages such as Turkish, Finnish and Hungarian. Hence, it is expected to obtain higher coverage rates when syllables are used. Table 4.1 shows coverage rates for words and syllables for vocabulary sizes of 5K, 10K and 50K, respectively. The numbers indicate that even a 5K vocabulary of syllables is enough to cover the whole Turkish dataset. On the other hand, coverage rates are significantly smaller for words and they expectedly increase when larger vocabulary is used. The highest coverage rate is obtained when 50K vocabulary is used. Even in this case the coverage rate is relatively small and it is only 58%.

Table 4.3: Coverage rate of Turkish dataset when different vocabulary sizes are used for words and syllables.

Vocabulary Size	Word	Syllable
5K	44%	99%
10K	50%	99%
50K	58%	100%

4.4.4 Results and Discussions

This section presents AUC results obtained using distributed representations of words and syllables in different classifiers and makes related discussions.

4.4.4.1 Word LSTM vs. Word LSTM + Word vector avg. in Turkish

Experimental results show that using sentence representations obtained by averaging vectors in combination with LSTM layer improves classification performance of the basic LSTM layer for different parameter settings of word2vec models. A number of deductions can also be made from results. Using smaller vector sizes in word2vec models gives better results. Similarly, our model performs better when smaller vector sizes are used. Proposed model achieves its best performance with a context window of 15, vocabulary of 5K and vector dimension of 200. Performance gain is more than 1% with respect to the basic LSTM. Simulation results also point out that using wider context window for word2vec models improves classification performance most of the time. Another interesting observation is that the better results are obtained for RF and SVM when vocabulary is 5K. The reason for this behavior may be due to the sparsity introduced by larger vocabulary. While basic LSTM performs best when the vocabulary size is 50K, we obtain the best performance of our model at 5K vocabulary. This shows the benefit of using word vector averaging. Table 4.4 shows results for various combinations of vocabulary and context window size when the vector size is

200 whereas Table 4.5 and Table 4.6 demonstrate results when the vector sizes are 300 and 400, respectively.

Table 4.4: AUC scores for Turkish dataset using distributed representations of words when the vector size is 200 for different vocabulary (v) and context window parameters.

Parameters \ Classifier			RF	SVM	LSTM	Our Model
vector size: 200	v: 5K	context window: 10	**0.8052**	0.8298	0.7717	0.7755
		context window: 15	0.7988	0.8300	0.7765	**0.7864**
		context window: 20	0.7975	0.8273	0.7782	0.7775
	v: 10K	context window: 10	0.7945	0.8159	0.7717	0.7724
		context window: 15	0.7945	0.8159	0.7714	0.7648
		context window: 20	0.7918	0.8165	0.7669	0.7683
	v: 50K	context window: 10	0.7837	0.8031	0.7719	0.7773
		context window: 15	0.7835	0.8040	0.7767	0.7741
		context window: 20	0.7841	0.8038	**0.7798**	0.7713

Table 4.5: AUC scores for Turkish dataset using distributed representations of words when the vector size is 300 for different vocabulary (v) and context window parameters.

Parameters \ Classifier			RF	SVM	LSTM	Our Model
vector size: 300	v: 5K	context window: 10	0.7995	0.8324	0.7702	0.7683
		context window: 15	0.7959	0.8310	0.7691	0.7656
		context window: 20	0.7957	0.8281	0.7693	0.7679
	v: 10K	context window: 10	0.7919	0.8133	0.7600	0.7627
		context window: 15	0.7919	0.8169	0.7616	0.7643
		context window: 20	0.7901	0.8171	0.7658	0.7602
	v: 50K	context window: 10	0.7827	0.8029	0.7667	0.7658
		context window: 15	0.7827	0.8024	0.7641	0.7668
		context window: 20	0.7827	0.8026	0.7669	0.7664

Table 4.6: AUC scores for Turkish dataset using distributed representations of words when the vector size is 400 for different vocabulary (v) and context window parameters.

Parameters / Classifier			RF	SVM	LSTM	Our Model
vector size: 400	v: 5K	context window: 10	0.7998	**0.8333**	0.7589	0.7607
		context window: 15	0.7956	0.8324	0.7601	0.7677
		context window: 20	0.7966	0.8298	0.7593	0.7601
	v: 10K	context window: 10	0.7939	0.8147	0.7531	0.7560
		context window: 15	0.7903	0.8134	0.7525	0.7526
		context window: 20	0.7866	0.8131	0.7537	0.7579
	v: 50K	context window: 10	0.7826	0.8020	0.7635	0.7648
		context window: 15	0.7816	0.8016	0.7635	0.7644
		context window: 20	0.7805	0.8006	0.7628	0.7641

Note that basic LSTM layer and our model do not perform as good as state of the art methods. The reason for that behavior may be the small size of the dataset we use. Also, since the examples in sentences are tweets their lengths are limited to 140 characters. Using longer sentences could result in exploitation of sequential information in a better way and hence could improve the performance.

4.4.4.2 Word LSTM vs. Syllable LSTM

Experimental results show that using distributed representations of syllables improves word2vec performance for several parameter combinations. We also provide AUC results obtained by using direct averaging of syllable embeddings classified with RF and SVM for the sake of completeness. The performance of LSTM layer using sequential information hidden in syllable embeddings outperforms the state of the art methods and provides a performance increase up to 8%. Unlike word case we cannot infer that using small vector size leads to better performance. While using models with a context window of 20 performs better for the basic LSTM, our algorithm works better

when a context window of 25 is used. When distributed representations of syllables are utilized the best results for the basic LSTM and our model are 0.7695 and 0.7737, respectively. These figures are higher than most of the results obtained with different parameter settings when distributed representations of words are used. Table 4.7 shows results for various combinations of vocabulary and context window size when the vector size is 100 whereas Table 4.8 and Table 4.9 demonstrate results when the vector sizes are 150 and 200, respectively.

Table 4.7: AUC scores for Turkish dataset using distributed representations of syllables when the vector size is 100 for different vocabulary (v) and context window parameters.

Parameters \ Classifier			RF	SVM	LSTM	Our Model
vector size: 100	v: 5K	context window: 15	0.7154	0.7346	0.7635	0.7632
		context window: 20	0.7116	0.7374	0.7690	0.7640
		context window: 25	0.7174	0.7356	0.7642	0.7646
	v: 10K	context window: 15	0.7136	0.7392	0.7646	0.7546
		context window: 20	0.7221	0.7414	0.7625	0.7571
		context window: 25	0.7107	0.7371	0.7689	0.7690
	v: 50K	context window: 15	0.7149	0.7356	0.7597	0.7670
		context window: 20	0.7177	0.7332	0.7643	0.7570
		context window: 25	0.7171	0.7344	0.7558	0.7598

Directly averaging syllable embeddings to represent the sentence does not lead to good results when random forest and SVM are used as classifiers. When compared to word case there is an evident decrease in AUC values obtained using state of the art results. However, the values obtained via basic LSTM and our model with embedding vectors of syllables does not seem to show degradation. On the contrary, the performance improves for a number of different parameter settings for syllables.

Table 4.8: AUC scores for Turkish dataset using distributed representations of syllables when the vector size is 150 for different vocabulary (v) and context window parameters.

Parameters \ Classifier			RF	SVM	LSTM	Our Model
vector size: 150	v: 5K	context window: 15	0.7190	0.7488	0.7669	0.7651
		context window: 20	0.7222	0.7486	0.7656	0.7609
		context window: 25	0.7181	0.7494	0.7662	0.7556
	v: 10K	context window: 15	0.7212	0.7469	0.7621	0.7521
		context window: 20	0.7199	0.7466	**0.7695**	0.7541
		context window: 25	**0.7245**	0.7460	0.7614	**0.7737**
	v: 50K	context window: 15	0.7176	0.7455	0.7577	0.7706
		context window: 20	0.7208	0.7471	0.7612	0.7687
		context window: 25	0.7161	0.7462	0.7596	0.7692

Table 4.9: AUC scores for Turkish dataset using distributed representations of syllables when the vector size is 200 for different vocabulary (v) and context window parameters.

Parameters \ Classifier			RF	SVM	LSTM	Our Model
vector size: 200	v: 5K	context window: 15	0.7189	**0.7526**	0.7675	0.7616
		context window: 20	0.7160	0.7525	0.7632	0.7631
		context window: 25	0.7208	0.7524	0.7630	0.7646
	v: 10K	context window: 15	0.7196	0.7515	0.7582	0.7667
		context window: 20	0.7180	0.7526	0.7650	0.7630
		context window: 25	0.7209	0.7512	0.7631	0.7622
	v: 50K	context window: 15	0.7181	0.7528	0.7655	0.7569
		context window: 20	0.7219	0.7546	0.7654	0.7587
		context window: 25	0.7176	0.7552	0.7650	0.7502

4.4.4.3 Word LSTM vs. Word LSTM + Word Vector avg. in Arabic

We also perform experiments in Arabic language. While we set vector size as 300 and context window as 10, we test 3 different size for vocabulary which are 5K, 10K and 50K. We also use bag-of-words and TF-IDF [62] feature extraction techniques to construct feature vectors. Similar to the experiments for Turkish dataset, random forest and SVM are chosen as state of the art classifiers. Simulation results show that our algorithm improves the performance of the basic LSTM layer for each vocabulary size. The performance of restaurant reviews dataset improves for 5K and 10K vocabulary whereas the performance of BBN dataset improves for 5K and 50K vocabulary. In addition, our algorithm provides enhancement in AUC for product reviews and Syrian tweets datasets when the vocabulary sizes are 50K and 10K, respectively. Furthermore, AUC values tend to increase when larger vocabulary is used. Table 4.10 shows results for various classifiers when the vocabulary size is 5K. Table 4.11 and Table 4.12 demonstrate results when the vocabulary size are 10K and 50K, respectively. Context window of size 10 and vector size of 300 are used in these experiments.

Table 4.10: AUC scores for Arabic datasets using distributed representations of words when the vector size is 300 with a vocabulary of size 5K and context window of 10.

Classifier / Datasets		PROD	RES	BBN Dataset	Syrian Tweets
vec = 300 win = 10 v = 5K	$\mathrm{RF}_{\mathrm{bow}}$	0.9009	0.9434	0.7446	0.7842
	$\mathrm{SVM}_{\mathrm{bow}}$	0.9060	0.9466	0.6508	0.8376
	$\mathrm{RF}_{\mathrm{tfidf}}$	0.8883	0.9362	0.7503	0.8097
	$\mathrm{SVM}_{\mathrm{tfidf}}$	0.9073	0.9433	0.6427	0.8354
	$\mathrm{RF}_{\mathrm{w2v}}$	0.8606	0.9154	0.7752	0.8348
	$\mathrm{SVM}_{\mathrm{w2v}}$	0.8839	0.9281	0.7918	0.8246
	LSTM	0.8785	0.9326	0.7458	0.8137
	Our Model	0.8774	0.9346	0.7514	0.8068

Table 4.11: AUC scores for Arabic datasets using distributed representations of words when the vector size is 300 with a vocabulary of size 10K and context window of 10.

Datasets \ Classifier		PROD	RES	BBN Dataset	Syrian Tweets
vec = 300 win = 10 v = 10K	$\mathrm{RF}_{\mathrm{bow}}$	0.8997	0.9368	0.7564	0.8270
	$\mathrm{SVM}_{\mathrm{bow}}$	0.9097	0.9367	0.7533	0.8514
	$\mathrm{RF}_{\mathrm{tfidf}}$	0.8966	0.9333	0.7460	0.8255
	$\mathrm{SVM}_{\mathrm{tfidf}}$	0.9093	0.9385	0.6867	0.8391
	$\mathrm{RF}_{\mathrm{w2v}}$	0.8869	0.9235	0.7853	0.8470
	$\mathrm{SVM}_{\mathrm{w2v}}$	0.8886	0.9283	0.7960	0.8547
	LSTM	0.8931	0.9339	0.7638	0.8346
	Our Model	0.8898	0.9399	0.7599	0.8348

Table 4.12: AUC scores for Arabic datasets using distributed representations of words when the vector size is 300 with a vocabulary of size 50K and context window of 10.

Datasets \ Classifier		PROD	RES	BBN Dataset	Syrian Tweets
vec = 300 win = 10 v = 50K	$\mathrm{RF}_{\mathrm{bow}}$	0.8958	0.9469	0.7625	0.8430
	$\mathrm{SVM}_{\mathrm{bow}}$	0.9040	0.9428	0.6720	0.8548
	$\mathrm{RF}_{\mathrm{tfidf}}$	0.8945	0.9435	0.7499	0.8438
	$\mathrm{SVM}_{\mathrm{tfidf}}$	0.9070	0.9422	0.7570	0.8468
	$\mathrm{RF}_{\mathrm{w2v}}$	0.8962	0.9358	0.8018	0.8610
	$\mathrm{SVM}_{\mathrm{w2v}}$	0.9079	0.9486	0.8134	0.8641
	LSTM	0.9028	0.9491	0.7657	0.8389
	Our Model	0.9046	0.9393	0.7712	0.8355

Chapter 5

Conclusion

We study multi class classification of tweets in Chapter 3, where we present preprocessing techniques since the tweets are freely worded. We construct feature vectors from tweets using our vector space model. Vector space model uses unigrams and bigrams to represent tweets. Since text inputs are represented with high dimensional vectors we introduce highly efficient dimensionality reduction techniques. To this end, we present principal component analysis and random projection as elegant dimensionality reduction techniques. We show that we can significantly reduce the computational complexity with insignificant change in classification performance. We also present piecewise linear models suitable for online processing of tweets. These models are constructed by adaptively combining linear models trained in disjoint regions that are generated by partitioning space with separator functions. We demonstrate their performance over minimization of time accumulated regression error.

Similarly, in Chapter 4, we study multi class categorization of short texts using LSTM neural networks. In this framework, we introduce a novel model based on LSTM layer that constructs two distributed representations for the given short text so that weighted combination of these representations can be used to achieve superior performance in classification task. One representation is obtained by mean pooling the output vectors from LSTM layer at each time step after processing of distributed vectors corresponding to words consecutively while the other one is acquired taking

average of word vectors directly. As a further extension, we derive distributed representations of the syllables for the first time in literature and use these representations in short text categorization task. Experimental results demonstrate that our novel model improves classification performance on our datasets and we show that using distributed representations for syllables also increases classification performance.

Bibliography

[1] V. Hatzivassiloglou and K. R. McKeown, "Predicting the semantic orientation of adjectives," in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98, (Stroudsburg, PA, USA), pp. 174–181, Association for Computational Linguistics, 1997.

[2] M. A. Hearst, "Text-based intelligent systems," ch. Direction-based Text Interpretation As an Information Access Refinement, pp. 257–274, Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1992.

[3] J. M. Wiebe, "Identifying subjective characters in narrative," in *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, (Stroudsburg, PA, USA), pp. 401–406, Association for Computational Linguistics, 1990.

[4] J. M. Wiebe, R. F. Bruce, and T. P. O'Hara, "Development and use of a gold-standard data set for subjectivity classifications," in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, (Stroudsburg, PA, USA), pp. 246–253, Association for Computational Linguistics, 1999.

[5] T. Nasukawa and J. Yi, "Sentiment analysis: Capturing favorability using natural language processing," in *Proceedings of the 2Nd International Conference on Knowledge Capture*, K-CAP '03, (New York, NY, USA), pp. 70–77, ACM, 2003.

[6] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the*

12th International Conference on World Wide Web, WWW '03, (New York, NY, USA), pp. 519–528, ACM, 2003.

[7] P. D. Turney, "Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, (Stroudsburg, PA, USA), pp. 417–424, Association for Computational Linguistics, 2002.

[8] J. Wiebe, "Learning subjective adjectives from corpora," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pp. 735–740, AAAI Press, 2000.

[9] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Found. Trends Inf. Retr.*, vol. 2, pp. 1–135, Jan. 2008.

[10] B. Liu, *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers, 2012.

[11] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, pp. 1–47, Mar. 2002.

[12] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," *CoRR*, vol. abs/1502.04681, 2015.

[13] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb 1989.

[14] M. Deshpande and G. Karypis, *Evaluation of Techniques for Classifying Biological Sequences*, pp. 417–431. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[15] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," *CoRR*, vol. abs/1303.5778, 2013.

[16] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014.

[17] R. Pascanu, T. Mikolov, and Y. Bengio, “On the diffsdsdiculty of training Recurrent Neural Networks,” *ArXiv e-prints*, Nov. 2012.

[18] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, Mar 1994.

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, Nov 1997.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *CoRR*, vol. abs/1301.3781, 2013.

[21] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *CoRR*, vol. abs/1310.4546, 2013.

[22] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[23] T. Kenter, A. Borisov, and M. de Rijke, “Siamese CBOW: optimizing word embeddings for sentence representations,” *CoRR*, vol. abs/1606.04640, 2016.

[24] X. Zhang and Y. LeCun, “Text understanding from scratch,” *CoRR*, vol. abs/1502.01710, 2015.

[25] K. Blekas, D. I. Fotiadis, and A. Likas, “Motif-based protein sequence classification using neural networks.,” *Journal of Computational Biology*, vol. 12, no. 1, pp. 64–82, 2005.

[26] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, “Fast time series classification using numerosity reduction,” in *In ICML06*, pp. 1033–1040, 2006.

[27] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, “Some effective techniques for naive bayes text classification,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 1457–1466, Nov 2006.

[28] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *J. Mach. Learn. Res.*, vol. 2, pp. 419–444, Mar. 2002.

[29] S. Zhu, X. Ji, W. Xu, and Y. Gong, "Multi-labelled classification using maximum entropy method," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, (New York, NY, USA), pp. 274–281, ACM, 2005.

[30] A. Sun and E.-P. Lim, "Hierarchical text classification and evaluation," in *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 521–528, 2001.

[31] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine Learning*, vol. 39, no. 2, pp. 103–134, 2000.

[32] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, pp. 1–47, Mar. 2002.

[33] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," *CoRR*, vol. cs.CL/0205070, 2002.

[34] M. T. Maybury, *New Directions in Question Answering*. AAAI Press, 2004.

[35] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[36] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, (London, UK, UK), pp. 137–142, Springer-Verlag, 1998.

[37] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, (Stroudsburg, PA, USA), pp. 90–94, Association for Computational Linguistics, 2012.

[38] A. Graves, M. Liwicki, S. Fernndez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 855–868, May 2009.

[39] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013.

[40] R. Józefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *CoRR*, vol. abs/1602.02410, 2016.

[41] J. Y. Lee and F. Dernoncourt, "Sequential short-text classification with recurrent and convolutional neural networks," *CoRR*, vol. abs/1603.03827, 2016.

[42] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM neural network for text classification," *CoRR*, vol. abs/1511.08630, 2015.

[43] K. Oflazer, "Two-level description of turkish morphology," *Literary and Linguistic Computing*, vol. 9, no. 2, pp. 137–148, 1994.

[44] D. Jurafsky and J. H. Martin, *SPEECH and LANGUAGE PROCESSING An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Second Edition*. Prentice Hall, 2009.

[45] A. Rajaraman and J. D. Ullman, *"Data Mining" Mining of Massive Datasets*. Cambridge University Press, 2011.

[46] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *in Knowledge Discovery and Data Mining*, pp. 245–250, ACM Press, 2001.

[47] S. Dasgupta and A. Gupta, "An elementary proof of the johnson-lindenstrauss lemma," tech. rep., 1999.

[48] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," in *Contemporary Mathematics*, vol. 26, (Providence, RI), p. 189206, American Mathematical Society, 1984.

[49] N. Vanli and S. Kozat, "A comprehensive approach to universal piecewise nonlinear regression based on trees," *Signal Processing, IEEE Transactions on*, vol. 62, pp. 5471–5486, Oct 2014.

[50] L. Bottou and C.-J. Lin, "Support vector machine solvers," in *Large Scale Kernel Machines* (L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, eds.), pp. 301–320, Cambridge, MA.: MIT Press, 2007.

[51] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," in *ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS*, pp. 573–582, 1994.

[52] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. NJ: John Wiley & Sons, 1980.

[53] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. `http://is.muni.cz/publication/884893/en`.

[54] M. Cetin and M. F. Amasyali, "Supervised and traditional term weighting methods for sentiment analysis," in *2013 21st Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, April 2013.

[55] H. ElSahar and S. R. El-Beltagy, *Building Large Arabic Multi-domain Resources for Sentiment Analysis*, pp. 23–34. Cham: Springer International Publishing, 2015.

[56] M. Salameh, S. Mohammad, and S. Kiritchenko, "Sentiment after translation: A case-study on arabic social media posts," in *HLT-NAACL*, 2015.

[57] M. Nabil, M. A. Aly, and A. F. Atiya, "LABR: A large scale arabic book reviews dataset," *CoRR*, vol. abs/1411.6718, 2014.

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[59] F. Chollet, "Keras." `https://github.com/fchollet/keras`, 2015.

[60] C. G. Weng and J. Poon, "A new evaluation measure for imbalanced datasets," in *Proceedings of the 7th Australasian Data Mining Conference - Volume 87*, AusDM '08, (Darlinghurst, Australia, Australia), pp. 27–32, Australian Computer Society, Inc., 2008.

[61] T. Fawcett, "An introduction to roc analysis," *Pattern Recogn. Lett.*, vol. 27, pp. 861–874, June 2006.

[62] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[63] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *CoRR*, vol. abs/1503.04069, 2015.

Appendix A

Forward Pass

We present the formulas for training of our model. We also provide formulas for forward pass for the sake of completeness.

Let m be the number of LSTM blocks and k the number of inputs. The following weights are obtained.

- Input weights: $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_c, \mathbf{W}_o \in \mathbb{R}^{m \times k}$
- Recurrent weights: $\mathbf{U}_i, \mathbf{U}_f, \mathbf{U}_c, \mathbf{U}_o \in \mathbb{R}^{m \times m}$
- Bias weights: $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_c, \mathbf{b}_o \in \mathbb{R}^m$

$\mathbf{x}^{1:l}$ are input vectors through LSTM layer where each one corresponds to a specific timestep. The symbols $\sigma(\cdot)$ and $\tanh(\cdot)$ refer to the elements wise sigmoid and

hyperbolic tangent functions, and $\odot$ is the element wise multiplication.

$$
\begin{aligned}
\mathbf{i}^t &= \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{h}^{t-1} + \mathbf{b}_i) \\
\tilde{\mathbf{c}}^t &= \tanh(\mathbf{W}_c\mathbf{x}^t + \mathbf{R}_c\mathbf{h}^{t-1} + \mathbf{b}_c) \\
\mathbf{f}^t &= \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{h}^{t-1} + \mathbf{b}_f) \\
\mathbf{c}^t &= \mathbf{f}^t \odot \mathbf{c}^{t-1} + \mathbf{i}^t \odot \tilde{\mathbf{c}}^t \\
\mathbf{o}^t &= \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{h}^{t-1} + \mathbf{b}_o) \\
\mathbf{h}^t &= \mathbf{o}^t \odot \tanh(\mathbf{c}^t) \\
\mathbf{s} &= \frac{1}{l}\sum_{t=1}^{l}\mathbf{h}^t \\
\mathbf{y} &= \frac{1}{l}\sum_{t=1}^{l}\mathbf{x}^t \\
\mathbf{z} &= \text{softmax}(\mathbf{W}_s\mathbf{s} + \mathbf{W}_y\mathbf{y} + \mathbf{b}_z)
\end{aligned}
$$

where softmax$(\mathbf{z})$ is defined as $\frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for $j = 1, \ldots, K$.

The final output $\mathbf{z}$ represents the probability distribution over the set of C classes. j^{th} element of z_i corresponds to the probability that i^{th} short text belongs to the j^{th} class.

Appendix B

Backward Pass

In this section we give the equations [63] used for backward pass:

$$\delta \mathbf{h}^t = \Delta^t + \mathbf{R}_i^T \delta \mathbf{i}^{t+1} + \mathbf{R}_i^T \delta \mathbf{i}^{t+1} + \mathbf{R}_f^T \delta \mathbf{f}^{t+1} + \mathbf{R}_o^T \delta \mathbf{o}^{t+1}$$

$$\delta \mathbf{o}^t = \delta \mathbf{h}^t \odot \tanh(\mathbf{c}^t) \odot \sigma'(\mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{h}^{t-1} + \mathbf{b}_o)$$

$$\delta \mathbf{c}^t = \delta \mathbf{h}^t \odot \mathbf{o}^t \odot \tanh'(\mathbf{c}^t) + \delta \mathbf{c}^{t+1} \odot \mathbf{f}^{t+1}$$

$$\delta \mathbf{f}^t = \delta \mathbf{c}^t \odot \mathbf{c}^{t-1} \odot \sigma'(\mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{h}^{t-1} + \mathbf{b}_f)$$

$$\delta \mathbf{i}^t = \delta \mathbf{c}^t \odot \tilde{\mathbf{c}}^{t-1} \odot \sigma'(\mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{h}^{t-1} + \mathbf{b}_i)$$

$$\delta \tilde{\mathbf{c}}_t = \delta \mathbf{c}^t \odot \mathbf{i}^{t-1} \odot \tanh'(\mathbf{W}_c \mathbf{x}^t + \mathbf{R}_c \mathbf{h}^{t-1} + \mathbf{b}_c)$$

$\Delta^t = \frac{\partial E}{\partial h^t}$ and $\Delta_j^t = \frac{\partial E}{\partial h_j^t} = \sum_{i=1}^{C} \frac{\partial E}{\partial \bar{z}_i} \frac{\partial \bar{z}_i}{\partial h_j^t}$

where $\bar{\mathbf{z}} = \mathbf{W}_s \mathbf{s} + \mathbf{W}_y \mathbf{y} + \mathbf{b}_z$.

$$
\begin{aligned}
\bar{z}_i &= \sum_{n=1}^{m} \mathbf{W}_{s_{in}} \left(\frac{1}{l} \sum_{p=1}^{l} \mathbf{h}_n^p \right) + \sum_{n=1}^{k} \mathbf{W}_{y_{in}} \left(\frac{1}{l} \sum_{p=1}^{l} \mathbf{x}_n^p \right) + \mathbf{b}_{z_i} \\
\frac{\partial E}{\partial \bar{z}_i} &= z_i - d_i \\
\frac{\partial z_i}{\partial h_j^t} &= \frac{1}{l} \sum_{n=1}^{m} \mathbf{W}_{s_{in}} \Bigg(\underbrace{\sum_{p=1}^{l} \frac{\partial h_n^p}{\partial h_j^t}}_{} \Bigg) \\
&\qquad \cancelto{0}{\frac{\partial h_n^1}{h_j^t}} + \ldots + \cancelto{0}{\frac{\partial h_n^{t-1}}{\partial h_j^t}} + \frac{\partial h_n^t}{\partial h_j^t} + \frac{\partial h_n^{t+1}}{\partial h_j^t} + \ldots + \frac{\partial h_n^l}{\partial h_j^t}, \\
\frac{\partial h_n^l}{\partial h_j^t} &= \sum_r \frac{\partial h_n^l}{\partial h_r^{l-1}} \sum_u \frac{\partial h_r^{l-1}}{\partial h_u^{l-2}} \cdots \sum_v \cdot \sum_d \frac{\partial h_v^{t+2}}{\partial h_d^{t+1}} \frac{\partial h_d^{t+1}}{\partial h_j^t} \qquad \text{for } l \geq t
\end{aligned}
$$

Therefore, if we know the partial derivative of output with respect to previous output we can calculate $\frac{\partial z_i}{\partial h_j^t}$ using chain rule. The partial derivative is given by

$$
\begin{aligned}
\frac{\partial h_i^t}{\partial h_j^{t-1}} = &\left[o_i^t (1 - o_i^t) \mathbf{R}_{o_{ij}} \right] \tanh(c_i^t) \\
&+ o_i^t \big[(1 - \tanh^2(c_i^t)) \big[f_i^t (1 - f_i^t) \mathbf{R}_{f_{ij}} c_i^{t-1} + (1 - (\tilde{c}_i^t)^2) \mathbf{R}_{c_{ij}} i_i^t \\
&\qquad + i_i^t (1 - i_i^t) \mathbf{R}_{i_{ij}} \tilde{c}_i^t \big] \big].
\end{aligned}
$$

The gradients for the weights are calculated by equations given below. $*$ can be any of $\{\tilde{\mathbf{c}}, \mathbf{i}, \mathbf{f}, \mathbf{o}\}$:

$$
\begin{aligned}
\delta \mathbf{W}_* &= \sum_{t=0}^{l} \langle \delta *^t, \mathbf{x}^t \rangle \\
\delta \mathbf{R}_* &= \sum_{t=0}^{l-1} \langle \delta *^{t+1}, \mathbf{h}^t \rangle \\
\delta \mathbf{b}_* &= \sum_{t=0}^{l} \delta *^t
\end{aligned}
$$

$\langle *_1, *_2 \rangle$ denotes the outer product of two vectors. The gradients for $\mathbf{W}_s$ and $\mathbf{W}_y$ are also given below:

$$\delta \mathbf{W}_{s_{ji}} = (z_i - d_i) s_j \qquad \delta \mathbf{W}_{y_{ji}} = (z_i - d_i) y_j$$
$$\delta \mathbf{W}_s = \langle \mathbf{s}, \mathbf{e} \rangle \qquad \delta \mathbf{W}_y = \langle \mathbf{y}, \mathbf{e} \rangle$$