

**ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES**

MSc THESIS

Funda GÜVEN

**USING TEXT REPRESENTATION AND DEEP LEARNING
METHODS FOR TURKISH TEXT CLASSIFICATION**

DEPARTMENT OF COMPUTER ENGINEERING

ADANA-2019

**ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES**

**USING TEXT REPRESENTATION AND DEEP LEARNING METHODS
FOR TURKISH TEXT CLASSIFICATION**

Funda GÜVEN

MSc THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on 21/06/2019.

.....
Prof. Dr. S.Ayşe ÖZEL
SUPERVISOR

.....
Asst. Prof. Dr. B.Melis ÖZYILDIRIM
MEMBER

.....
Asst. Prof. Dr. Mümine KAYA KELEŞ
MEMBER

This MSc Thesis is written at the Computer Engineering Department of Institute of Natural And Applied Sciences of Çukurova University.

Registration Number:

**Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences**

Not: The usage of the presented specific declarations, tables, figures, and photographs either in this thesis or in any other reference without citation is subject to "The law of Arts and Intellectual Products" number of 5846 of Turkish Republic

ABSTRACT

MSc THESIS

**USING TEXT REPRESENTATION AND DEEP LEARNING METHODS
FOR TURKISH TEXT CLASSIFICATION**

Funda GÜVEN

**ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING**

Supervisor : Prof. Dr. Selma Ayşe ÖZEL

Year: 2019, Pages: 115

Jury : Prof. Dr. Selma Ayşe ÖZEL

: Asst. Prof. Dr. Buse Melis ÖZYILDIRIM

: Asst. Prof. Dr. Mümine KAYA KELEŞ

The heavy use of the Internet has led to a significant increase in the amount of text content produced in online platforms. Huge amount of online textual data is difficult to process, and new techniques have begun to be developed to process online data automatically. New word and document representation methods and deep learning-based classifiers have emerged recently to work with large text datasets as an alternative way to traditional text processing methods. The vast majority of studies using these methods were done with English texts. For Turkish texts, these methods have been used in the last 2 or 3 years.

In this thesis, our aim is to evaluate the performances of new text representation and deep learning-based methods on classification of Turkish texts having different characteristics to show the usability of these methods on different document types. Therefore, these methods are used for the problems of sentiment and document classification and their performances are compared with traditional text classification methods. In order to make performance comparisons of the classifiers for the two text classification tasks that studied, deep learning-based convolutional neural networks and long short-term memory networks are used; as well as traditional classifiers which frequently used for Turkish texts in the literature. In the experimental evaluations it is found that embedding methods have similar performance with the traditional *tf* and *tf-idf* weighting methods, and in some cases achieve higher classification success. Deep learning-based classifiers have equal or higher classification success than the traditional classifiers.

Key Words: Word embedding, deep learning, sentiment analysis, document classification, Turkish text classification

ÖZ

YÜKSEK LİSANS TEZİ

TÜRKÇE METİN SINIFLAMA İÇİN METİN TEMSİLİ VE DERİN
ÖĞRENME YÖNTEMLERİNİN KULLANIMI

Funda GÜVEN

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Prof. Dr. Selma Ayşe ÖZEL

Yıl: 2019, Sayfa: 115

Jüri : Prof. Dr. Selma Ayşe ÖZEL

: Dr. Öğr. Üyesi Buse Melis ÖZYILDIRIM

: Dr. Öğr. Üyesi Mümine KAYA KELEŞ

İnternet kullanımının giderek yaygınlaşması, beraberinde dijital ortamlarda üretilen metin içeriği miktarında ciddi bir artışa neden olmuştur. Artan miktardaki metin verisini işlemek zorlaşmıştır ve bu ihtiyaca yönelik çözümler geliştirilmeye başlanmıştır. Geleneksel yöntemlere alternatif olarak daha büyük boyutlu verilerle çalışmayı mümkün kılan, derin öğrenme tabanlı sınıflayıcılar ve yapay sinir ağı tabanlı metin temsil yöntemleri geliştirilmiştir. Geliştirilen bu yöntemler kullanılarak yapılan çalışmaların büyük çoğunluğu İngilizce metinler ile yapılmıştır. Türkçe metinler için bu yöntemler son 2 ya da 3 yılda kullanılmaya başlanmıştır.

Bu tezde yapay sinir ağı tabanlı kelime ve doküman temsil yöntemleri ile derin öğrenme tabanlı sınıflayıcıların farklı karakteristiklere sahip Türkçe metinlerdeki sınıflama performanslarını değerlendirmek amacıyla, duygu ve doküman sınıflama problemleri için kullanılmış, geleneksel metin temsil yöntemleri ve sınıflayıcılar ile karşılaştırılmıştır. Sınıflayıcıların bu iki problem için karşılaştırmasını yapmak amacıyla, derin öğrenme tabanlı evrişimli sinir ağları, uzun kısa süreli bellek ağları ve literatürde Türkçe metinler için sıklıkla uygulanan geleneksel sınıflayıcılar kullanılmıştır. Yapılan deneyler sonucunda yapay sinir ağı bazlı metin temsil yöntemlerinin, *tf* ve *tf-idf* ağırlıklandırma yöntemlerinin başarılarına yakın ve bazı durumlarda daha yüksek sınıflama başarısı elde ettiği gözlenmiştir. Derin öğrenme tabanlı sınıflayıcılar ise geleneksel sınıflayıcılara eşit veya daha yüksek sınıflama başarısına sahip olmuştur.

Anahtar Kelimeler: Kelime yerleştirme, derin öğrenme, duygu analizi, doküman sınıflama, Türkçe metin sınıflama

EXTENDED SUMMARY

Due to the widespread use of the Internet and the increase in the number of users each day, the number of contents produced on online platforms has increased significantly. If the Information Society Statistics (2014-2018) is examined according to the use of information technologies in households statistic; Internet access in households, which was 7% in 2014, has reached to 83.8% in 2018; and total internet use was 18.8% in 2014, and it has increased to 72.9% in 2018. Taking into account these statistics, the increase in the number of contents produced can be perceived more concretely when a similar increase is observed worldwide. For this reason, the Internet can be thought of as a rich data source where people from all age groups, many social backgrounds, genders, and various professional groups come together to create and share content. In recent years, due to the increasing amount and variety of data, the number of scientific studies conducted using data such as text, photos, audio, video, which has been shared on the Internet, has increased.

Social media platforms such as Facebook and Twitter started to emerge when the Internet became widespread. Nowadays, the number of users of these platforms has reached millions by leaving behind the population of many countries. Studies on such a rich data source have also been diversified in terms of objectives and subjects. There are studies in quite different fields such as crime rate analysis (Aghababaei and Makrehchi, 2016), crisis management (Onorati et al., 2016), behavior analysis (Olivera et al., 2013), talent discovery (Davcheva, 2014) and periodic disease prediction (Lee et al., 2015).

Sentiment analysis is a natural language processing problem that is frequently studied on text data provided from these platforms. It can be defined as classifying the shared text into positive, negative, neutral or diversifiable categories. Most of the sentiment analysis studies in the literature have been conducted for English. Although the number of Turkish studies has increased in the last few years, there are still not enough studies and resources for Turkish.

In most of the sentiment analysis studies conducted for Turkish traditional classifiers such as Naive Bayes, decision support machines, random forests, decision trees are used with bag-of-words representation method. In the bag-of-words

representation method, documents are represented by large and sparse term-document matrices. The order of words is insignificant, and all words are used in document representation.

In recent years, the concept of deep learning has become widespread with the decrease in hardware costs and the usage of graphics processing units in calculations due to the development of technology. In this way, artificial neural network-based word and document representation methods, which are called word or document embedding methods, have been introduced. Differences of these methods from the traditional bag-of-words method are; these methods obtain smaller or denser vectors to represent words or documents and preserve semantic relations between words. Word2vec (Mikolov et al., 2013a), Doc2vec (Le and Mikolov, 2014), Fasttext (Bojanowski et al., 2016) and Glove (Pennington et al., 2014) are the most popular word and document embedding methods.

Word and document embedding methods are often the first steps in natural language processing with deep learning. Deep learning is used in the field of natural language processing for text classification, sentiment analysis, question answering, named entity recognition, machine translation and solving many different problems (Young et al., 2018). Two artificial neural network architectures commonly used for sentiment and document classification are convolutional neural networks and long short-term memory.

In the studies conducted with Turkish texts in the last two or three years, it is observed that; Hayran and Sert (2014), Ayata et al. (2017), Ay Karakuş et al. (2018), Şahin (2017) used Word2vec word representation method. In addition, there are other studies in the literature that use Word2vec, Doc2vec, and Glove representation methods as word and document representation methods and compare these methods with the traditional bag-of-words method. For deep learning-based approaches, Amasyalı et al. (2018) applied convolutional neural networks and long short-term memory artificial neural network architectures as classifiers. In addition to these two artificial neural network architectures, Ay Karakuş et al. (2018) achieved higher classification success with an artificial neural network architecture in which two architectures are used together.

In this thesis; based on the studies in the literature, two different classification problems have been studied for Turkish texts. First of all, the problem of two-class sentiment classification on social media messages containing short texts and frequently typographical errors is studied. Artificial neural network-based word and document representation methods and deep learning-based classifiers are compared with traditional text representation methods and classifiers. Then, in order to measure the effect of the characteristics of the dataset on the success of deep learning based methods; in contrast to the data of social media platform, the problem of classification of multi-class documents is studied also, in which the news texts consist of long documents containing longer sentences and rarely encountered spelling errors are used, and the results of classification problems are compared.

For the sentiment classification problem, experiments are done by using Turkish Sentiment Dataset (TSD), which contains thirty-two thousand tweets shared by Hayran and Sert (2017). First, preprocessing steps are performed on the TSD and the data set is represented by conventional *tf* and *tf-idf* weighting methods. Then, the classification process is performed by applying traditional classifiers. In the next step, Word2vec, Doc2vec, Fasttext, and Glove embedding vectors are trained for 10 iterations using 20 million tweet datasets shared by Kemik Natural Language Processing Group. Traditional classifiers are applied to the texts represented using these vectors. Embedding methods and bag-of-words method are compared for document representation. By using trained Word2vec, Fasttext and Glove vectors, convolutional neural networks, long short-term memory neural networks, and network structures obtained by combining the two architectures are trained. The results are compared with the traditional classifiers.

Experiments for the problem of classifying news documents are conducted by using the SuDer dataset shared by Şen and Yanıkoğlu (2018). Word2vec, Doc2vec, and Fasttext vectors are trained for 10 iterations using this data set, which includes more than 600,000 documents in total. Experiments are carried out separately for Cumhuriyet and Sabah sets, which are two different data sets. Both datasets are represented by using bag-of-words and embedding methods; and classified by traditional classifiers. The success of these two text representation methods for long texts such as news texts is compared. In this step, the most successful embedding

vectors are used for training the deep learning-based classifiers. The results obtained in this step are compared with the success of traditional methods.

For sentiment classification problem; it is observed that the classification accuracy of the word and document embedding methods trained on two different datasets are similar to those of traditional weighting methods (*tf* and *tf-idf*) and in some cases, their performances are higher. The deep learning-based classifiers' successes are closer to the traditional methods and higher when the single-layer LSTM architecture is used.

Word and document embedding vectors and traditional weighting methods trained in document classification problems have similar classification achievements. Deep learning-based classifiers, on the other hand, have higher or equal classification accuracy with traditional classifiers.

When two classification problems are considered together; word and document embedding methods can be considered as an alternative to the traditional term weighting methods. However; if embedding vectors learned for a problem similar to the problem studied are not readily available, it is necessary to consider the computational time and hardware cost required for the training of the vectors.

For both classification problems studied, deep learning-based classifiers have equal classification success or higher classification success than the traditional classifiers. Therefore, it can be concluded that deep learning-based classifiers are also successful for Turkish texts. But; the success assessment for these classifiers depends not only on the network architecture and the parameters used but also on the input of the classifiers.

Unlike traditional classifiers used in the study, a large number of parameters are needed to be adjusted during the training of deep learning-based classifiers. There does not exist publicly available optimal parameter values defined for all problems. These parameters need to be adjusted by observing the training process of artificial neural networks. Also; deep learning-based classifiers require more training data. Therefore, the cost of hardware and time required for the training of these classifiers are high and it must be considered before their usage.

GENİŞLETİLMİŞ ÖZET

İnternet kullanımının yaygınlaşması ve her geçen gün kullanıcı sayısındaki artış nedeniyle online platformlarda üretilen içerik sayısında da ciddi oranda artış görülmüştür. TÜİK'in hanelerde bilişim teknolojileri kullanımı istatistiklerine göre; 2014'te %7 olan internet erişimi 2018 yılında %83,8'e ve 2014 yılında %18,8 olan toplam internet kullanımı 2018 yılında %72,9'a ulaşmıştır (TÜİK– Bilgi Toplumu İstatistikleri 2014-2018). Bu istatistikleri göz önüne alarak Dünya çapında da benzer bir artış olduğu düşünüldüğünde üretilen içerik sayısının artışı da daha somut bir şekilde algılanabilir. Bu nedenle günümüzde İnternet; her yaş grubundan, birçok sosyal altyapıdan, farklı cinsiyetlerden, farklı meslek gruplarından insanın bir araya gelerek içerik oluşturup paylaştığı zengin bir veri kaynağı olarak düşünülebilir. Son yıllarda artan veri miktarı ve çeşitliliğine bağlı olarak İnternet ortamında paylaşılan metin, fotoğraf, ses, video gibi veriler kullanılarak yapılan bilimsel çalışma sayısı da artmıştır.

İnternetin yaygınlaşmasıyla Facebook ve Twitter gibi sosyal medya platformları da ortaya çıkmaya başlamıştır. Günümüzde kullanıcı sayıları milyonları aşarak birçok ülke nüfusunu geride bırakan bu platformlar zengin birer veri kaynağı haline gelmiştir. Bu platformlardan sağlanan veriler üzerinde; suç oranı tahmini (Aghababaei ve Makrehchi, 2016), kriz durumları yönetimi (Onorati ve ark., 2016), davranış analizi (Olivera ve ark., 2013), yetenek keşfi (Davcheva, 2014) ve periyodik hastalık tahmini (Lee ve ark., 2015) gibi çeşitli alanlarda çalışmalar yapılmaya başlanmıştır.

Duygu analizi bu platformlardan sağlanan metin verileri üzerinde sıklıkla çalışılan bir doğal dil işleme problemidir. Paylaşılan metnin olumlu, olumsuz, nötr ya da çeşitlendirilebilir kategorilere ayrılması olarak tanımlanabilir. Literatürdeki duygu analizi çalışmalarının büyük çoğunluğunda İngilizce için çalışılmıştır. Türkçe için yapılan çalışma sayısı son birkaç yılda artmış olsa da Türkçe için hala yeterli sayıda çalışma ve kaynak bulunmamaktadır.

Türkçe için yapılan duygu analizi çalışmalarının büyük bir kısmında kelime torbası temsil yöntemi ile Naive Bayes, karar destek makinaları, rastgele orman, karar ağaçları gibi geleneksel sınıflayıcılar kullanılmıştır. Kelime torbası temsil yönteminde

dokümanlar büyük ve seyrek terim doküman matrisleri ile temsil edilir. Kelimelerin sırası önemsizdir ve bütün kelimeler doküman temsilinde kullanılır.

Son yıllarda teknolojinin gelişimine bağlı olarak donanım maliyetlerinin azalması ve grafik işlem birimlerinin hesaplamalarda kullanılmaya başlanması ile derin öğrenme kavramı yaygınlaşmaya başlamıştır. Bu sayede, kelime ya da doküman yerleştirme (embedding) yöntemleri olarak adlandırılan yapay sinir ağı tabanlı kelime ve doküman temsil yöntemleri kullanılmaya başlanmıştır. Bu yöntemlerin geleneksel kelime torbası yönteminden farkları; kelime ya da dokümanları temsil etmek için daha küçük boyutlu ve sık vektörler elde etmeleri ve kelimeler arası anlam ilişkilerini gözetmeleridir. Word2vec (Mikolov ve ark., 2013a), Doc2vec (Le ve Mikolov, 2014), Fasttext (Bojanowski ve ark., 2016) ve Glove (Pennington ve ark., 2014) en popüler kelime ya da doküman yerleştirme yöntemlerindedir.

Kelime ya da doküman yerleştirme yöntemleri genellikle derin öğrenme ile doğal dil işlemenin ilk adımlarını oluşturmaktadır. Derin öğrenme doğal dil işleme alanında metin sınıflama, duygu sınıflama, soru cevaplama, varlık ismi tanıma, makine çevirisi ve farklı birçok problemin çözümüne yönelik kullanılmaktadır (Young ve ark., 2018). Duygu ve doküman sınıflama için sıklıkla kullanılan iki derin yapay sinir ağı mimarisi ise evrişimli yapay sinir ağları ve uzun kısa-süreli bellek yapay sinir ağlarıdır.

Son iki üç yıl içinde Türkçe metinler ile yapılan çalışmalara bakıldığında; Hayran ve Sert (2014), Ayata ve ark. (2017), Ay Karakuş ve ark. (2018), Şahin (2017) çalışmalarında Word2vec kelime temsil yönteminin Türkçe için kullanıldığı görülmektedir. Ayrıca literatürde kelime ve doküman temsil yöntemleri olarak Word2vec, Doc2vec ve Glove temsil yöntemlerini kullanan ve bu yöntemleri geleneksel kelime torbası yöntemi ile karşılaştıran başka çalışmalar da bulunmaktadır. Derin öğrenme tabanlı yaklaşımlarda örneğin Amasyalı ve ark. (2018) sınıflayıcı olarak evrişimli yapay sinir ağları ve uzun kısa-süreli bellek yapay sinir ağı mimarilerini uygulamıştır. Ay Karakuş ve ark. (2018) ise çalışmalarında bu iki yapay sinir ağı mimarisine ek olarak iki mimarinin birlikte kullanıldığı bir yapay sinir ağı mimarisi kullanmış ve daha yüksek sınıflama başarısına ulaşmıştır.

Bu tezde; literatürdeki çalışmalardan yola çıkılarak Türkçe metinler için iki farklı sınıflama problemi çalışılmıştır. Öncelikle kısa metinler içeren ve yazım yanlışlarına sıklıkla rastlanan sosyal medya mesajları üzerinde iki sınıflı duygu analizi

problemi ele alınmıştır. Kullanılan yapay sinir ağı tabanlı kelime ve doküman temsil yöntemlerinin ve derin öğrenme tabanlı sınıflayıcıların geleneksel metin temsil yöntemleri ve sınıflayıcılarla karşılaştırması yapılmıştır. Daha sonra çalışılan veri kümesinin karakteristik özelliklerinin derin öğrenme tabanlı yöntemlerdeki başarıya etkisini ölçmek amacıyla; sosyal medya platformu verilerinin tersine daha uzun cümleler içeren uzun dokümanlardan oluşan ve yazım yanlına çok az rastlanan haber metinleri üzerinde çok sınıflı doküman sınıflama problemi üzerinde çalışılıp, sonuçlar karşılaştırılmıştır.

Duygu sınıflama probleminde, deneyler Hayran ve Sert (2017) tarafından paylaşılan otuz iki bin tweet içeren Turkish Sentiment Dataset (TSD) kullanılarak gerçekleştirilmiştir. Öncelikle TSD üzerinde ön işleme adımları uygulanıp, veri kümesi geleneksel *tf* ve *tf-idf* ağırlıklandırma yöntemleri ile temsil edilmiştir. Ardından geleneksel sınıflayıcılar uygulanarak sınıflama işlemi yapılmıştır. Sonraki adımda TSD ve Kemik Doğal Dil İşleme Grubu'nun paylaştığı 20 milyon tweet veri kümeleri kullanılarak 10 iterasyon boyunca Word2vec, Doc2vec, Fasttext ve Glove yerleştirme vektörleri eğitilmiştir. Bu vektörler kullanılarak temsil edilen metinler üzerinde geleneksel sınıflayıcılar uygulanmıştır. Elde edilen sonuçlar doğrultusunda yerleştirme yöntemleri ile kelime torbası yönteminin doküman temsili için başarı karşılaştırılması yapılmıştır. Eğitilen Word2vec, Fasttext ve Glove vektörleri kullanılarak evrişimli sinir ağları, uzun kısa-süreli bellek sinir ağları ve iki mimari birlikte kullanılarak elde edilen ağ yapıları ile tekrar sınıflama yapılmıştır. Buradan elde edilen sonuç da geleneksel sınıflayıcılar ile karşılaştırılmıştır.

Haber dokümanlarını sınıflama problemi için deneyler Şen ve Yanıkoğlu (2018) tarafından paylaşılan SuDer veri kümesi kullanılarak gerçekleştirilmiştir. Toplamda 600.000'den fazla sayıda doküman içeren bu veri kümesi kullanılarak 10 iterasyon boyunca Word2vec, Doc2vec ve Fasttext vektörleri eğitilmiştir. Veri kümesinin içerdiği iki farklı veri kümesi olan Cumhuriyet ve Sabah kümeleri için deneyler ayrı ayrı uygulanmıştır. Her iki veri kümesi kelime torbası yöntemi ve yerleştirme yöntemleri ile temsil edilerek geleneksel sınıflayıcılar ile sınıflanmıştır. Bu iki temsil yönteminin haber metinleri gibi uzun metinler için başarısı kıyaslanmıştır. Bu adımda en başarılı bulunan yerleştirme vektörleri ile derin öğrenme tabanlı

sınıflayıcılar kullanılarak sınıflama yapılmıştır. Bu adımda elde edilen sonuçlar ile geleneksel yöntemlerin başarı karşılaştırması yapılmıştır.

Duygu analizi problemi için; iki farklı veri kümesi üzerinden eğitilen kelime ve doküman yerleştirme yöntemlerinin sınıflama başarıları, geleneksel ağırlıklandırma yöntemlerinin (*tf* ve *tf-idf*) başarılarına benzer ve bazı durumlarda daha yüksek olarak gözlenmiştir. Kullanılan derin öğrenme tabanlı sınıflayıcılar ise geleneksel sınıflama yöntemlerine yakın ve tek katmanlı LSTM mimarisi kullanıldığı durumda daha yüksek sınıflama başarısı elde edilmiştir.

Doküman sınıflama probleminde eğitilen kelime ve doküman yerleştirme vektörleri ile geleneksel ağırlıklandırma yöntemleri benzer sınıflama başarıları elde etmiştir. Derin öğrenme tabanlı sınıflayıcılar ise geleneksel sınıflayıcılara eşit ve daha yüksek sınıflama başarılarına ulaşmıştır.

İki problem birlikte düşünüldüğünde; kelime ve doküman yerleştirme yöntemleri, geleneksel ağırlıklandırma yöntemlerine bir alternatif olarak düşünülebilir. Fakat; çalışılan probleme benzer bir problem için öğrenilen hazır kelime ya da doküman yerleştirme vektörleri bulunmuyorsa, vektörlerin eğitimi için gereken hesaplama zamanını ve donanım maliyetini göz önünde bulundurmak gerekmektedir.

Çalışılan her iki problem için derin öğrenme tabanlı sınıflayıcılar, geleneksel sınıflayıcılara eşit ve daha yüksek sınıflama başarıları elde etmiştir. Bu nedenle Türkçe metinler için de derin öğrenme tabanlı sınıflayıcıların başarılı olduğu sonucuna varılabilir. Fakat; bu sınıflayıcılar için başarı değerlendirmesi sadece ağ mimarisi ve kullanılan parametrelere değil aynı zamanda sınıflayıcılara verilen girdiye de bağlıdır.

Çalışmada kullanılan geleneksel sınıflayıcılardan farklı olarak derin öğrenme tabanlı sınıflayıcıların eğitimi sırasında çok sayıda parametrenin ayarlanması gerekmektedir. Problemlere özel tanımlı parametreler bulunmamaktadır. Bu parametreler yapay sinir ağlarının eğitimi gözlemlenerek ayarlanmak durumundadır. Ayrıca; derin öğrenme tabanlı sınıflayıcılar daha fazla eğitim verisi gerektirmektedir. Bu nedenle bu sınıflayıcıların eğitimi için gereken donanım ve zaman maliyeti de artmaktadır.

ACKNOWLEDGEMENTS

I am grateful to my supervisor, Prof. Dr. Selma Ayşe ÖZEL, whose encouragement, guidance and support motivated me in research and study of the thesis.

It is a pleasure to thank each and all members of the evaluation committee for their guidance.

I would like to extend my gratitude to my parents and my brother for their belief, support, patience, motivation, and encouragement.

Last but not least, I would like to thank İsmail Mert AY AK for his endless motivation and support.

| CONTENTS | PAGE |
|--|-------------|
| ABSTRACT..... | I |
| ÖZ | II |
| EXTENDED SUMMARY..... | III |
| GENİŞLETİLMİŞ ÖZET | VII |
| ACKNOWLEDGEMENTS | XI |
| CONTENTS..... | XII |
| LIST OF TABLES | XVI |
| LIST OF FIGURES | XVIII |
| LIST OF ABBREVIATIONS | XX |
| 1. INTRODUCTION | 1 |
| 2. RELATED WORKS | 5 |
| 2.1. Studies in English | 5 |
| 2.2. Studies in Other Languages | 7 |
| 2.3. Studies in Turkish | 8 |
| 3. MATERIALS AND METHODS..... | 13 |
| 3.1. Text Representation Methods | 13 |
| 3.1.1. Term Weighting Methods..... | 13 |
| 3.1.1.1. Tf (term frequency)..... | 13 |
| 3.1.1.2. Tf-idf (term frequency-inverse document frequency)..... | 14 |
| 3.1.2. Embedding Methods | 14 |
| 3.1.2.1. Word2vec..... | 15 |
| 3.1.2.2. Doc2vec | 17 |
| 3.1.2.3. Fasttext..... | 18 |
| 3.1.2.4. Glove..... | 19 |
| 3.2. Text Classification Methods | 19 |
| 3.2.1. Traditional Classification Methods..... | 19 |
| 3.2.1.1. Naive Bayes Classifier | 19 |

| | |
|--|----|
| 3.2.1.2. Support Vector Machines | 21 |
| 3.2.1.3. Random Forest | 21 |
| 3.2.1.4. Logistic Regression..... | 22 |
| 3.2.2. Deep Learning-based Classification Methods | 22 |
| 3.2.2.1. Convolutional Neural Networks | 23 |
| 3.2.2.1.(1). Convolution Layer | 23 |
| 3.2.2.1.(2). Activation Functions | 25 |
| 3.2.2.1.(3). Stride and Padding Operations | 27 |
| 3.2.2.1.(4). Pooling Layer | 28 |
| 3.2.2.1.(5). Output Layer | 28 |
| 3.2.2.2. Recurrent Neural Networks | 30 |
| 3.2.2.3. Long Short-Term Memory Networks | 31 |
| 3.3. Text Classification | 33 |
| 3.4. Datasets..... | 35 |
| 3.5. Preprocessing..... | 38 |
| 3.6. Feature Engineering..... | 39 |
| 3.6.1. Applying Tf and Tf-idf | 40 |
| 3.6.2. Training Embedding Methods | 40 |
| 3.6.3. Usage of Embedding Vectors for Document Representations..... | 44 |
| 3.7. Classification Process | 47 |
| 3.7.1. Parameter Settings for Traditional Classifiers | 47 |
| 3.7.2. Using Deep Learning-based Classifiers..... | 49 |
| 3.7.3. Deep Learning Models for Sentiment Analysis Task | 52 |
| 3.7.4. Deep Learning Models for Document Classification Task..... | 59 |
| 4. RESULTS AND DISCUSSION..... | 65 |
| 4.1. Evaluation Metrics | 65 |
| 4.2. Classification Performance Measurement for Deep Learning-Based Classifiers | 68 |
| 4.3. Results for Sentiment Classification..... | 69 |

| | |
|---|-----|
| 4.3.1. Performance of Traditional Classifiers | 69 |
| 4.3.2. Performance of Deep Learning based Classifiers | 78 |
| 4.4. Results for Document Classification | 82 |
| 4.4.1. Performance of Traditional Classifiers | 82 |
| 4.4.2. Performance of Deep Learning based Classifiers | 89 |
| 4.5. Comparison of the Used Methods | 92 |
| 4.6. Comparison of Results with Studies using the Same Datasets | 93 |
| 4.6.1. Comparison of Results for Sentiment Analysis | 93 |
| 4.6.2. Comparison of Results for Document Classification..... | 96 |
| 5. CONCLUSIONS..... | 99 |
| REFERENCES | 101 |
| BIOGRAPHY | 107 |



| LIST OF TABLES | PAGE |
|---|-------------|
| Table 3.1. Class distribution of Turkish Sentiment Dataset | 36 |
| Table 3.2. Class distribution of SuDer dataset | 37 |
| Table 3.3. Class distribution of Cumhuriyet dataset | 38 |
| Table 3.4. Class distribution of Sabah dataset..... | 38 |
| Table 3.5. Grid search parameters | 48 |
| Table 3.6. Classifier parameters for document classification task | 48 |
| Table 4.1. Experimental results of traditional classifiers using bag-of-words method for sentiment analysis | 70 |
| Table 4.2. Experimental results of traditional classifiers using Word2vec embedding vectors trained from TSD | 71 |
| Table 4.3. Experimental results of traditional classifiers using Word2vec embedding vectors trained from 20M tweets dataset | 72 |
| Table 4.4. Experimental results of traditional classifiers using Doc2vec embedding vectors trained from TSD | 74 |
| Table 4.5. Experimental results of traditional classifiers using Doc2vec embedding vectors trained from 20M tweets | 74 |
| Table 4.6. Experimental results of traditional classifiers using Fasttext embedding vectors trained from TSD | 75 |
| Table 4.7. Experimental results of traditional classifiers using Fasttext embedding vectors trained from 20M tweets | 76 |
| Table 4.8. Experimental results of traditional classifiers using Glove embedding vectors trained from TSD | 77 |
| Table 4.9. Experimental results of traditional classifiers using Glove embedding vectors trained from 20M tweets | 78 |
| Table 4.10. Experimental results of CNN for TSD | 79 |
| Table 4.11. Experimental results of CNN3 for TSD | 80 |
| Table 4.12. Experimental results of LSTM for TSD..... | 81 |

| | |
|---|----|
| Table 4.13. Experimental results of CNN-LSTM for TSD | 82 |
| Table 4.14. Experimental results of CNN-LSTM2 for TSD | 82 |
| Table 4.15. Experimental results of traditional classifiers using BOW method for Cumhuriyet dataset | 83 |
| Table 4.16. Experimental results of traditional classifiers using BOW method for Sabah dataset | 84 |
| Table 4.17. Experimental results of traditional classifiers using Word2vec method for Cumhuriyet dataset | 85 |
| Table 4.18. Experimental results of traditional classifiers using Word2vec method for Sabah dataset | 86 |
| Table 4.19. Experimental results of traditional classifiers using Fasttext method for Cumhuriyet dataset | 87 |
| Table 4.20. Experimental results of traditional classifiers using Fasttext method for Sabah dataset | 87 |
| Table 4.21. Experimental results of traditional classifiers using Doc2vec method for Cumhuriyet dataset | 88 |
| Table 4.22. Experimental results of traditional classifiers using Doc2vec method for Sabah dataset | 88 |
| Table 4.23. Class distribution of 25% of Sabah dataset | 90 |
| Table 4.24. Experimental results of traditional classifiers on 25% of Sabah dataset using BOW method..... | 90 |
| Table 4.25. Experimental results of deep learning-based classifiers on Cumhuriyet dataset..... | 91 |
| Table 4.26. Experimental results of deep learning-based classifiers on 25% Sabah dataset | 91 |

| LIST OF FIGURES | PAGE |
|--|-------------|
| Figure 3.1. Word2vec CBOW architecture..... | 16 |
| Figure 3.2. Word2vec skip-gram architecture | 16 |
| Figure 3.3. Doc2vec PV-DM architecture | 18 |
| Figure 3.4. Doc2vec PV-DBOW architecture | 18 |
| Figure 3.5. SVM | 21 |
| Figure 3.6. Basic CNN structure for sentence classification | 23 |
| Figure 3.7. Sparse connection (top) and dense connection (bottom)..... | 24 |
| Figure 3.8. ReLU activation function | 25 |
| Figure 3.9. Sigmoid activation function | 26 |
| Figure 3.10. Tanh activation function..... | 26 |
| Figure 3.11. Softmax activation function | 27 |
| Figure 3.12. Stride operation | 27 |
| Figure 3.13. Max-pooling operation | 28 |
| Figure 3.14. An example of CNN for sentence classification | 29 |
| Figure 3.15. RNN blocks | 30 |
| Figure 3.16. LSTM architecture | 32 |
| Figure 3.17. Methodology of sentiment classification task | 34 |
| Figure 3.18. Methodology of document classification task..... | 35 |
| Figure 3.19. An example of Gensim Word2vec input format | 41 |
| Figure 3.20. An example for Gensim Doc2vec input format | 42 |
| Figure 3.21. Parameters in Glove demo.sh file..... | 43 |
| Figure 3.22. Documents representations with word embedding for the sentiment classification task..... | 44 |
| Figure 3.23. Documents representations with word embedding for the document classification task..... | 45 |
| Figure 3.24. Sum representation model for embedding..... | 45 |
| Figure 3.25. Avg representation model for embedding | 46 |

| | |
|---|----|
| Figure 3.26. Var representation model for embedding | 47 |
| Figure 3.27. Input format for deep learning-based classifiers | 50 |
| Figure 3.28. Distribution of the number of words in documents from Cumhuriyet dataset..... | 51 |
| Figure 3.29. Distribution of the number of words in documents from Sabah dataset..... | 51 |
| Figure 3.30. CNN1 architecture for sentiment classification task | 53 |
| Figure 3.31. CNN3 architecture for sentiment classification task | 55 |
| Figure 3.32. LSTM architecture for sentiment classification task..... | 56 |
| Figure 3.33. CNN-LSTM architecture for sentiment classification task | 57 |
| Figure 3.34. CNN-LSTM2 architecture for sentiment classification task | 58 |
| Figure 3.35. CNN3 architecture for document classification task..... | 60 |
| Figure 3.36. LSTM architecture for document classification task..... | 61 |
| Figure 3.37. CNN-LSTM architecture for document classification task..... | 62 |
| Figure 3.38. CNN-LSTM2 architecture for document classification task..... | 63 |
| Figure 4.1. Confusion matrix..... | 65 |
| Figure 4.2. An example of classification output of sentiment classification task | 67 |
| Figure 4.3. An example of classification output of document classification task | 67 |
| Figure 4.4. Result of a deep learning model on test data..... | 69 |
| Figure 4.5. Screenshot for GPU usage on AWS..... | 90 |

LIST OF ABBREVIATIONS

| | |
|--------|---|
| ANN | : Artificial Neural Network |
| BiLSTM | : Bidirectional Long Short-Term Memory |
| CBOW | : Continuous Bag-Of-Words |
| CC | : Centroid Classifier |
| CFSVM | : Centroid Feature Support Vector Machine |
| CNN | : Convolutional Neural Network |
| DBOW | : Distributed Bag-Of-Words |
| DCNN | : Dynamic Convolutional Neural Network |
| DM | : Distributed Memory |
| DT | : Decision Tree |
| GPU | : Graphics Processing Unit |
| GRU | : Gated Recurrent Unit |
| HDNN | : Hierarchical deep neural network |
| HS | : Hierarchical Softmax |
| KNN | : K-Nearest Neighbors |
| LR | : Logistic Regression |
| LSTM | : Long Short-Term Memory |
| ME | : Maximum entropy |
| MLP | : Multilayer Perceptron |
| NB | : Naïve Bayes |
| NBM | : Naïve Bayes Multinomial |
| NC | : Nearest Centroid |
| NLP | : Natural Language Processing |
| NS | : Negative Sampling |
| ReLU | : Rectified Linear Unit |
| RF | : Random Forest |

RNN : Recurrent Neural Network
SVC : C-Support Vector Machines
SVM : Support Vector Machine
TSD : Turkish Sentiment Dataset



1. INTRODUCTION

With the increasing use of the Internet and the growing number of users every day, the amount of digital contents produced and shared in online environments has grown tremendously. If the Information Technology Usage Statistics in Households (2014–2018) for Turkey is examined, it can be observed that the increase in the Internet usage in our country, such that while Internet access in households was 7% in 2004, it has reached to 83.8% in 2018; and total Internet usage was 18.8% in 2004 and it has increased to 72.9% in 2018. If we consider that the increase in the world is like in our country, it can be said that the amount of digital content is rising rapidly due to the increase in the number of users on online platforms. Accordingly, Internet can be seen as a rich source of data from users of many different social classes, different age groups, genders, and various professional groups. Depending on the growth rate of the online resources, there has been a great increase in the studies on the online content such as text, photo, audio and video which have been created and shared in online platforms in recent years. As the number of shared data increases, it has become necessary to process the data automatically in order to use and extract meaningful data.

Social media platforms such as Facebook and Twitter started to emerge when the Internet became widespread. Nowadays, the number of users of these platforms has reached millions by leaving behind the population of many countries. Studies on such a rich data source have also been diversified in terms of objectives and subjects. There are studies in quite different fields such as crime rate analysis (Aghababaei and Makrehchi, 2016), crisis management (Onorati et al., 2016), behavior analysis (Olivera et al., 2013), talent discovery (Davcheva, 2014) and periodic disease prediction (Lee et al., 2015).

Sentiment analysis through social media platforms is one of the popular research topics of today because it is contemporary and contains fertile content. In sentiment analysis, sentiments and thoughts are determined through a text. The

majority of the work done is for English texts and the number of studies for Turkish texts is limited.

Sentiment analysis is treated as a two-class (positive, negative) or three-class (positive, negative, neutral) classification problem. It basically consists of four phases; (i) data collection, (ii) data pre-processing, (iii) selection of features, (iv) classification. In most of the studies for Turkish, classical machine learning algorithms such as Naïve Bayes, support vector machine, maximum entropy, k-nearest neighbor and decision trees have been applied for classification in the learning phase of the model. Studies on texts that are in languages different from Turkish have revealed that deep learning algorithms are more successful than the classical machine learning algorithms. Therefore, more recent studies have started to apply deep learning techniques to make sentiment analysis from Turkish texts.

Before applying deep learning-based classifiers, text documents must be converted into numeric vectors by using some word or document embedding methods. In the last few years, in the sentiment analysis studies conducted for Turkish texts, word embedding methods such as Word2vec were used as the method of text representation (Hayran and Sert, 2017; Ayata et al., 2017; Ay Karakuş et al., 2018). Although there are studies comparing bag-of-words with Fasttext, Word2vec, Doc2vec methods separately as a method of text representation, there is no study presenting a comparison of all of these word or document representation methods for Turkish texts. In this thesis, a comparison of traditional *tf* and *tf-idf* weighting methods with Word2vec, Doc2vec, Fasttext, and Glove embedding vectors that are trained on Turkish Twitter messages is made for sentiment classification problem.

In order to compare the embedding methods, the classifiers used for the studies that carried out sentiment classification in Turkish texts are examined. It is observed that Naive Bayes Multinomial, Support Vector Machines, Random Forest, Logistic Regression, K-nearest Neighbor classifiers are used until recently. In recent years, studies have been carried out by comparing Convolutional Neural Networks, Long Short-Term Memory (Amasyalı et al., 2018), neural networks (Şen and Yanıkoğlu, 2018), and multiple deep learning architectures (Ay Karakuş

et al., 2018). Based on these studies, *tf*, *tf-idf*, Word2vec, Doc2vec, Fasttext, and Glove methods are used for document representation; and Naive Bayes Multinomial (NBM), Support Vector Machines (SVM), Random Forest (RF), Logistic Regression (LR), Convolutional Neural Networks (CNN), Long Short Term Memory (LSTM), and a hybrid network architecture that uses both CNN and LSTM architectures are applied to make classification for sentiment analysis problem.

In this sense; this thesis is the first comprehensive study comparing 4 different embedding methods for Turkish sentiment classification problem. Also, a detailed comparison of traditional classifiers and deep learning-based classifiers are made for sentiment classification problems.

The data set that used for the sentiment classification problem consisted of Turkish Twitter messages (Hayran and Sert, 2017). For this reason, the instances in our dataset have frequent typographical errors and consist of short-length texts. After experiments for the problem of sentiment analysis; the results of the embedding methods and the deep learning-based classifiers for a dataset that has opposite characteristics of the used dataset are evaluated. Based on this idea, the problem of document classification is included in this thesis by using SuDer (Şen and Yanıkoğlu, 2018) dataset which consists of the news texts from the two major Turkish newspapers.

The problem of document classification can be defined as labeling automatically the documents into pre-determined categories to facilitate access to the searched information. As the number of documents on the web increases day by day, this classification task becomes more challenging nowadays. Since 2017, there have been studies using the word and document vectors generated by the embedding methods that are Word2vec (Şahin, 2017; Şen and Yanıkoğlu, 2018), and Doc2vec (Çelenli, 2018; Bilgin and Şentürk, 2017) for Turkish document classification problem. However, there is no study comparing Word2vec, Doc2vec and Fasttext methods. For this reason, in this thesis, these 3 methods are compared with the traditional *tf* and *tf-idf* weighting methods. For the document classification problem, Naive Bayes Multinomial, Logistic Regression, Random Forest,

Convolutional Neural Networks, Long Short-Term Memory, and two hybrid artificial neural network structures that are created by using the two architectures are used as classifiers and their performances are compared.

In summary, in this thesis, experiments are conducted for two different classification tasks: sentiment analysis and document classification. For the sentiment classification task; word vectors are obtained for four different embedding methods over 2 corpora, one with 32 thousand tweets and the other with 20 million tweets. In the text representation, a comparison of the *tf*, *tf-idf* weighting methods with the embedding methods is made. In this comparison, both traditional classifiers and deep learning based convolutional neural network and long short-term memory artificial neural network architectures are used. For the document classification task, Word2vec, Doc2vec, and Fasttext vectors are obtained on a corpus with more than 600,000 news. The effect of the use of these vectors on text representation are compared with the *tf* and *tf-idf* weighting methods. For the comparison, traditional classifiers which have been frequently used when working with Turkish texts, and deep learning-based classifiers are used as did in the sentiment classification experiments. Therefore, this thesis makes a detailed performance comparison of word and document embedding methods with the traditional bag-of-words model for short and long text classification problems for Turkish. Also, a comparison of deep learning-based classifiers with the traditional ones is made for Turkish text classification problems.

The rest of this thesis is organized as follows: in the next section a brief summary of the related work is given, in section 3 background information about the text representation methods and classifiers are presented. Section 3 also covers other materials and methods used in this thesis. Section 4 presents the experimental results and discussions. Finally, section 5 concludes our study.

2. RELATED WORKS

In this section, the text classification studies that use word and document embedding methods, and deep learning classifiers are summarized for several natural languages.

2.1. Studies in English

There are many studies in the literature that use text embedding methods which are popular in recent years. In addition to traditional machine learning methods, CNN, LSTM, and many similar neural networks are used in the studies. Although most of the studies have been done for English texts, the number of studies for other languages has increased in recent years.

Kalchbrenner et al. (2014) described a CNN architecture (DCNN) using dynamic k -max pooling and variable-length input for semantic modeling of sentences. The defined network structure consists of the following layers; dynamic k -max pooling applied dynamic pooling layers, non-linear feature function, multiple feature maps, and folding layer. Several experiments have been performed to test this defined network structure. These experiments include binary and multi-class sentiment analysis on film reviews with Stanford Sentiment Treebank, question type classification on TREC question data, and sentiment prediction from Twitter messages with distant supervision. According to the results of the experiments, the proposed network structure has achieved the highest success among the studies conducted with these data sets in sentiment classification and question type classification.

In the same year, Kim (2014), who used CNN for sentence classification, studied the most popular dataset MR, SST-1, SST-2, Subj, TREC, CR, and MPQA, and used a simpler network architecture than the previous study. Kim (2014) applied CNN with different variations; rand, static, non-static and multichannel,

and used pre-trained Word2vec vectors in variations other than the rand. As a result of the experiments, the highest classification performance is achieved in 4 of the 7 datasets. Although the network architecture is simpler than the network architecture suggested by Kalchbrenner et al. (2014), the proposed network has been more successful on TREC data used by two studies.

Severyn and Moschitti (2015) used word embedding methods and CNN to apply the message-level and phrase-level sentiment analysis on the Semeval-15 corpus. During the experimental analysis; random values, Word2vec vectors trained on 50 million tweets, and Word2vec vectors with distant supervision are used for network parameters. As a result of the experiments; the proposed method for phrase-level sentiment analysis is found as best, and sentence-level sentiment analysis is observed as the second-best method for sentiment analysis.

Artificial neural networks make it easier to work with large data; moreover, there are studies that prove that the performance of the classification increases as the data size increases. One of these studies was conducted by Hu et al. (2015) who proposed HDNN architecture for document-level sentiment analysis in large-scale data. The combination of word frequencies, contextual window, and POS tagging features are given as input to the artificial neural network. In the experiments, electronic product reviews from Amazon, film reviews from Amazon and IMDB, and hotel comments from TripAdvisor are classified with HDNN, SVM, and NB classifiers. The experiments are repeated for different sizes of each dataset. HDNN has achieved the highest classification success for all datasets. As the size of the data increases, there is an increase in the classification success of HDNN for the 3 datasets. It has been concluded that DNN solves both large-scale data problems and domain dependency problem.

In addition to usage of a single network structure, deep artificial neural network architectures can be used together. Hassan and Mahmood (2017) proposed an artificial neural network called ConvLstm where CNN and LSTM architectures are used together. In this proposed architecture, CNN is used to extract properties

from input data; and the LSTM is used instead of the pooling layer to remember important information and capture long-term dependencies. In the study; experiments are conducted for sentiment analysis using pre-trained Word2vec vectors on IMDB and Stanford Sentiment Treebank. The results obtained are found to be more successful compared to the previous studies using these datasets. As a result of the experiments; this network architecture, which is trained with fewer parameters, has been accepted as an alternative to other methods by achieving higher success than other models.

2.2. Studies in Other Languages

Before studying for Turkish, the studies in different languages other than English are examined in order to see how the word embedding and deep learning methods are used in these languages and their classification performances. In their study, Yang and Xia (2016) use a CNN architecture, the last layer of which is a linear classifier, to classify documents according to their sentiments by using the Word2vec method. They use a collection of Chinese hotel reviews corpus containing 4 different data sets (3 balanced, 1 unbalanced class distribution). To classify documents; a CNN architecture, SVM and NBM classifiers are used. At the end of the experiments, the worst results are obtained with NBM in all datasets including the unbalanced data set, while the best results are observed with CNN.

Another study done for Chinese belongs to Huang et al. (2017) who obtains the word vectors by using Word2vec on the Chinese micro-blog data and apply CNN, LSTM, a single layer CNN and a network structure, and SVM with two LSTM layers as classifiers. According to the results of the sentiment classification; the proposed method (one-layer CNN + 2-layer LSTM) has 87.2% accuracy, while SVM, CNN, CNN-LSTM (each single layer), and LSTM have 86%, 85.6%, 84.2%, and 83.8% accuracies, respectively. As a result; it is concluded that CNN or LSTM is less successful than the hybrid models.

Vateekul and Koomsubha (2016) have made the first sentiment analysis on Thai Twitter messages using Word2vec word vectors, with DCNN and LSTM deep learning models. In the study; SAE, NB, SVM and ME classifiers are used in addition to two deep neural networks, and the results are compared. It is observed that the highest classification success belongs to DCNN with 75.35% accuracy, while the classification accuracy of LSTM is 75.30%, and other classifiers are behind them.

Vo et al. (2017) worked on 2 different Vietnamese Twitter datasets. CNN, LSTM, multi-channel CNN-LSTM, and SVM classifiers are used. SVM is implemented with attributes obtained by using bag-of-words; for other classifiers, the attributes from the embedding layer of the proposed method are used. When the classification performances are compared, the highest success is obtained by using CNN and LSTM together.

2.3. Studies in Turkish

In the literature, there are some studies that applied sentiment and document classification by using machine learning methods and deep learning methods on the Turkish texts represented with traditional representation methods and word embedding methods.

Şen and Erdoğan (2014) used two data sets, including 52 million words Wikipedia dataset, and the text dataset created by Boğaziçi University. In the study, the skip-gram model of Word2vec word embedding method is selected. Word vectors are obtained by using two algorithms: negative sampling and hierarchical softmax. In the study, these two methods are compared, and negative sampling is found to be more successful. In addition, the effect of vector size on semantic and syntactic accuracy is measured and the highest success is obtained in the range of 200-400.

Şahin (2017) used different text representation methods on 22729 Turkish documents of 7 different classes and compared the effect of these methods on the

classification success. BOW and Word2vec methods are used as word representation methods. The Word2vec vectors are learned through a dataset with 12 million sentences and 500 million words in Turkish. While training these vocabulary vectors; the vector size is 400 and the window is taken in a range of 5-10. In experiments using SVM classifier; the architectures of the Word2vec method are compared among themselves and when the skip-gram architecture is applied with negative sampling algorithm, 91% classification accuracy is achieved. In comparison to BOW, the classification success for Word2vec is 3% higher.

Çoban and Karabey (2017) applied the document classification on a data set containing 1250 lyrics of 5 different categories in their work. Word2vec, Doc2vec, and BOW are applied as text representation and *tf-idf* weighting method is used for BOW. Word2vec word vector averages are calculated for document representation. The vector size is taken in the range of 100-500 for Word2vec and Doc2vec. The classification accuracy for Word2vec shows an increase in parallel to the vector size; an opposite situation is observed for Doc2vec. In the experiments conducted with the SVM classifier, the BOW method is found more successful than the Word2vec and Doc2vec methods with 67.28% classification success.

Hayran and Sert (2017) have obtained word vectors using Word2vec on 16000 negative and 16000 positive Turkish tweets. Tweets are expressed using the average, sum, variance, and binary and triple combinations of these vectors. They compared the classification performance by using the SVM classifier and achieved the highest result (80.05%) when the average, total, and variance of the vectors are used together.

Ayata et al. (2017) have obtained word vectors with Word2vec over 5808 tweets of 4 classes; they represent tweets with the sum and product of these vectors. Experiments are conducted with SVM and RF classifiers to compare total and multiplication models. Among the classifiers applied to the texts represented by the product representation model, the highest classification success is achieved

with RF classifier with 67.13%. A higher classification accuracy on texts represented using the total representation model is achieved with the SVM classifier with 74.60%. In addition to the data used in Word2vec training, 158,885 tweets are added, and experiments are repeated in order to measure the effectiveness of the corpus size on the classification accuracy. Even though accuracy does not change much, the usage of large collections increases the success slightly.

Çelenli (2018) compared the BOW and Doc2vec methods in the study. He used news and tweet data. News texts are represented by applying BOW model with *tf* and *tf-idf* weighting and Doc2vec document vectors. In this study; the Doc2vec vectors are trained with a vector size of 100 over 20 million tweet data. Then, classification is applied with NBM, SVM, and NC classifiers. The success achieved with Doc2vec for these news texts is lower than BOW. In the repeated experiments for the tweets, a success increases of 3% is observed for the KNN classifier using Doc2vec.

Şen and Yanıkoğlu (2018) have created a large collection of news texts from 2 different newspapers in their work. This collection is represented in two ways; BOW model by using *tf-idf* weighting method and average of the vectors obtained with the Word2vec skip-gram model. SVM, LDA and NN classifiers are applied. The artificial neural network consists of 2 hidden layers with 50 nodes, ReLU as the activation function, RMSprop as the optimization method, and learning rate as 0.01 are selected. At the end of the experiments; while the LDA has the worst classification success; when word representation vectors are used with SVM, the success rate of 85% and above is obtained. The best classification success is 88% which is obtained by using 100-dimensional word vectors and artificial neural networks.

Amasyalı et al. (2018) compared the word, semantic and character representations for text classification. They created a data set with 3 classes and 17,289 tweets. BOW, n-gram, and Fasttext are used for text representation

methods. Both frequency and binary representation methods are used for BOW. SVM, RF, CNN, and LSTM are applied as classifiers. As a result; it has been observed that new generation approaches may be slightly more successful than traditional methods, and character-based representations for both have higher classification accuracy.

Bilgin and Şentürk (2017) performed sentiment analysis with paragraph vectors on English and Turkish tweets by using Doc2vec. DBOW and DM algorithms are applied to two datasets and then they are classified by using linear regression. As a result of the experiments; classification success for Turkish tweets are observed around 42% to 46%; while 61% to 66% classification accuracy is achieved for English tweets. DBOW algorithm is found to be more successful than the DM algorithm for both languages.

In Seyfioğlu and Demirezen (2017) studies, sentiment analysis and document classification on passenger comments of an airline company is performed by representing texts by using Word2vec, Doc2vec, and BOW. In the study, Xgboost classifier is applied. When BOW is used, 52.1% document classification accuracy and 75.2% accuracy for sentiment analysis are achieved. Their proposed method, in which the documents are represented by Word2vec vectors and the multiplication of the *tf-idf* values of the words, obtaining the Doc2vec document vectors, and solving the imbalance between the classes using the SMOTE method, yields a 71.16% document classification accuracy and 92.5% sentiment analysis success.

In the study of Çelenli et al. (2018), they compared the success of SVM, KNN, CC and CFSVM classifiers on multiple text data sets represented by Doc2vec, term frequency, term frequency and inverse document frequency. In the experimental results of work; it is observed that the paragraph vectors yield better results than SVM with *tf-idf* method in the case of small data size.

Ay Karakuş et al. (2018) have obtained word vectors using Word2vec skip-gram architecture on a dataset containing about 13M words. They used MLP,

CNN, LSTM, BiLSTM, and CNN-LSTM networks with 40,617 movie reviews to represent binary sentiment classification in Turkish texts. They tested their artificial neural networks using 2000 positive and 2000 negative movie reviews. As a result of the experiments; the highest success is achieved with 98.07% accuracy for the CNN-LSTM classifier, while the lowest success is observed with 78.27% accuracy from the MLP classifier.

Yıldırım and Yıldız (2018a) worked on datasets which contains 4900 documents of 7 classes, and 3600 documents of 6 classes. The texts are represented by using the BOW method and the vectors are trained by using the PV-DM and PV-DBOW architectures of the Doc2vec document representation method. Along with the BOW method, they have applied various feature selection methods such as information gain, chi-square, dice. They then applied the classification with traditional classifiers such as SVM, LR, ANN, and SGD. According to the results, it is concluded that the BOW representation method is more successful for Turkish texts than Doc2vec document representation method.

Another study of Yıldırım and Yıldız (2018b) used a dataset of 4900 documents of 7 classes. For the text representation; BOW, Word2vec, Doc2vec, and Glove methods are applied. Also, feature selection methods that are chi-square and information gain are applied to the BOW model. For the classification step, both traditional classifiers that are LR, SVM, KNN; and deep learning-based methods that are CNN, LSTM, GRU, and RNN are used. At the end of the study, it is concluded that traditional classifiers with bag-of-words text representations have better classification accuracy than word and document embedding methods used in the study for Turkish texts.

3. MATERIALS AND METHODS

In this section, the basics of the text representation methods, and the classifiers used are presented; after that, how these methods are applied in this thesis, and the datasets used in the experiments are explained.

3.1. Text Representation Methods

Text representation methods that are applied in this thesis can be categorized into two main titles: traditional text representation methods that are based on term weighting formulas; and word and document embedding methods that are based on neural network architecture.

3.1.1. Term Weighting Methods

In text mining and information retrieval, documents are expressed numerically in vector space. For the numeric representation of documents; first of all, terms occur in the collection are extracted then these terms are assigned numeric weights. In the literature, mostly *tf* and *tf-idf* (term frequency and inverse document frequency) term weighting methods are used. This document representation method is called as bag-of-words approach. In this thesis, BOW with these two weighting methods are used.

3.1.1.1. Tf (term frequency)

The term frequency is the number of times that a term is contained in a document. In this approach; only the frequency information is taken into account by ignoring the order of the terms.

The frequency of term k in the document d (i.e., $tf_{k,d}$) is used as the weight of the term k for document d (i.e., w_k) as shown in equation 3.1.

$$w_k = tf_{k,d} \quad (3.1)$$

3.1.1.2. Tf-idf (term frequency-inverse document frequency)

In this weighting method; the importance of each term in the collection of documents is represented. It is obtained by multiplying the term frequency with the inverse document frequency of the term.

The document frequency of a term (df) is the knowledge of how many documents in the document collection that contain the term. Inverse document frequency indicates the importance of a term in documents. It is obtained by scaling the document frequency with the number of documents. Inverse document frequency of the term k (idf_k) is obtained with equation 3.2.

$$idf_k = \log\left(\frac{\text{number of documents in collection}}{df_k}\right) \quad (3.2)$$

Tf-idf weight of the term k is computed as in equation 3.3;

$$tf - idf_k = tf_{k,d} \times idf_k \quad (3.3)$$

In the bag-of-words approach, a vocabulary consisting of words contained in all documents is obtained. This approach is called as the bag-of-words approach because each document is represented by a collection of words it contains. In this approach, document collection is represented by a document-term matrix where rows represent documents and columns represent each word in the vocabulary. For i^{th} row and j^{th} column of the document-term matrix; i^{th} row represents i^{th} document in the document collection and j^{th} column of the matrix represents tf or $tf-idf$ value of the j^{th} word in the vocabulary for the i^{th} document.

3.1.2. Embedding Methods

Another way to numerically represent documents for structures that cannot work directly with categorical variables such as artificial neural networks is to use vectors. When documents are represented by using bag-of-words, the order of words in the documents are lost, and vectors representing the documents are high

dimensional and sparse. Therefore, word embedding methods are recommended to prevent from these problems. The main objective of embedding methods is to represent words with low-dimensional vectors without losing semantic relations between words.

The embedding methods, which are very popular in recent years, and also, used in the thesis, are explained below.

3.1.2.1. Word2vec

Word2vec is a word embedding method offered by Mikolov et al. (2013a) and Mikolov et al. (2013b) in order to obtain high-quality vocabulary vectors through large-sized datasets consisting of millions of words. Basically, it is based on the principle that similar words are found close together, and words may have more than one degree of similarity. It is an artificial neural network structure consisting of three layers as input, projection, and output layer. It uses stochastic gradient descent and backpropagation during the training of the network. A word vector of the selected size is generated for each word represented by one-hot encoding in the input layer. The word vectors produced for close-meaningful words are located close to each other in the vector space. There are two log-linear architectures, CBOW, and skip-gram for word representation.

In CBOW architecture (in Figure 3.1), word ($w(t)$) prediction is made from the given context ($w(t-2)$, $w(t-1)$, $w(t+1)$, $w(t+2)$) in other words from the surrounding words.

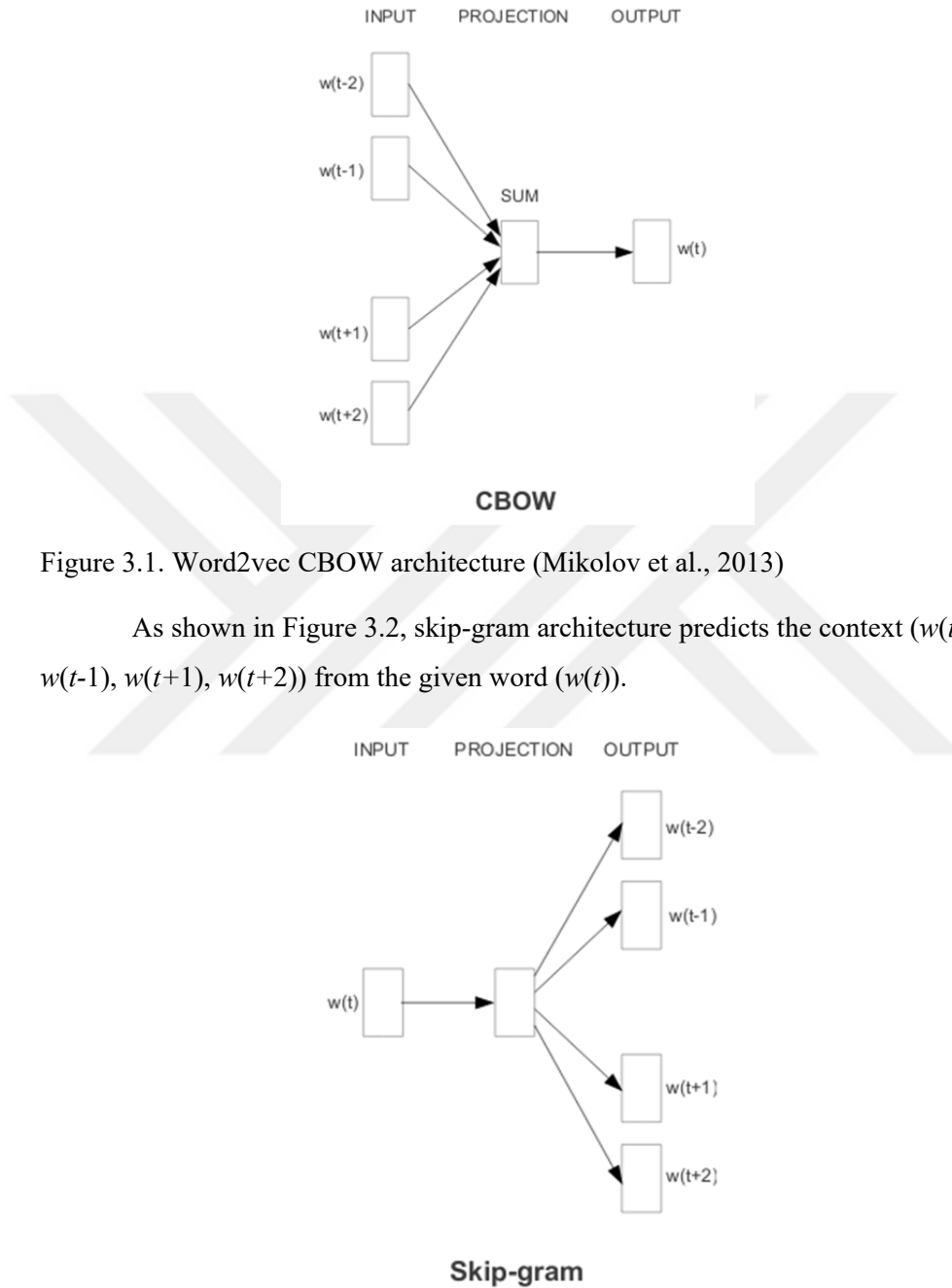


Figure 3.1. Word2vec CBOW architecture (Mikolov et al., 2013)

As shown in Figure 3.2, skip-gram architecture predicts the context ($w(t-2)$, $w(t-1)$, $w(t+1)$, $w(t+2)$) from the given word ($w(t)$).

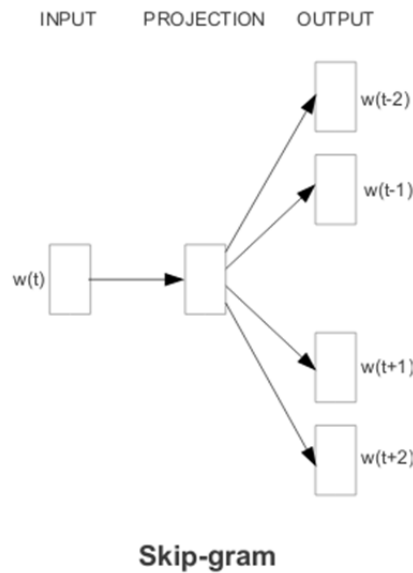


Figure 3.2. Word2vec skip-gram architecture (Mikolov et al., 2013)

In Word2vec word embedding method when word vectors are obtained; during each iteration, the output vectors are updated as well as the input vectors. Hierarchical softmax and negative sampling methods are used to make this effective.

Hierarchical softmax uses Huffman binary trees instead of one-hot vector representation in the output layer. It is an effective approach because of using the softmax function in the output layer.

In negative sampling, instead of updating all output vectors in each iteration, a certain number of negative samples are taken to speed up training.

3.1.2.2. Doc2vec

Doc2vec is the document representation model proposed by Le and Mikolov (2014) as “paragraph vector” to obtain fixed-length document vectors from variable-length texts. In this method, each document is expressed by a fixed-length vector that is inclined to predict the words in the document. In addition to the word vector; each paragraph has a unique vector. During the training; estimating of the next word or words are made by taking the sum or averages of word vectors and paragraph vectors. Doc2vec model uses stochastic gradient descent and backpropagation for training as well as the Word2vec model. It has two architecture: PV-DM (Paragraph Vector – Distributed Memory Model) and PV-DBOW (Paragraph Vector – Distributed Bag-Of-Words Model).

As can be seen in Figure 3.3, paragraph vector-distributed memory model tries to predict the middle word using randomly sampled context words and paragraph vectors.

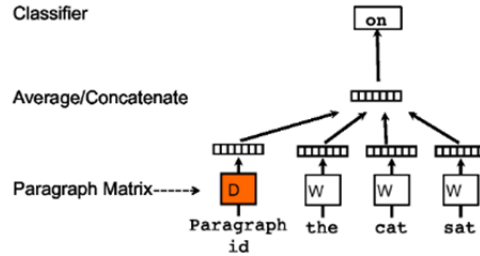


Figure 3.3. Doc2vec PV-DM architecture (Le and Mikolov, 2014)

In paragraph vector-distributed bag-of-words model (in Figure 3.4); paragraph vector is trained to predict a word in a small window.

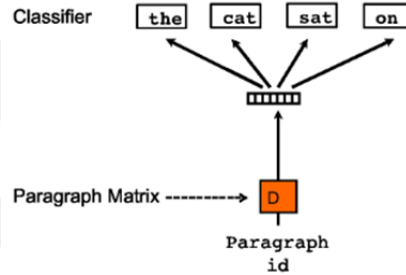


Figure 3.4. Doc2vec PV-DBOW architecture (Le and Mikolov, 2014)

3.1.2.3. Fasttext

In previous word embedding models; each word is represented by a distinct vector. Sub-word information such as the structure within the words, syllables and the coexistence of letters are ignored. Bojanowski et al. (2017) developed Fasttext that uses the character n-grams while obtaining vector representations. This method is based on the skip-gram model. Each word is represented by the sum of the character n-gram vectors. Training Fasttext is faster than other methods. Since the vectors have been learned for character n-grams; vectors for words not found in the corpus used for training can also be obtained. In addition; the use of character n-grams allows us to obtain better vector representations for morphologically rich languages such as Turkish.

3.1.2.4. Glove

Created by the combination of two important model families in the literature which are global matrix factorization and local context window methods, Glove has been proposed by Pennington et al. (2014). It is set out to obtain a model that incorporates both semantic structures and not limited to local context but also global count statistics. For this reason, Glove combines global matrix factorization methods such as Latent semantic analysis containing statistical information and local context window methods such as Word2vec, which are successful in analogy tasks. Based on a similar basis with Word2vec; which is co-occurrence ratio between two words in a context has a strong connection with meaning. Unlike Word2vec, it works through word-word co-occurrence matrix instead of processing words.

3.2. Text Classification Methods

In this thesis, classifiers that are used for text classification are categorized under the two main headings that are traditional classifiers and deep learning-based methods.

3.2.1. Traditional Classification Methods

These methods consist of basic supervised machine learning techniques which do not have any neural network architecture. In this section, classifiers that are the most successful ones for text classification are included.

3.2.1.1. Naive Bayes Classifier

Naive Bayes classifier, which is a very popular method for text classification, is a supervised learning algorithm based on Bayes' theorem. It assumes that all attributes are independent and equal. The purpose of Bayes classification is to estimate the value of class c when the X dependent variable is

known. X is the evidence and an n -dimensional vector $(x_1, x_2, x_3 \dots x_n)$ is the attribute vector.

Bayes Theorem which is given in equation 3.4 is used to estimate the probability of class c given an attribute vector.

$$P(c|x_1, x_2, x_3 \dots x_n) = P(x_1, x_2, x_3 \dots x_n|c)P(c)/P(X) \quad (3.4)$$

where, $P(c|x_1, x_2, x_3 \dots x_n)$ is posterior probability, $P(x_1, x_2, x_3 \dots x_n|c)$ is likelihood, $P(c)$ is probability of class, and $P(X)$ is probability of predictor.

Since $P(X)$ is constant, $P(x_1, x_2, x_3 \dots x_n |c)$ should be maximum. This is represented with following equation 3.5;

$$\operatorname{argmax}\{P(x_1, x_2, x_3 \dots x_n |c) * P(c)\} \quad (3.5)$$

Because Naive Bayes classifier is a fast and highly scalable classifier, it can be seen that many studies in the literature, such as spam filtering, document classification, and sentiment analysis have applied the Naïve Bayes classifier. The advantage of the Naïve Bayes classifier is that training on small data sets is very easy. But; the disadvantage is variables in real-life problems are mostly related; and therefore, the Naive Bayes classifier, which assumes that the attributes are independent of each other, cannot establish a connection between the attributes.

For the text classification problem, Naive Bayes Multinomial which is a Naive Bayes approach and assumes each of its features with multinomial distributions has achieved higher success according to McCallum and Nigam (1998). NBM is also frequently used in sentiment analysis and document classification problems with Turkish texts. For this reason, Naive Bayes Multinomial is included in this study.

3.2.1.2. Support Vector Machines

SVM is a supervised learning method that is mostly used in classification problems. It tries to find a hyperplane that separates the input data represented in n -dimensional space. SVM takes hyper-plane, which maximizes the distance between the nearest data points (called support vectors) of different classes. Support vectors and hyper-plane are shown in Figure 3.5.

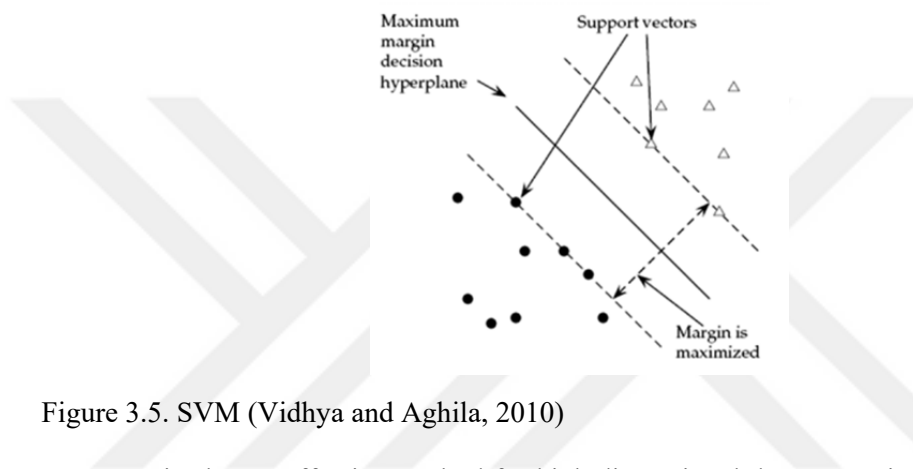


Figure 3.5. SVM (Vidhya and Aghila, 2010)

It is also an effective method for high dimensional data. SVM is a method commonly used especially in document classification and sentiment analysis.

LibSVM (Chang and Lin, 2011) is an SVM library that was created in the year 2000 and continues to be developed. LibSVM supports support vector classification, support vector regression, and one class SVM. The calculation time for the shrinking and caching problems of SVM is reduced in LibSVM. Special adjustments are made for unbalanced datasets. While SVM only estimates the class label, LibSVM makes the probability estimation for class prediction.

3.2.1.3. Random Forest

Random forest is a supervised machine learning method proposed by Breiman (2001). It is a preferred method because it can be used in both classification and regression problems. In its simplest definition; RF creates

multiple decision trees and combines the decision trees for more accurate estimation. These decision trees are subsets randomly selected from the dataset. This method is determined by a random selection when splitting a node. According to Cutler et al. (2011), some advantages of RF are; training and estimation time is short, can be used for multi-class classification problems and regression, the number of parameters for training algorithm is low, and can be used in large-scale problems.

3.2.1.4. Logistic Regression

Logistic regression is a special version of linear regression used when the data is categorical (Jurafsky and Martin, 2018). Unlike linear regression, logistic regression uses a sigmoid function. Logistic regression can be applied in binary and multiclass problems. In LR, the dependent variable follows the Bernoulli distribution and estimates are made using maximum likelihood. A few reasons why LR classifier is preferred are; training is fast, easy to implement, does not require data scaling, and calculates the probability of class estimation for instances in the dataset.

3.2.2. Deep Learning-based Classification Methods

In traditional machine learning methods; hand-designed features are obtained from sentences. Documents are represented by using these features and classification is made by classification algorithms such as SVM, NB, classification trees, etc. In these methods, feature extraction depends mostly on the problem. For each problem the properties must be reproduced; this increases time and resource utilization.

With the ability to represent words and documents in smaller dimensional space, artificial neural network-based methods have gained importance in order to obtain a large number of attributes from these representations. In artificial neural network-based methods, the input sequence is passed through a multi-layer network structure and many important features are extracted. Input sequence; can be the lookup tables learned during the training of the network as described in

Collobert and Weston (2008), as well as word embedding methods that have reached state-of-art results in many NLP problems. In this thesis; CNN, LSTM, and artificial neural network architecture which is the combination of these two architectures are used.

3.2.2.1. Convolutional Neural Networks

A convolutional neural network is a feed-forward artificial neural network that was proposed by LeCun et al. (1989) for computer vision. In the following years, it has been used for different natural language processing tasks; such as sentence modeling (Kalchbrenner et al., 2014), part of speech, chunking, named entity recognition, semantic role labeling (Collobert et al., 2011), sentence classification (Kim, 2014).

Convolutional neural network architecture simply consists of three layers: an input layer, hidden layer, and output layer. But in the hidden layer; unlike traditional neural network architectures, there are consecutive convolution and pooling layers. Figure 3.6 shows a simple CNN architecture for sentence classification.

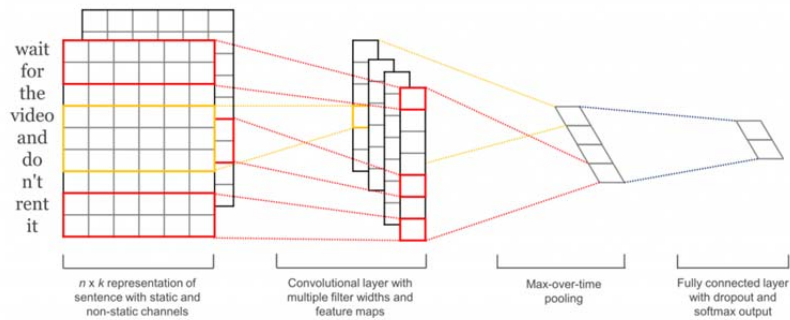


Figure 3.6. Basic CNN structure for sentence classification (Kim, 2014)

3.2.2.1.(1). Convolution Layer

Convolution layer transforms the input data using local neurons from the previous layer (i.e., extracts features from input data). To do this; it creates feature

maps by applying kernels of different sizes and weights on the input sequence. A completed convolutional layer consists of multiple feature maps with different weight vectors so that multiple features can be obtained from each location (LeCun et al., 1989). Convolution layer takes its name from the convolution operation, which is a mathematical method applied in this layer. Convolution operation aims to improve machine learning methods in three ways: sparse connection, parameter sharing, and equivariant representations. (Goodfellow et al., 2016)

Sparse Connection: Traditional artificial neural networks have connections between all input and output neurons. As shown in Figure 3.7, in CNN architecture since the applied core size is smaller than the input size, there is no connection between each input and each output neuron. At the top of Figure 3.7, a network structure applied convolution operation with a kernel width of 3 can be seen, and at the bottom of the figure, there is a traditional network structure that is fully connected.

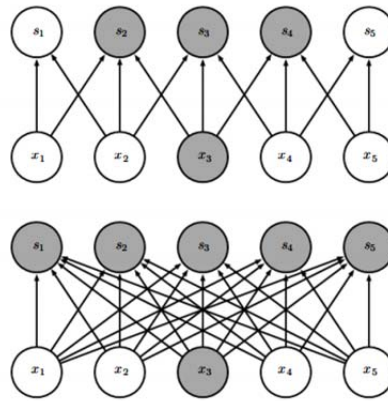


Figure 3.7. Sparse connection (top) and dense connection (bottom) (Goodfellow et al., 2016)

Elimination of the necessity of the connection between all neurons; reduces the need for memory, requires less calculation, and simplifies statistical calculations.

Parameter Sharing: In convolution operation, the kernel moves in all positions on the input sequence by sliding. In this way; different parameter learning, and calculation requirements are eliminated for each location. Although this does not affect the running time; it provides an important advantage in terms of memory gain due to the ease of statistical calculation and reduction of matrix multiplications.

Equivariant Representations: Due to parameter sharing; changes in the input are also reflected in the output in the same size. Considering the object in an image as an example; a few units shift of the object's input means that the object will shift at the same rate in the output.

3.2.2.1.(2). Activation Functions

After convolution operation, the results are passed through an activation function. Activation functions; are mostly nonlinear functions and provide the use of artificial neural networks in non-linear real-world problems. The most commonly used activation functions are RELU, sigmoid, hyperbolic tangent, and Softmax.

Rectified linear Units (ReLU) function is computed as shown in equation 3.6 where x is input, in Figure 3.8.

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \quad (3.6)$$

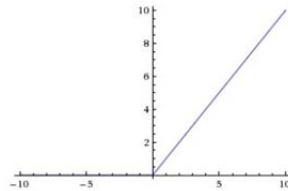


Figure 3.8. ReLU activation function (Source: <http://cs231n.github.io/neural-networks-1>)

Sigmoid function is computed as in equation 3.7 where x is input in Figure 3.9.

$$f(x) = 1 / (1 + e^{-x}) \quad (3.7)$$

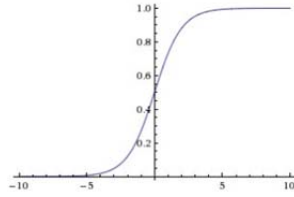


Figure 3.9. Sigmoid activation function (Source: <http://cs231n.github.io/neural-networks-1>)

Hyperbolic Tangent function is computed as in equation 3.8 where x is input in Figure 3.10.

$$f(x) = \tanh(x) = (2 / (1 + e^{-2x})) - 1 \quad (3.8)$$

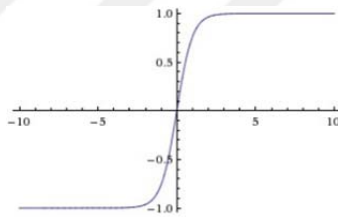


Figure 3.10. Tanh activation function (Source: <http://cs231n.github.io/neural-networks-1>)

Finally, softmax function is computed as in equation 3.9 where x is input in Figure 3.11.

$$f(x) = e^x / \sum_j e_j^x \quad (3.9)$$

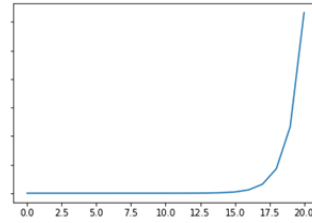


Figure 3. 11. Softmax activation function

3.2.2.1.(3). Stride and Padding Operations

Stride refers to how many units the filters move over the input data. Figure 3.12 shows the application of a 3x3 filter to an input data size of 7x7. Stride=1 is given at the top, and stride=2 is presented at the bottom of the figure. Filter moves 1 unit in x and y -directions when stride=1 and moves 2 units in x and y -directions when stride=2.

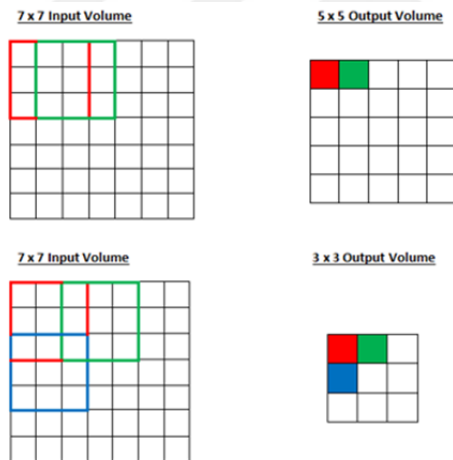


Figure 3.12. Stride operation (Source: <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2>)

Padding prevents the size of the input data from shrinking and thus increases the number of properties passed to the next layers. The output size after the stride and padding operations is expressed by the equation 3.10.

$$output = \frac{input - filter + 2 \times padding}{stride} + 1 \quad (3.10)$$

where *output* is size of output feature map, *input* is size of input, *filter* is size of filter, *padding* is size of padding value, and *stride* is size of stride value.

3.2.2.1.(4). Pooling Layer

In the pooling layer, subsampling is performed on the feature vector from the convolutional layer according to the selected method by applying different approaches such as max-pooling, min-pooling, and average-pooling. The purpose at the pooling layer is to obtain a fixed size output from different length vectors (i.e., allows the use of different lengths of input), and to reduce the size of the features without losing important information in the feature vectors.

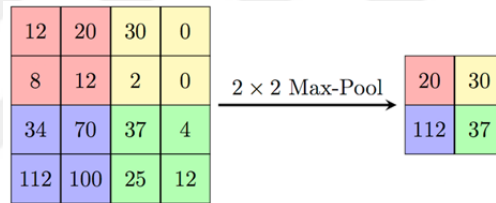


Figure 3.13. Max-pooling operation (Source: <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2>)

There are different subsampling methods such as maximum pooling, minimum pooling, and average pooling which use different information. As shown in Figure 3.13; 2x2 max pooling is performed by taking the maximum value in each 2x2 frames.

3.2.2.1.(5). Output Layer

The output layer is also the fully connected layers of the multi-layer perceptron. The input of this layer is the output from the pooling layer. Class probability is made by weighting these inputs through various activation functions.

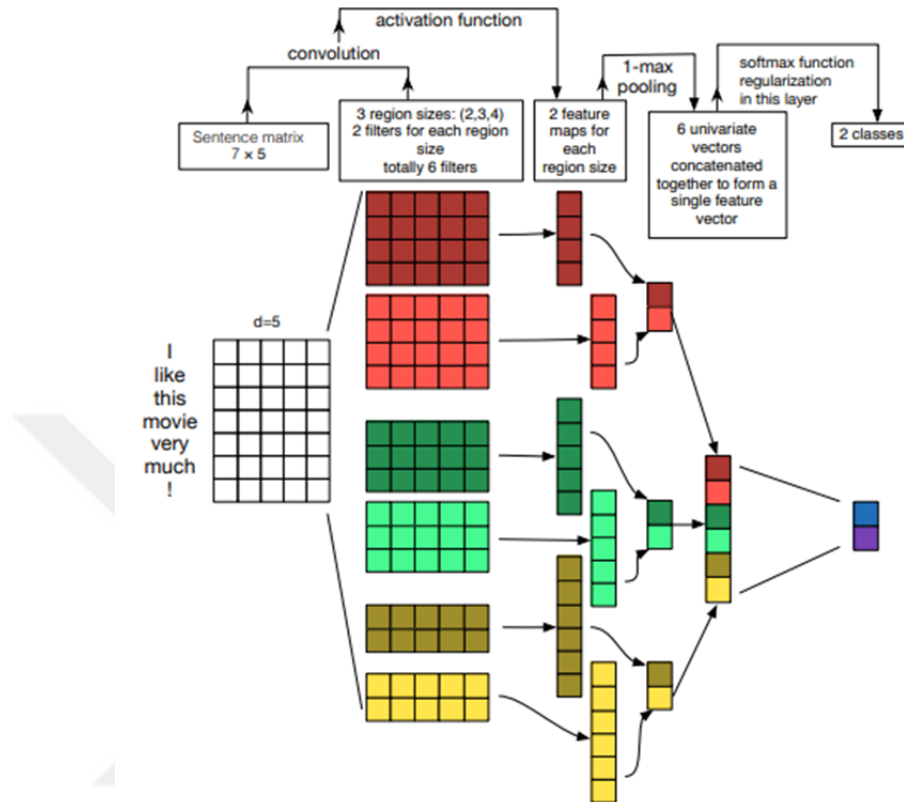


Figure 3.14. An example of CNN for sentence classification (Zhang and Wallace, 2017)

The use of CNN in text classification can be explained through Figure 3.14 (Zhang and Wallace, 2017) as follows: in the input layer, word vectors are obtained by word embedding methods on the dataset. The input size is determined based on how many words the longest sentence in the documents contains. In Figure 3.14; a sentence of length 7 is represented by using 5-dimensional word embedding vectors. Therefore, the input matrix or sentence matrix size equal to 7x5. In the convolution layer; when working with texts, the kernels are moved in the y-direction because each line represents a word. In this example, a total of 6 filters are applied with dimensions 2, 3 and 4. At the end of the convolution, 6 feature maps are obtained. In the pooling layer; 1-max pooling is applied to each 6 feature

maps, then the largest values in each feature maps are obtained. In the flattening layer, the values obtained from the pooling layer are merged. Finally, in the output layer; class predictions are obtained by applying the Softmax activation function.

3.2.2.2. Recurrent Neural Networks

Recurrent neural network (Elman, 1990) is an artificial neural network architecture that stores history information and uses this information to predict the future and is developed to process sequential data. This architecture is represented by successive blocks as can be seen in Figure 3.15. The same calculations are made for all elements in the input data and these calculation results are the input of the next block. Unlike CNN, it has a memory concept and can learn long-term dependencies. It is more flexible than CNN to work with sentences and documents of different sizes. RNN is used in different NLP problems such as language modeling, machine translation, speech recognition.

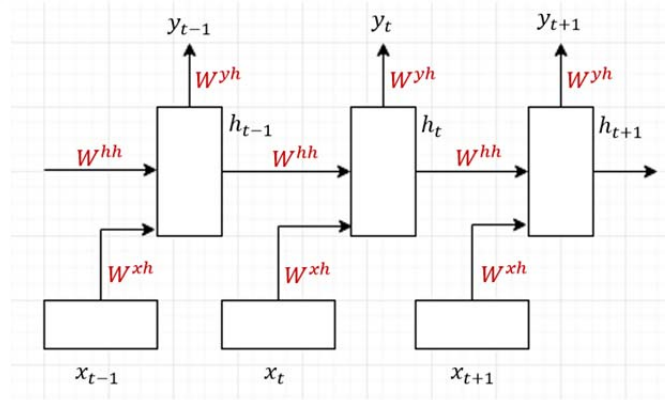


Figure 3.15. RNN blocks

In Figure 3.15; h_t is a hidden state at time step t , x_t represents the input vector at time step t , and h_t is calculated by equation 3.11.

$$h_t = W^{hh}h_{t-1} + W^{xh}x_t \quad (3.11)$$

where W^{hh} is weight matrix used to condition previous state h_{t-1} , W^{xh} is weight matrix used to condition input x_t , and y_t is the output of probability distribution over vocabulary at time t and calculated by equation 3.12.

$$y_t = \text{softmax}(W^{yh}h_t) \quad (3.12)$$

where W^{yh} is weight matrix at time t .

In the RNN architecture, the hidden state is the memory of the neural network. As equation 3.12 suggests, the prediction is made according to this hidden state. Also, in this architecture, W weight values are common as seen in Figure 3.15. This situation reduces the number of parameters and calculation time for each block.

However, although RNN can calculate using its past knowledge, in practice it can only take a few steps back. Also; only a small portion can be transmitted during the backpropagation of the error calculated in the final layers. This situation is expressed as vanishing gradient problem. This RNN structure cannot overcome the vanishing gradient problem. Different RNN architectures such as bidirectional RNN, deep bidirectional RNN and LSTM have been developed for RNN insufficiency.

3.2.2.3. Long Short-Term Memory Networks

LSTM is an RNN architecture developed by Hochreiter and Schmidhuber (1997) to overcome the problem of long-term dependencies. Although it has a more mixed architecture than RNN, it is expressed as repetitive blocks because it is based on RNN. RNN architecture has only hidden state, while LSTM has two states, hidden and cell state. In the LSTM architecture, there are three gates that allow the transfer the information to the cell state. They are cell state and forget gate structures that provide solutions to the vanishing gradient problem in LSTM. Each LSTM block consists of 2 states and 4 layers. An LSTM block (in the blue

rectangle) and layers (in the red rectangles) it contains can be better understood from Figure 3.16.

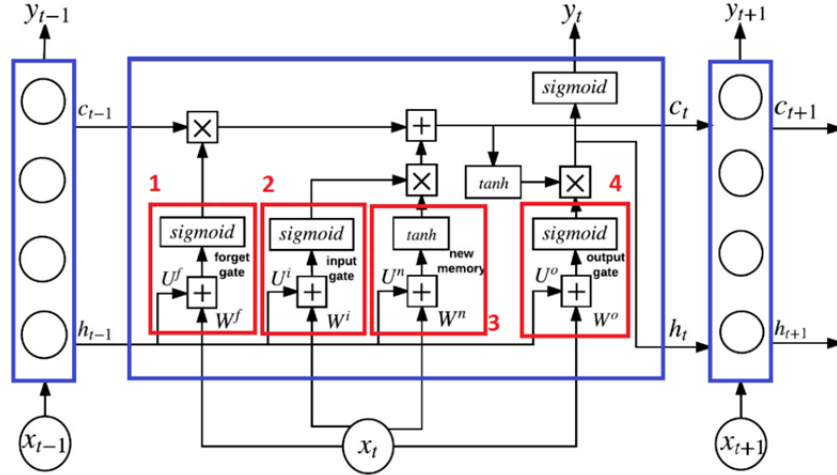


Figure 3.16. LSTM architecture (Zhang et al., 2018)

In the first step of LSTM (red rectangle with 1 in Figure 3.16), it finds out what information to delete from cell state with the forget gate with equation 3.13.

$$f(t) = \sigma(W^f x_t + U^f h_{t-1}) \quad (3.13)$$

For equation 3.13, $f(t) = 0$ means delete this information, and $f(t) = 1$ means keep the information.

In the next step (the red rectangle with 2 in Figure 3.16), it finds out what information is stored in the cell state. For this, the input gate layer; LSTM will decide what information to update using equation 3.14.

$$i_t = \sigma(W^i x_t + U^i h_{t-1}) \quad (3.14)$$

Then; tanh function will generate new candidate values (\tilde{c}) to be added to the layer cell state using equation 3.15.

$$\tilde{c}_t = \tanh(W^n x_t + U^n h_{t-1}) \quad (3.15)$$

In the next step (red rectangle with 3 in Figure 3.16), the old cell state (c_{t-1}) needs to be updated to the new cell state (c_t) using the candidate vectors created in the previous step (\tilde{c}_t) using equation 3.16.

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad (3.16)$$

Finally, in the last step, the output is calculated according to c_t by equation 3.17. For this, the output gate layer will decide the output first. The tangent function will be applied (equation 3.18) to the cell state and multiplied by o_t to decide which parts of the LSTM will be output.

$$o_t = \sigma(W^o x_t + U^o h_{t-1}) \quad (3.17)$$

$$h_t = o_t * \tanh(c_t) \quad (3.18)$$

3.3. Text Classification

Text classification process can be summarized in a few basic steps. These steps include; obtaining the data, preparing and preprocessing the data, converting the text into numeric vectors, and finally applying the classifier. In this thesis; a similar methodology is followed for the document classification and sentiment classification. The steps applied for sentiment analysis and document classification are given in Figure 3.17 and Figure 3.18, respectively.

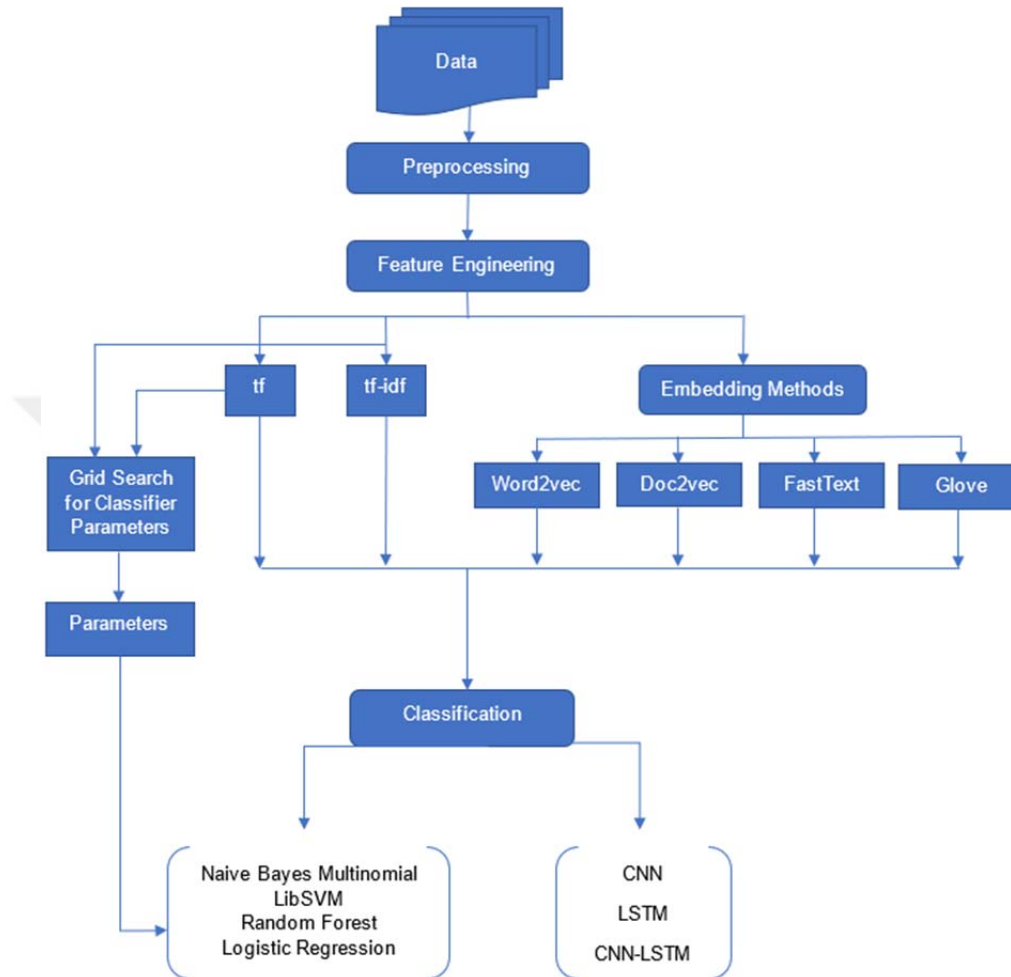


Figure 3.17. Methodology of sentiment classification task

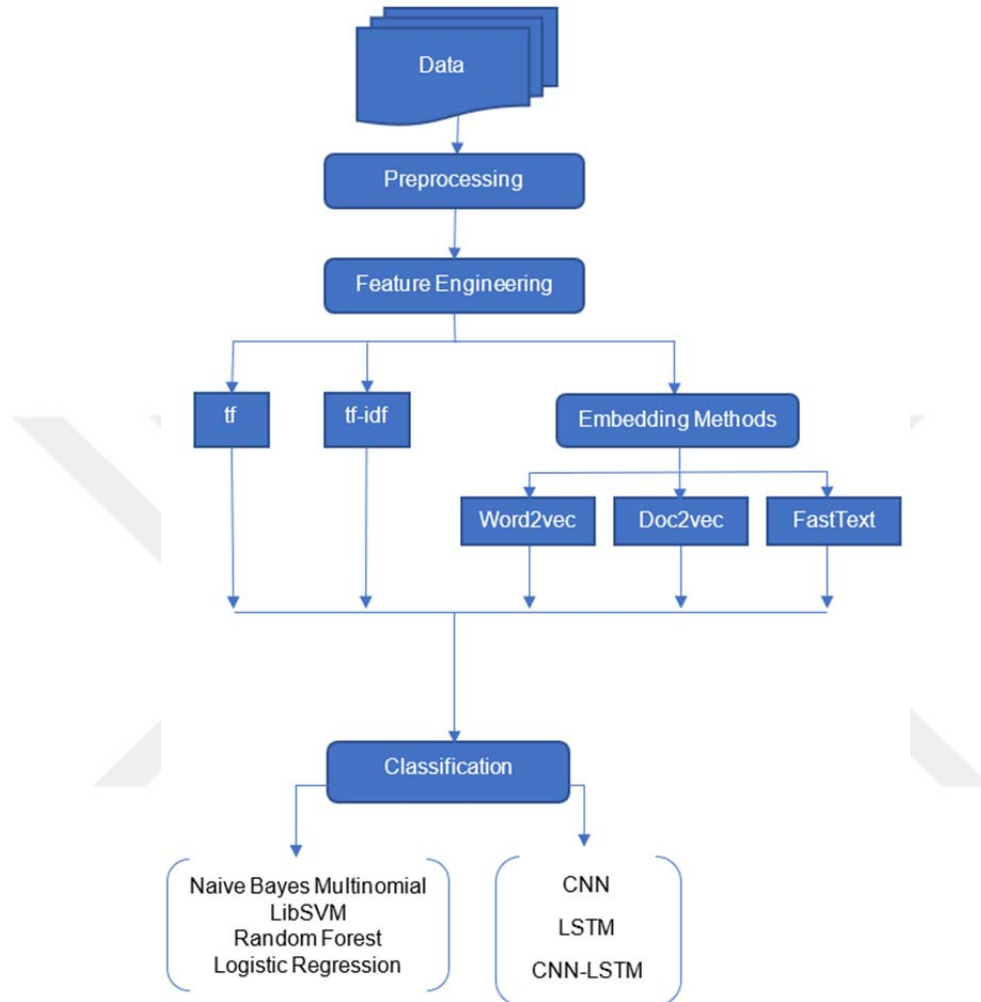


Figure 3.18. Methodology of document classification task

3.4. Datasets

In this thesis; different datasets are used for sentiment analysis and document classification tasks. The details of the data sets used for the two tasks are explained below.

For sentiment classification task; the Turkish Sentiment Dataset, shared by Hayran and Sert (2017) and the 20M tweet datasets prepared by the Yıldız

Technical University natural language processing group which is called as Kemik Natural Language Processing Group are used.

Turkish Sentiment Dataset is a dataset that contains 16000 positive and 16000 negative Turkish twitter messages and used for sentiment analysis in the studies of Hayran and Sert (2017). After the preprocessing steps applied to the data, a total of 31,250 samples are obtained. Each sample in the dataset contains 9 words on the average. This dataset is used for the classification of sentiment in our experiments. For these experiments; 70% of randomly selected data is used as training data and the remaining 30% is used as test data. In the following table, Table 3.1 shows the number of instances for each class train and test sets for the Turkish Sentiment Dataset.

Table 3.1. Class distribution of Turkish Sentiment Dataset

| class | Number of samples in test data | Number of samples in train data | total |
|----------|-----------------------------------|------------------------------------|-------|
| positive | 4586 | 10783 | 15369 |
| negative | 4789 | 11092 | 15881 |
| total | 9375 | 21875 | 31250 |

20M tweet dataset contains approximately 20 million tweets and shared by Kemik Natural Language Processing Group. There are only tweet examples in this dataset and no class information is available. After our pre-processing steps; a total of 17,125,128 samples are obtained from this dataset. This dataset is used to train embedding vectors in sentiment classification experiments. The purpose of using this dataset in our study is to measure the effect of the size of the corpus on which the embedding vectors are trained. This dataset is the largest dataset from the same domain as Turkish Sentiment Dataset.

For document classification task; the SuDer dataset shared by Şen and Yanıkoğlu (2018) is used. SuDer is collected from two known newspapers in Turkey which are Sabah and Cumhuriyet. In this thesis; the documents taken from

these two newspapers are used separately and together. For this reason, the statistics of these two separate datasets, which are generated by the news texts from the Cumhuriyet and Sabah newspapers is given. The number of documents obtained after some pre-processing steps and the elimination of blank documents is different from the shared original data. Class distribution of SuDer after preprocessing step is shown in Table 3.2.

After the preprocessing steps, a dataset of 220,602 samples for the Cumhuriyet dataset and 419,983 samples for the Sabah dataset are obtained. 640,585 samples of 13 classes, consisting of the sum of these two datasets are used to train embedding vectors for the document classification task.

Table 3.2. Class distribution of SuDer dataset

| class | Number of examples per class |
|--------------|------------------------------|
| türkiye | 84664 |
| yazarlar | 101923 |
| spor | 31355 |
| dünya | 20836 |
| siyaset | 15960 |
| ekonomi | 93639 |
| teknoloji | 7899 |
| kültür-sanat | 6486 |
| yaşam | 127464 |
| sağlık | 2561 |
| eğitim | 2347 |
| çevre | 1688 |
| gündem | 143763 |
| total | 640585 |

In our document classification experiments; a total of 97,298 documents in 9 classes from Cumhuriyet dataset, and a total of 419,983 documents belonging to 4 classes from Sabah dataset are used. For both datasets; based on the class

distribution of the data, randomly selected 75% of the samples are used as training set and the remaining 25% are used as the test set. For the two datasets, class distributions and number of documents in train and test sets can be seen in Table 3.3 and Table 3.4, respectively.

Table 3.3. Class distribution of Cumhuriyet dataset

| class | Number of samples in train data | Number of samples in test data | total |
|--------------|------------------------------------|-----------------------------------|-------|
| çevre | 1266 | 422 | 1688 |
| eğitim | 1760 | 587 | 2347 |
| sağlık | 1921 | 640 | 2561 |
| kültür-sanat | 4865 | 1621 | 6486 |
| teknoloji | 5924 | 1975 | 7899 |
| ekonomi | 6124 | 2042 | 8166 |
| siyaset | 11970 | 3990 | 15960 |
| dünya | 15627 | 5209 | 20836 |
| spor | 23516 | 7839 | 31355 |
| total | 72973 | 24325 | 97298 |

Table 3.4. Class distribution of Sabah dataset

| class | Number of samples in train data | Number of samples in test data | total |
|----------|------------------------------------|-----------------------------------|--------|
| yazarlar | 51075 | 17025 | 68100 |
| ekonomi | 64105 | 21368 | 85473 |
| yaşam | 91985 | 30662 | 122647 |
| gündem | 107822 | 35941 | 143763 |
| total | 314987 | 104996 | 419983 |

3.5. Preprocessing

Texts from online platforms, especially social media platforms such as Twitter, often have spelling errors or undefined characters due to character

constraints or writing speed. In text processing, the words with misspelling increase the size of the vocabulary which causes to increase in computation cost, therefore, runtime for the methods to be applied.

For this reason, some language-specific pre-processing steps are applied to the texts before working with the text in NLP. In this thesis, pre-processing steps that are mostly preferred in Turkish studies are applied. The preprocessing steps applied for the data used in this thesis can be listed as follows: For the Turkish Sentiment Dataset and 20M tweets dataset;

- All characters except the letters are eliminated.
- Since the number of consecutive repeating letters in Turkish is limited to 2, the number of consecutive repeating letters is restricted to 2.
- Small case conversion is applied.

In SuDer dataset;

- Characters other than letters and numbers are eliminated as the texts consist of news documents, misspelling errors are very infrequent, therefore corrections are not applied for the words.

- The suffixes that are added to special names in Turkish are separated using an apostrophe. In text separated by an apostrophe, the part before the apostrophe is taken. Because words are got as attributes in our study.

- Small case conversion is applied.

3.6. Feature Engineering

In this step, the documents in our dataset are converted to numerical vectors and the documents for the classifiers to be applied in the next steps are prepared. For computing document vectors, conventional term frequency and inverse document frequency methods are used. In addition, embedding methods are applied as follows.

3.6.1. Applying Tf and Tf-idf

All the words contained in all the documents in the dataset are extracted and these words form the vocabulary. Each document is expressed by the terms that it contains. For each term in each document, *tf* and *tf-idf* values are computed and the whole dataset is represented by a term-document matrix such that the number of columns is equal to the number of unique words in the vocabulary, and the number of rows is equal to the number documents in the corpus. Value in the i^{th} row and j^{th} column of the term-document matrix is *tf* or *tf-idf* value of term j for the document i .

3.6.2. Training Embedding Methods

The embedding methods are applied for document vector computations. Embedding methods have become popular in recent years and started to be used when working with Turkish texts. One of our aims is to measure the success of embedding methods against the traditional *tf* and *tf-idf* representation methods. In this step, Word2vec, Doc2vec, Fasttext, and Glove embedding methods are used respectively. When implementing Word2vec, Doc2vec and Fasttext embedding methods, the Genism¹ library with version 3.7.2, a very popular and free Python library that was cited 1539 times from 2010 to May 2019 is used. For Glove word vector representations, the codes² created by StanfordNLP³ and shared on GitHub are used.

Within the scope of the thesis; for sentiment classification task the Word2vec, Doc2vec, Fasttext, and Glove vectors are trained; however, for the document classification task, only Word2vec, Doc2vec, and Fasttext vectors are trained. Because after the sentiment analysis experiments, it is observed that Glove

¹ <https://radimrehurek.com/gensim/>

² <https://github.com/stanfordnlp/GloVe>

³ <https://nlp.stanford.edu/projects/glove/>

vectors have lower success than other methods. Therefore, the other 3 embedding methods are used for document classification experiments.

In this section; the parameters in the libraries used to train embedding vectors and the default values of these parameters are explained. Then, the selected parameter values according to the studies conducted for Turkish texts in the literature (Şen and Erdoğan, 2014; Şahin, 2017; Hayran and Sert, 2017) and the documentation of the libraries used are explained.

- Word2vec default parameters defined in the Gensim library are;

```
gensim.models.Word2vec.Word2Vec(sentences=None, size=100,
window=5, min_count=5, workers=3, sg=0, hs=0, negative=5,
iter=5)
```

Sentences refer to all the documents used to train the vectors. To prepare our data as input to this parameter; dataset must be a list of documents and a document should be a list of words. The following Figure 3.19 shows the format of the input data form for Word2vec.

```
[[['güreşle', 'aktif', 'spor', 'hayatına', 'veda', 'etti'], ['kamil', 'ilk', 'mii', 'bugün', 'dua', 'etsem', 'kabul', 'oluyor', 'şükürler', 'olsun'], ['takılma', 'sen'], ['yok', 'çıkarmam', 'yeni', 'aldım', 'çalarlar']]]
```

Figure 3.19. An example of Gensim Word2vec input format

Size represents the vector size or number of neurons in the hidden layer during the training. In our experiments, our aim is to measure the effect of the vector dimensions on the classification performance by taking the parameter 100 and 300 for the sentiment classification task and 100, 200, and 300 for the document classification task. The window is the number of words to be looked at before and after the current word during the training. In our experiments; the default value of 5 for the window size is used. Min_count is the threshold such that all words with a frequency less than the min_count are eliminated from the corpus. In all our experiments, the min_count value is set as 1 to use all words. Workers represent the number of threads to be used to speed up training on

multi-core machines. *Sg* refers to the training algorithm used such that for skip-gram $sg=1$, for cbow $sg=0$. During training $hs=1$ is used for hierarchical softmax, and $hs=0$ and $neg>0$ are set for negative sampling. *Negative* indicates how many noise words to select. It is selected between 5-20 according to the documentation⁴. In terms of taking an average, the value is set to 10 in our experiments. *Iter* parameter specifies how many iterations will run on the corpus. In all experiments, the number of *iter* is taken as 10 in all word vectors.

- Doc2vec default parameters defined in the Gensim library are;
`gensim.models.Doc2vec.Doc2vec(documents=None, dm=1, size=100, window=5, min_count=5, workers=3, hs=0, negative=5, epochs=5)`

Documents: Each document in the corpus must be in TaggedDocument format. In this format; each document consists of a word list and a tag value. The following Figure 3.20 shows an example for the code and corpus that in the TaggedDocument format.

```
# create d2v input -- TaggedDocument
documents = [TaggedDocument(doc, labels[i]) for i, doc in enumerate(documents)]

print(documents[15:19])

[TaggedDocument(words=['güreşle', 'aktif', 'spor', 'hayatına', 'veda', 'etti'], tags=1), TaggedDocument(words=['kam il', 'ilk', 'mii', 'bugün', 'dua', 'etsem', 'kabal', 'oluyor', 'şükürler', 'olsun'], tags=1), TaggedDocument(words=['takılma', 'sen'], tags=1), TaggedDocument(words=['yok', 'çıkarmam', 'yeni', 'aldım', 'çalarlar'], tags=1)]
```

Figure 3.20. An example for Gensim Doc2vec input format

Other parameters are used for the same purposes as the parameters explained for Word2vec. The only difference in parameters is; the *iter* parameter in Word2vec is the *epoch* parameter in Doc2vec.

- Fasttext default parameters defined in the Gensim library are;

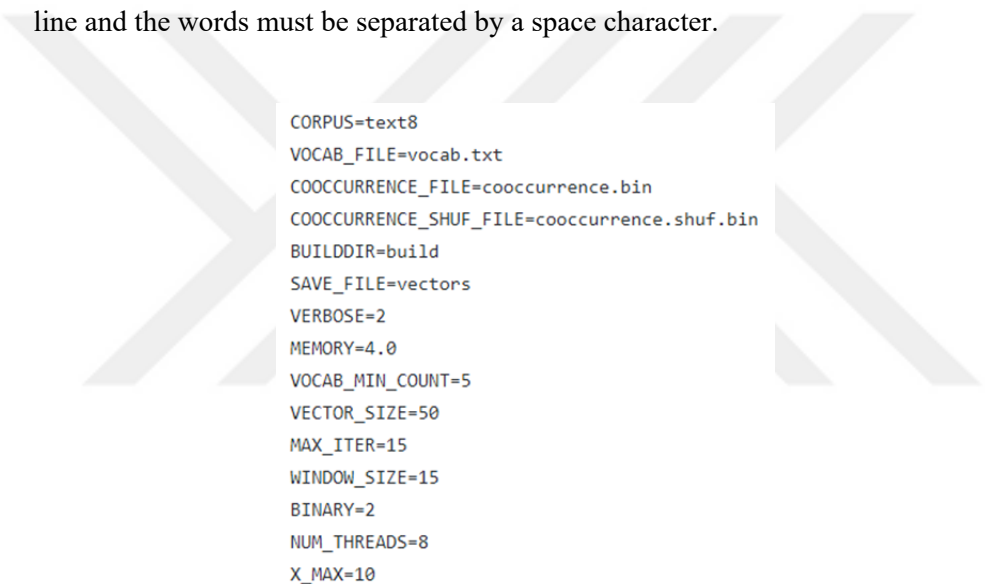
⁴ <https://radimrehurek.com/gensim/models/word2vec.html>

```
gensim.models.Fasttext.Fasttext(sentences=None, sg=0, hs=0,
size=100, window=5, min_count=5, workers=3, negative=5,
iter =5)
```

The input format for Fasttext and all parameters are the same as Word2vec.

For Glove embedding vectors; open source codes that StanfordNLP shared with GitHub are used. In order to learn the vectors from our corpus using these codes, our corpus file has given to the CORPUS variable in demo.sh file.

The corpus file, which given as input, should contain a document on each line and the words must be separated by a space character.



```
CORPUS=text8
VOCAB_FILE=vocab.txt
COOCCURRENCE_FILE=cooccurrence.bin
COOCCURRENCE_SHUF_FILE=cooccurrence.shuf.bin
BUILDDIR=build
SAVE_FILE=vectors
VERBOSE=2
MEMORY=4.0
VOCAB_MIN_COUNT=5
VECTOR_SIZE=50
MAX_ITER=15
WINDOW_SIZE=15
BINARY=2
NUM_THREADS=8
X_MAX=10
```

Figure 3.21.Parameters in Glove demo.sh file

Figure 3.21 includes a screenshot from demo.sh. The other parameters here are the same as the parameters of the embedding vectors had described earlier.

While training all embedding vectors for the sentiment classification task, vectors is trained by taking vector size as 100 and 300, window size as 5, number of iterations as 10, and minimum word count as 1. For all methods, experiments are performed by trying all possible combinations of training algorithms.

For document classification task, vectors are trained by taking vector size as 100, 200 and 300, window size as 5, number of iterations as 10, and minimum word count as 1. Experiments are performed by trying all possible combinations of training algorithms.

3.6.3. Usage of Embedding Vectors for Document Representations

In the Doc2vec representation method, while the representation vectors are obtained for the documents; in the Word2vec, Fasttext, and Glove embedding methods, vectors are obtained for words. For this reason, the methods applied by Çoban and Karabey (2017), Ayata et al. (2017), and Hayran and Sert (2017) are applied in order to obtain document vectors from the word vectors. The average, sum, and variance of embedding word vectors are taken for document representations. Figure 3.22 and Figure 3.23 summarize how document vectors are obtained from the word vectors for the sentiment analysis and document classification tasks.

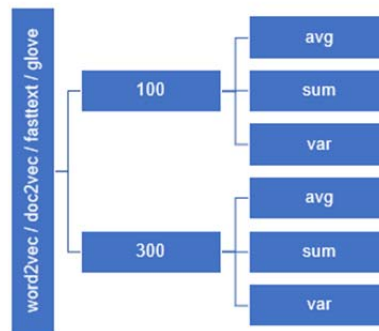


Figure 3.22. Documents representations with word embedding for the sentiment classification task

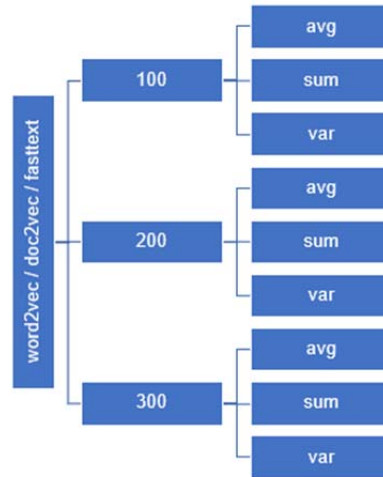


Figure 3.23. Documents representations with word embedding for the document classification task

In the sum representation model shown in Figure 3.24, each document is expressed by taking the sum of the vectors of the words contained in that document.

```

def sum_(model, docs, model_size):
    sum_vectors = []
    for doc in docs:
        doc = doc.split()
        twtVecs = np.zeros(model_size)
        for word in doc:
            try:
                c = model.wv[word]
            except:
                c = np.zeros(model_size)
            twtVecs += np.sum([c], axis=0)
        sum_vectors.append(twtVecs)
    return np.array(sum_vectors)
  
```

Figure 3.24. Sum representation model for embedding

In the average representation model (avg) shown in Figure 3.25, document representations are obtained by dividing the sum of the vectors of the words in the document by the number of words in the document.

```
def avg(model, docs, model_size):
    avg_vectors = []
    for doc in docs:
        doc = doc.split()
        twtVec = np.zeros(model_size)
        twtVecs = 0
        for word in doc:
            try:
                c = model.wv[word]
            except:
                c = np.zeros(model_size)

            twtVec += np.sum([c], axis=0)

        twtVecs = twtVec / len(doc)
        avg_vectors.append(twtVecs)

    return np.array(avg_vectors)
```

Figure 3.25. Avg representation model for embedding

In the variance representation model (var) shown in Figure 3.26, document representations are obtained by the variance of the vectors of the words in that document.

```

def var(model, docs, model_size):
    var_vectors = []
    for doc in docs:
        C = []
        doc = doc.split()
        twtVecs = np.zeros(model_size)
        for word in doc:
            try:
                c = model.wv[word]
            except:
                c = np.zeros(model_size)
            C.append(c)
        twtVecs = np.var(C, axis=0)
        var_vectors.append(twtVecs)
    return np.array(var_vectors)

```

Figure 3.26. Var representation model for embedding

3.7. Classification Process

3.7.1. Parameter Settings for Traditional Classifiers

In the classification step; Naive Bayes Multinomial, LibSVM, Random Forest and Logistic Regression classifiers, which frequently used for Turkish text classification are applied to the documents represented by using *tf* and *tf-idf*. In order to select the optimal parameters of the classifiers, a grid search is performed with 10-fold cross-validation in all classifiers in the classification step. For the LibSVM classifier, the SVC classifier, a LibSVM implementation of scikit-learn, is used. When applying the grid search; the alpha parameter for Naive Bayes Multinomial classifier, kernel, C and gamma parameters for SVC, C and penalty parameters for Logistic Regression; and criterion, max_depth and n_estimators parameters for Random Forest are used with 10-fold cross-validation. Table 3.5 summarizes the classifiers applied and the parameters used for training these classifiers.

Table 3.5. Grid search parameters

| classifier | parameters |
|-------------------------|--|
| Naïve Bayes Multinomial | alpha: [0.001, 0.01, 1, 10, 100] |
| SVC | kernel: [linear, rbf] c: [0.1, 1, 10, 100] gamma:[0.001, 0.01, 0.1, 1, 10] |
| Logistic Regression | c: [1, 10, 100] penalty:[l1, l2] |
| Random Forest | criterion: [gini, entropy] max_depth: [30,40,50] n_estimators: [100, 200, 300] |

The results obtained with the most optimal parameters in this step are the baseline for the sentiment analysis task. In this step, the parameters are taken for which achieved the highest classification success for both *tf* and *tf-idf* methods, and in the next step, the same parameters are used to classify the documents that are represented with the embedding methods.

For document classification, the Naive Bayes Multinomial, Logistic Regression, Random Forest and SVC classifiers are applied with the scikit-learn default parameters to the documents represented with *tf* and *tf-idf* respectively. Cumhuriyet and Sabah datasets are used separately, and the classifiers' success are obtained by classifying the test data of the datasets with the classifiers that are trained with training data. These classifier results are taken as the baseline for these two datasets. The classifiers and parameters used for baseline are listed in Table 3.6.

Table 3.6. Classifier parameters for document classification task

| classifier | Naïve Bayes Multinomial | Logistic Regression | Random Forest | SVC |
|------------|-------------------------|----------------------|---------------------------------------|-------------------------|
| parameters | alpha=1.0 | C=1, penalty='l2' | criterion='gini', n_estimators=100 | C=1, kernel='linear' |

3.7.2. Using Deep Learning-based Classifiers

CNN, LSTM, and artificial neural networks created by combining these two artificial neural networks are applied both for the sentiment analysis and the document classification tasks in this thesis. Network structures are developed, and their parameters are chosen based on the work of Kim (2014), and Zhang and Wallece (2017).

To apply deep learning-based methods, first of all; the inputs (the documents) for the neural network structures are prepared as follows; the length of the documents given to artificial neural networks as inputs must be constant. Therefore; first the number of words in the longest tweet in our corpus is computed and then padding is applied to other tweets having smaller length. It is observed that the longest tweet in our corpus has 24 words, so by applying padding the length of all the samples is increased to 24 words. Let w_j represents word j in the tweet i ; in this case, if there are 6 words in the $tweet_i$ the form of the tweet becomes as below after applying padding.

$$tweet_i = \langle w_1, w_2, w_3, w_4, w_5, w_6, \text{PAD}, \text{PAD}, \dots, \text{PAD} \rangle$$

For each word w_j in the tweet, word vector is computed by using each of the Word2vec, Fasttext, and Glove embedding methods and vectors of sizes 100 and 300 are generated. To generate word vectors, the embedding vectors are trained with the 20M tweet and Turkish Sentiment Datasets, respectively. Since PAD is not a defined word in these models, 0 value is chosen for all embedding methods for PAD.

Let d represents the size of the embedding vector; Figure 3.27. shows the input structure created for deep network models.

| | d | | |
|------------------|-----|---|-----|
| w_1 | | | ... |
| w_2 | | | ... |
| w_3 | | | ... |
| | . | . | . |
| | . | . | . |
| | . | . | . |
| | | | ... |
| $w_{24} (<PAD>)$ | | | ... |

Figure 3.27. Input format for deep learning-based classifiers

For document classification, a series of operations are performed to determine the input length. Since the news texts are longer than tweets and having a memory problem in the GPU based on the longest document, the average document lengths for both news datasets are used. To obtain the average document length; the documents containing more than 3000 words in both datasets are eliminated. At this stage, 66 documents from the Cumhuriyet and 195 documents from the Sabah datasets are eliminated. For the remaining documents, the average number of words in both datasets is obtained. It is observed that the average length of the document is 212 words for the Sabah dataset, and it is 258 words for the Cumhuriyet dataset. In order to represent all the words in more documents, the input length in these two datasets is determined to be 300. Following Figure 3.28 and Figure 3.29 show the number of words distribution of Cumhuriyet and Sabah datasets, respectively.

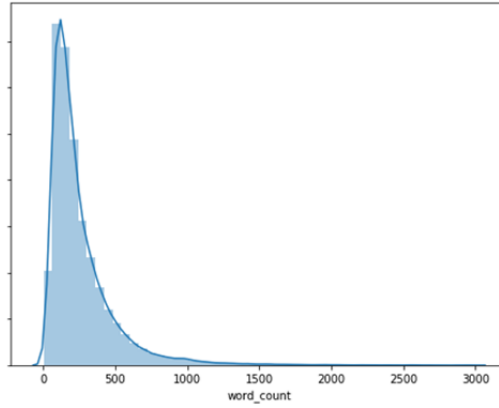


Figure 3.28. Distribution of the number of words in documents from Cumhuriyet dataset

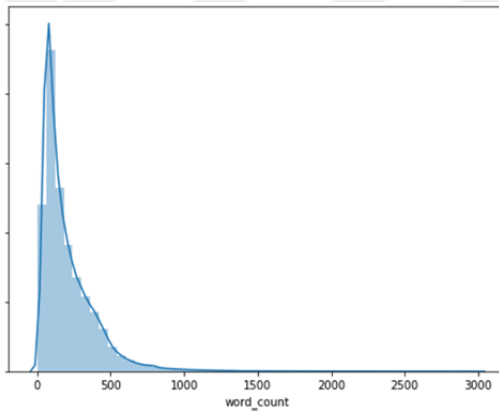


Figure 3.29. Distribution of the number of words in documents from Sabah dataset

Keras API⁵ is used for all the experiments designed for deep learning-based classifiers. During the training of the models, a structure called callback which allows us to see the changes and statistics in the model during the training is used.

In the experiments that are conducted within the scope of the thesis, the applied functions from the Keras library are:

⁵ <https://keras.io/>

- At the end of the training, the `history()` function is used to get the training accuracy, training loss, validation accuracy and validation loss values obtained for each model.

- `ModelCheckpoint()` function which records the model at the end of each iteration is called with the `val_loss` monitor parameter and `save_best_only = True` parameter. In this case; this function records the first iteration result as the best model when the model training starts. Because of the `val_loss` monitor parameter that set; the previously saved model information is updated in every iteration where `val_loss` is dropped. At the end of the training, this function saves the model with the lowest `val_loss`.

- At the point where the loss of validation begins to increase, the model begins to memorize instead of continuing to learn. For this reason, the training starts with a predetermined number of epochs. Using the `EarlyStopping()` function (with specified monitor and patience parameters), the point at which the training stops is determined. In the thesis, this `EarlyStopping()` function is used with the `val_loss` parameter. The `patience` parameter is set to 10 for the sentiment classification experiments and 5 for the document classification experiments. In the artificial neural network architectures that used in the sentiment analysis problem; a higher tolerance could be chosen such as 10 because the training time was shorter. But since the training for the document classification problem was much slower, the tolerance is taken as 5.

3.7.3. Deep Learning Models for Sentiment Analysis Task

In this section, the network structures created for sentiment analysis, and layers and parameters used to create these network structures are described.

- CNN1:

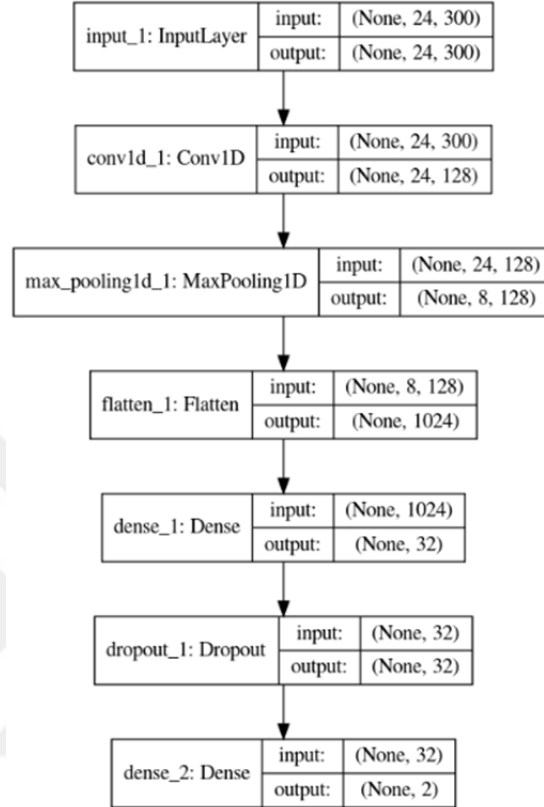


Figure 3.30. CNN1 architecture for sentiment classification task

The first architecture used in this thesis is given in Figure 3.30. The layers of this model can be explained as follows:

Input1: The input layer of the network. The dimensions in Figure 3.30, that are None, 24, 300 are the input dimensions of the network. None refers to how many instances will be given to the network i.e. batch size, 24 represents the maximum sequence length, and 300 represents the size of the embedding vectors.

Conv1d_1: This is the only convolution layer in the network architecture. In this layer, the parameters found by Kim (2014), and Zhang and Wallace (2017) are used. Layer parameters are chosen as filter number, which is 128, kernel size is equal to 3, the activation function is ReLU and the same padding is applied.

Max_pooling1d_1: is the layer in which the important features of the convolution layer are obtained. In this layer, max pooling is applied with the pool size = 3.

Flatten_1: is the layer in which pooling layers are combined and transferred to the next layers.

Dense_1: layer consists of 32 nodes with ReLU activation function. In order to prevent overfitting, l2 is applied with 0.001 as regularizer.

Dropout_1: In order to avoid overfitting, dropout applied with parameter 0.5.

Dense_2: is the output layer of the architecture. The number 2 in the layer represents the number of classes. In this layer; the softmax activation function, which expresses the probability of the samples belonging to the classes is used.

CNN3:

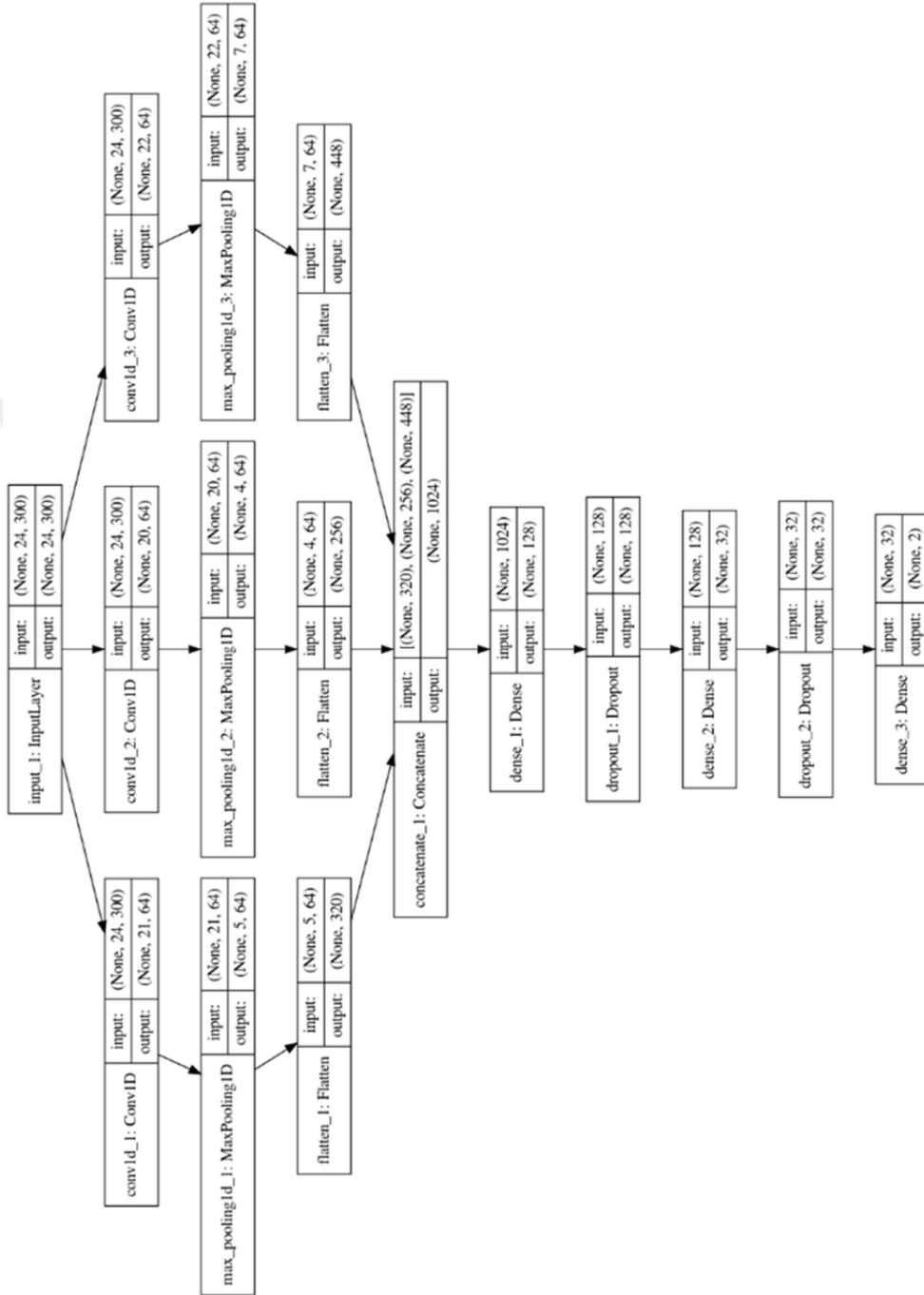


Figure 3.31. CNN3 architecture for sentiment classification task

This is the second convolutional neural network structure used in this thesis for sentiment analysis. The architecture of this network structure is presented in Figure 3.31. Compared to the first network structure, a larger network architecture is created based on the study of Kim (2014). In this architecture; 3, 4 and 5-sized kernels are identified for 3 convolution layers, each with 64 filters. 3, 4 and 5 values are used for max pooling respectively and the results are combined in flatten layers. The values from 3 flatten layers are combined in the concatenate layer. Then, in the Dense_1 and Dense_2 layers, there are 128 and 32 neurons with the ReLU activation function. In these layers, l2 regularizer is applied as kernel regularizer. In this network structure, as it is a larger architecture; dropout values are taken as 0.7 for Dropout_1 and 0.5 for Dropout_2. The last layer, as before, is the output layer with softmax activation function.

- LSTM:

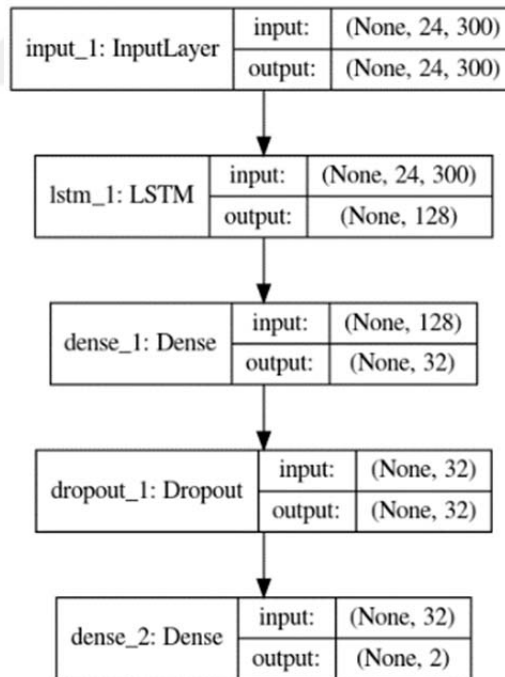


Figure 3.32. LSTM architecture for sentiment classification task

The third deep network structure used in this thesis is the LSTM architecture which is presented in Figure 3.32. In this network structure; dropout and recurrent dropout are applied with 0.3 on layer lstm_1. 32 neurons are used in the dense_1 layer with ReLU activation function and l2 kernel regularizer with parameter 0.001. Dropout is applied with 0.5 on the Dropout_1 layer. The last layer is the same as the layer described in the previous architectures.

- CNN-LSTM:

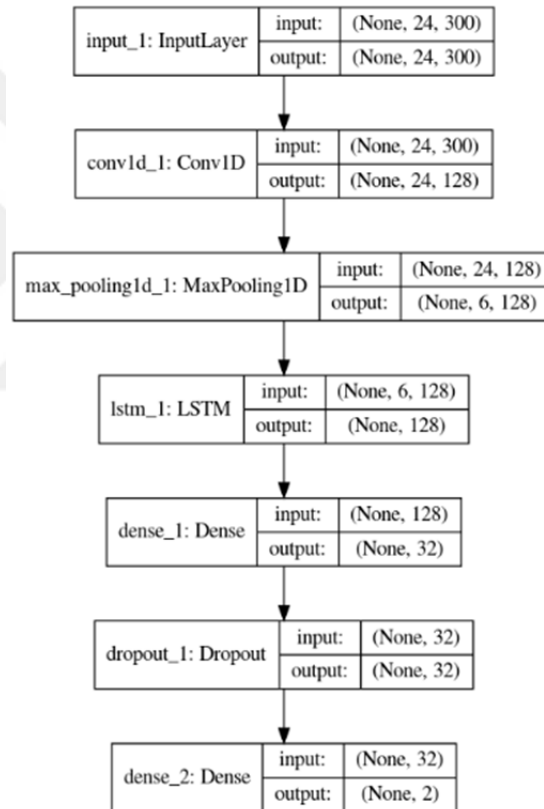


Figure 3.33. CNN-LSTM architecture for sentiment classification task

In the architecture shown in Figure 3.33, CNN and LSTM are used together, so that the convolutional layer is defined and obtain the features with 64 filters and 4-sized kernel defined in this layer. In the max pooling layer; a

maximum of 4 values are received by filtering the features obtained in the previous layer. These features that combined in the flatten layer are transferred to the LSTM layer. The features obtained in the LSTM layer are transferred to the dense_1 layer. This dense layer has the same properties as the previously described dense layers. After the dense layer, dropout is applied with 0.5. The last layer is the same as the output layer described in the previous architectures.

- CNN-LSTM2:

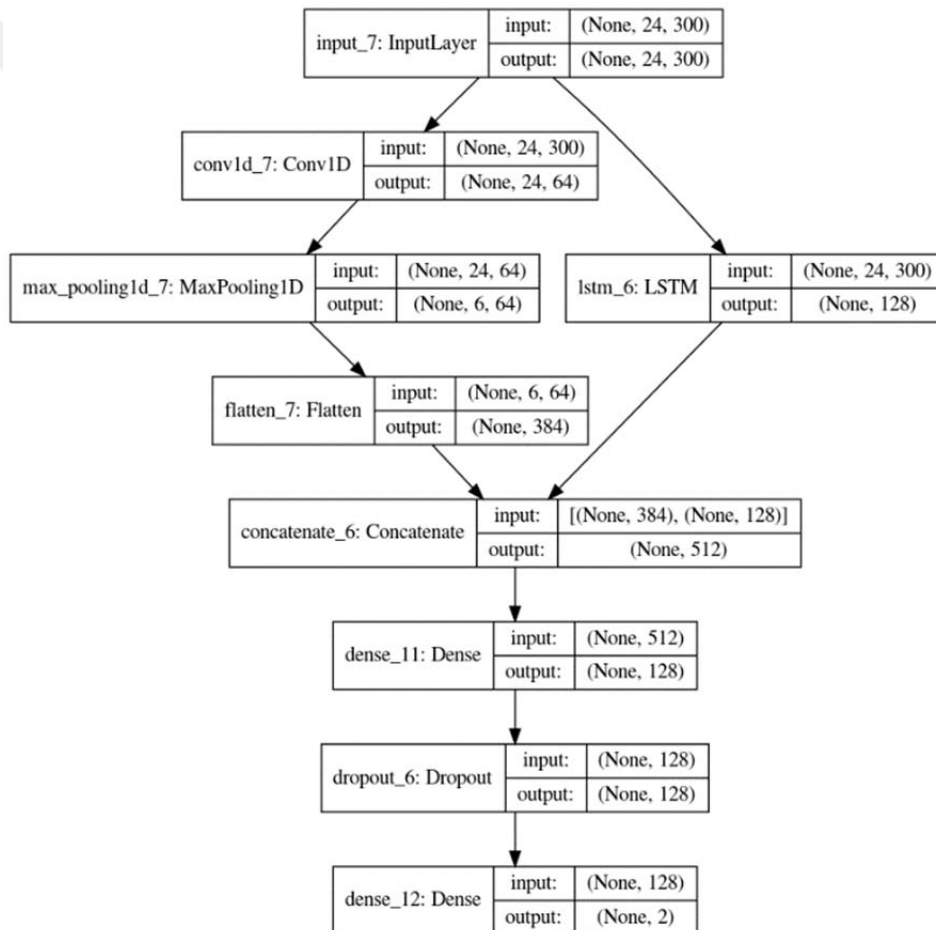


Figure 3.34. CNN-LSTM2 architecture for sentiment classification task

This architecture shown in Figure 3.34, uses CNN and LSTM together. In this architecture, an alternative architecture to the previous combination is defined. Instead of taking features just from the convolution layer; the features from the convolution and LSTM layers are combined in the concatenate layer and transferred to the next layers as shown in Figure 3.34. All parameters are used as the same as the parameters in the CNN-LSTM structure that defined earlier.

In the training phase of all models; batch size is received as 64 according to our sample numbers and GPU memory. In addition; binary cross-entropy selected as loss function, Adam optimizer applied as the optimizer, and accuracy selected as the performance metric during the training of all models. During the training of the models, 30% of the training data are used as the validation data.

3.7.4. Deep Learning Models for Document Classification Task

For the document classification problem, network architectures that are similar to the ones created for sentiment analysis task are created by using more filters and neurons in the layers. The parameters used in the architectures and the layers are explained in the below.

- CNN3:

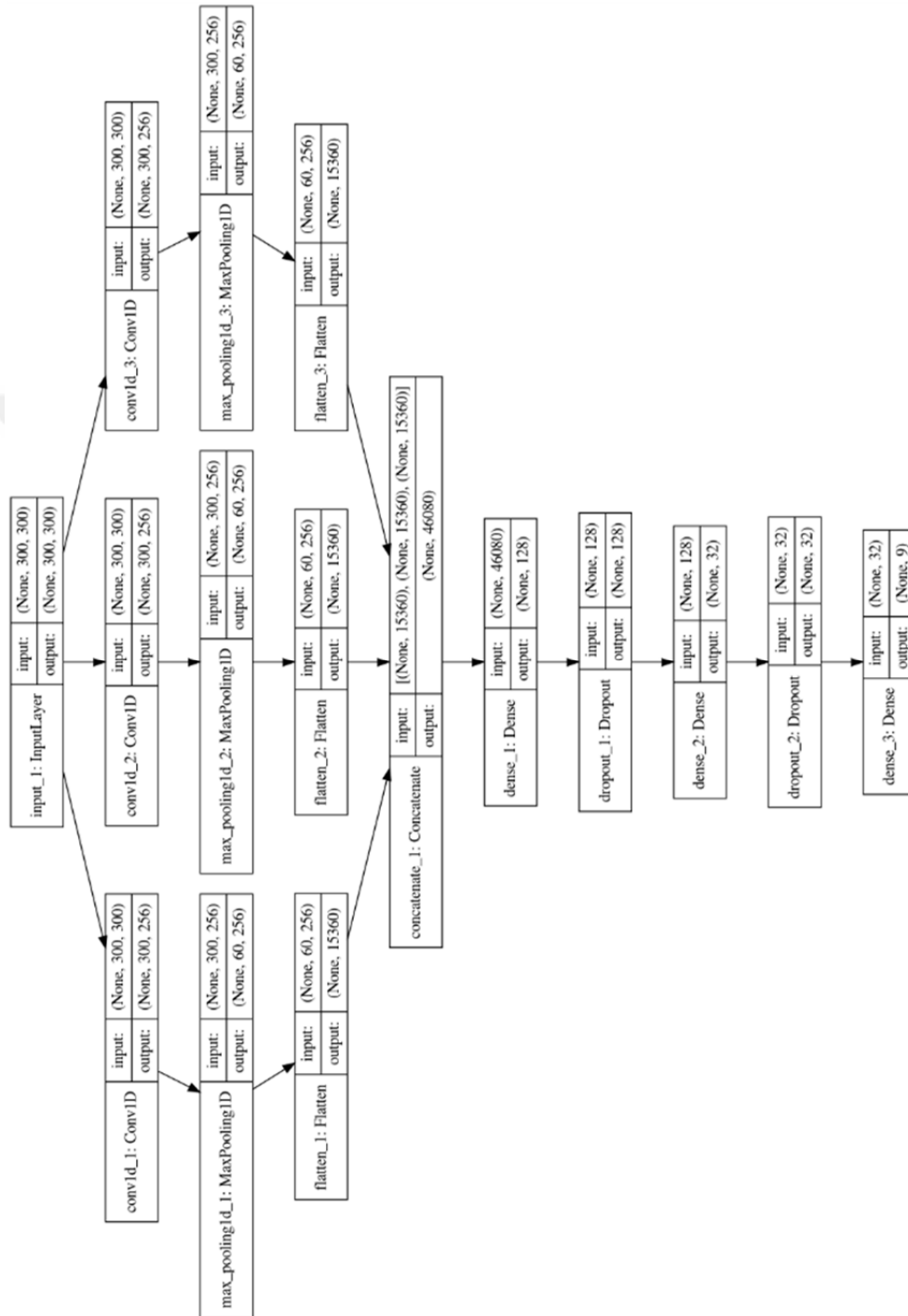


Figure 3.35. CNN3 architecture for document classification task

In this network architecture (see in Figure 3.35), there are 3 convolution layers; each with 256 filters, ReLU activation function, respectively, 4, 5, 3 sized kernels and the same padding. After each convolution layer, max pooling applied by taking pool_size as 5. In the flatten layers, the features from the pooling layers are combined. In the next concatenate layer, all the features from the flatten layers are combined. 128 neurons with ReLU activation function are identified at the dense_1 layer. To overcome the situation of underfitting observed during the experiments; 12 regularizer is not applied for all the dense layers for document classification. Also, smaller values (smaller than chosen in sentiment analysis task experiments) are used for the dropout value after the dense layers to overcome underfitting. There are 32 neurons in the dense_2 layer. In the output layer as dense_3 in Figure 3.35, there are n neurons where n is equal to the number of classes with softmax activation function.

- LSTM:

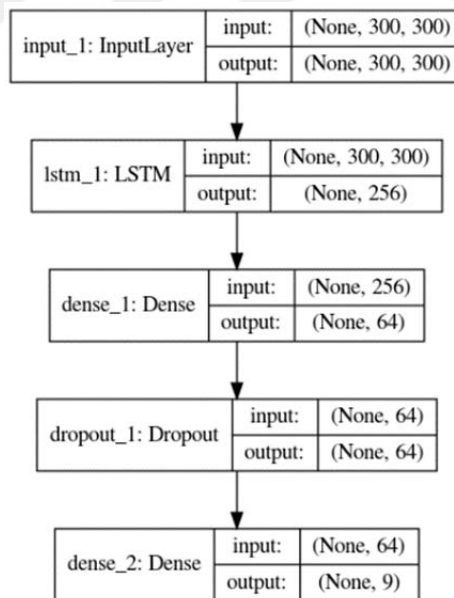


Figure 3.36. LSTM architecture for document classification task

In this network structure shown in Figure 3.36; dropout and recurrent dropout applied with 0.3 on layer lstm_1. There are 128 neurons in the dense_1 layer with ReLU activation function. Dropout is applied with 0.5 on the Dropout_1 layer. The last layer is the same as the output layers described in previous architecture.

- CNN-LSTM:

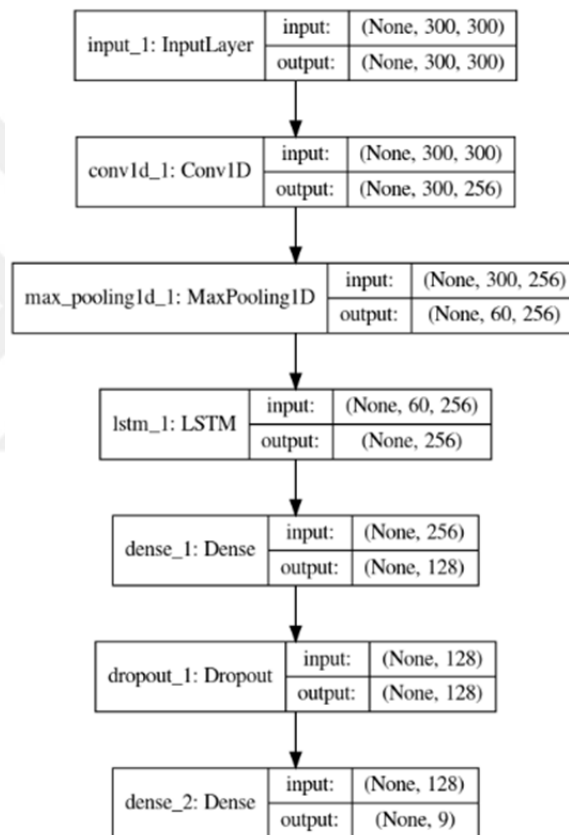


Figure 3.37. CNN-LSTM architecture for document classification task

In this architecture in Figure 3.37, where CNN and LSTM are used together, the convolutional layer is defined and the features are obtained with 256 filters, and the kernel size is 5. In the max pooling layer; a maximum of 5 values

are taken by filtering the features obtained in the previous layer. These features that combined in the flatten layer are transferred to the LSTM layer. The features obtained after the LSTM layer transferred to the dense_1 layer. This dense layer has the same properties as the previously described dense layers. After the dense layer, dropout applied with 0.5. The last layer is the output layer that contains as many neurons as the number of classes.

- CNN-LSTM2:

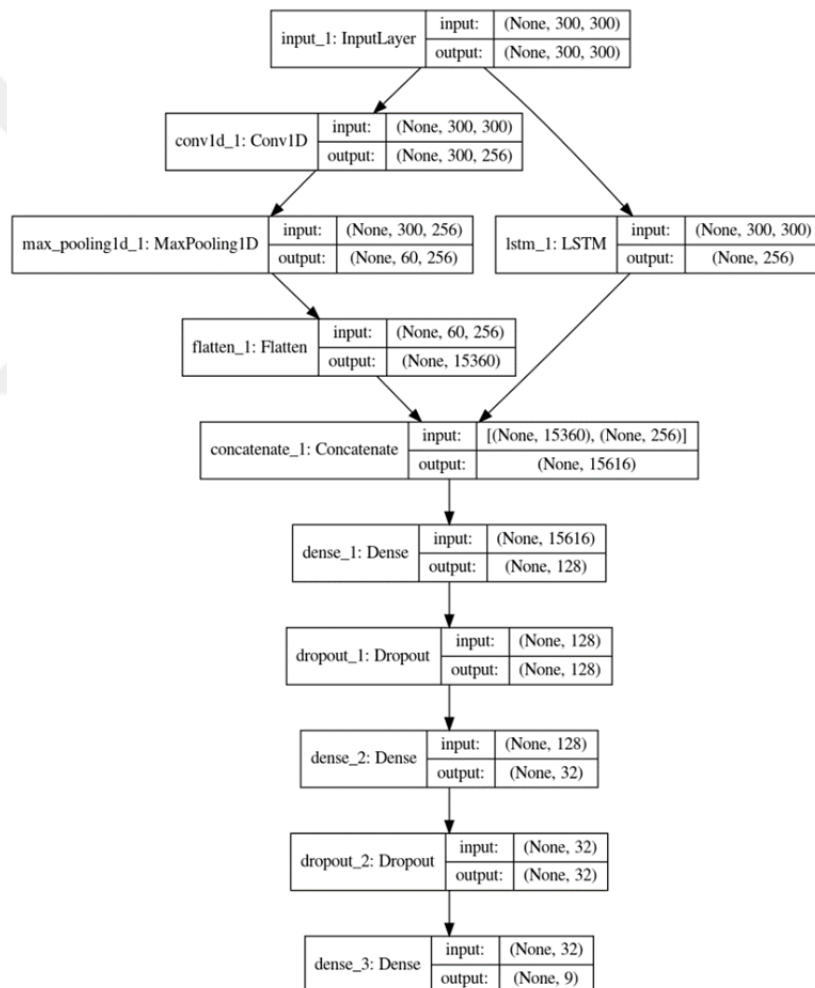


Figure 3.38. CNN-LSTM2 architecture for document classification task

In the structure in Figure 3.38, the features obtained from the CNN and LSTM layers are transferred together to the fully connected layers. In the convolution layer, there is 256 filters with kernel size 5, and 256 hidden units in LSTM layer with dropout and recurrent_dropout values of 0.3. The features of these two layers are transferred to the dense layers having 128 and 32 neurons respectively.

In the training phase of all models used; the batch size is set to 1024 according to our sample numbers and GPU memory. In addition, categorical cross-entropy used as loss function, Adam optimizer used as the optimizer, and accuracy used as the performance metric during the training of all models. During the training of models, 20% of randomly selected data based on the class distribution of the training dataset is used as the validation dataset.

4. RESULTS AND DISCUSSION

4.1. Evaluation Metrics

Performance metrics are used to measure and compare the success of the classification algorithms that are applied. In this thesis; multiclass classification is applied for document classification task and binary classification is applied for the sentiment analysis task. In order to measure the performances of the classifiers that are used in the experiments, the metrics that are described under this heading are computed.

- Confusion Matrix

It is the structure where it can be seen that the classifier performance on a matrix. Each row of the matrix represents the actual number of instances for the classes, and each column represents the number of instances estimated for the classes (in Figure 4.1.)

| | | Predicted label | |
|--------------|---------|------------------------|------------------------|
| | | Label 0 | Label 1 |
| Actual label | Label 0 | TP (true positive) | FP (false positive) |
| | Label 1 | FN (false negative) | TN (true negative) |

Figure 4.1. Confusion matrix

Classification performance metrics can be described by using the confusion matrix as follows: Precision is the ratio between correctly predicted positive samples to all samples which are predicted as positive, as in equation 4.1.

$$precision = TP / (TP + FP) \quad (4.1)$$

Recall is the ratio between correctly predicted positive samples to the samples which are actually positive (see in equation 4.2).

$$recall = TP / (TP + FN) \quad (4.2)$$

$F-1$ score is the harmonic mean of precision and recall. When working with a dataset with irregular class distributions, $F-1$ score, which is the average of both precision and recall values, is a more preferred metric of success with respect to using precision or recall alone. $F-1$ score is computed according to equation 4.3.

$$F - 1 \text{ score} = 2 \frac{precision \times recall}{precision + recall} \quad (4.3)$$

Accuracy is the ratio between correctly classified samples to all samples as in equation 4.4.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (4.4)$$

In our experiments; accuracy metric is used as a performance evaluation metric during the training of deep learning-based classifiers. The value of *micro average* $F-1$ score is taken for the measurement of classification success on the test data of deep learning-based and traditional classifiers used in this thesis.

Figure 4.2 shows the classification report of a result of one of our binary sentiment classification experiments, and Figure 4.3 shows the classification report of a result of one of our document classification experiments. The *micro average* $F-1$ score values are used in these reports.

| Classification Report | | | | |
|-----------------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.70 | 0.72 | 0.71 | 4789 |
| 1 | 0.70 | 0.68 | 0.69 | 4586 |
| micro avg | 0.70 | 0.70 | 0.70 | 9375 |
| macro avg | 0.70 | 0.70 | 0.70 | 9375 |
| weighted avg | 0.70 | 0.70 | 0.70 | 9375 |

Figure 4.2. An example of classification output of sentiment classification task

| Classification report | | | | |
|-----------------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.99 | 0.99 | 0.99 | 7839 |
| 1 | 0.92 | 0.91 | 0.92 | 5209 |
| 2 | 0.93 | 0.92 | 0.92 | 3990 |
| 3 | 0.84 | 0.87 | 0.86 | 2042 |
| 4 | 0.90 | 0.90 | 0.90 | 1975 |
| 5 | 0.89 | 0.92 | 0.91 | 1621 |
| 6 | 0.86 | 0.89 | 0.87 | 640 |
| 7 | 0.90 | 0.80 | 0.84 | 587 |
| 8 | 0.84 | 0.72 | 0.77 | 422 |
| micro avg | 0.93 | 0.93 | 0.93 | 24325 |
| macro avg | 0.90 | 0.88 | 0.89 | 24325 |
| weighted avg | 0.93 | 0.93 | 0.93 | 24325 |
| samples avg | 0.93 | 0.93 | 0.93 | 24325 |

Figure 4.3. An example of classification output of document classification task

When calculating *macro* and *micro average* F_1 score values which are the evaluation criteria for multi-class classification, following definitions are used:

To compute the macro average, performance metrics for each class are calculated independently. Then the average of the calculated values is computed by dividing to the number of classes as shown in equation 4.5. Therefore, each class has equal weight regardless of the number of samples.

$$\text{macro average } F_1 - \text{score} = \frac{\sum_{i=1}^c F_1\text{-score}_i}{c} \quad (4.5)$$

where c is the number of classes in the dataset, and $F_1\text{-score}_i$ is the $F_1\text{-score}$ for the class i .

Micro average is computed by giving equal weight to each instance in the dataset. It processes total TP , FN , and FP s. *Micro average F-1* score is the harmonic mean of micro precision (calculated by equation 4.6) and micro recall (calculated by equation 4.7), and as shown in equation 4.8.

$$\text{micro precision} = \frac{\sum_{i=1}^c TP_i}{\sum_{i=1}^c (TP_i + FP_i)} \quad (4.6)$$

$$\text{micro recall} = \frac{\sum_{i=1}^c TP_i}{\sum_{i=1}^c (TP_i + FN_i)} \quad (4.7)$$

$$\text{micro average F1 - score} = 2 \frac{\text{micro precision} \times \text{micro recall}}{\text{micro precision} + \text{micro recall}} \quad (4.8)$$

where c is the number of classes in the dataset, TP_i is the TP number for the class i , FP_i is the FP number for class i , and FN_i is the FN number of class i .

4.2. Classification Performance Measurement for Deep Learning-Based Classifiers

For all artificial neural network architectures that are used in this thesis, an output layer is defined with an equal number of neurons in the final layer. In this layer; the softmax activation function, which obtains the probability values of each class labels for each given input instance to be classified is used. When evaluating the classification performances, the class predictions are obtained for the test data by using Keras's `predict()` function for each architecture. Due to the softmax activation function, the sum of our class estimates obtained for each sample is equal to 1. Instead of specifying a threshold at this point, the class label for each instance is determined by taking the class label having the maximum probability for that instance. In the next step, using scikit-learn (with the `sklearn.metrics.classification_report()` function), a

classification report is created by using the actual classes of the test data and the estimated class values. As the performance criterion, the *micro average F1-score* is taken as in the other experiments. Figure 4.4 shows the class probabilities predicted by our model for the sentiment classification problem on the left and the classes assigned based on these estimates on the right.

| | 0 | 1 | | 0 | 1 |
|----|------------|-----------|----|---|---|
| 0 | 0.528216 | 0.471784 | 0 | 1 | 0 |
| 1 | 0.528216 | 0.471784 | 1 | 1 | 0 |
| 2 | 0.304385 | 0.695615 | 2 | 0 | 1 |
| 3 | 0.136314 | 0.863686 | 3 | 0 | 1 |
| 4 | 0.919093 | 0.0809071 | 4 | 1 | 0 |
| 5 | 0.266709 | 0.733291 | 5 | 0 | 1 |
| 6 | 0.438949 | 0.561051 | 6 | 0 | 1 |
| 7 | 0.00923072 | 0.990769 | 7 | 0 | 1 |
| 8 | 0.668868 | 0.331132 | 8 | 1 | 0 |
| 9 | 0.967078 | 0.0329225 | 9 | 1 | 0 |
| 10 | 0.778926 | 0.221074 | 10 | 1 | 0 |
| 11 | 0.0179179 | 0.982082 | 11 | 0 | 1 |
| 12 | 0.964134 | 0.0358663 | 12 | 1 | 0 |

Figure 4.4. Result of a deep learning model on test data

4.3. Results for Sentiment Classification

4.3.1. Performance of Traditional Classifiers

First, a series of experiments are performed with Naive Bayes Multinomial, LibSVM, Random Forest, and Logistic Regression to find the optimal parameters of these classifiers on the texts that are represented with BOW model using *tf* and *tf-idf* weighting schemes. A total of 106 experiments are conducted including 53 experiments for each of the texts represented by *tf* and *tf-idf*. For the two methods of representation, the best parameters of the classifiers used, are chosen.

The parameters that provide the highest classification success for each classifier and the micro average *F1-scores* for the 10-fold cross-validation on the train and the test datasets for the best parameter settings are shown in Table 4.1. According to this table while SVC classifier achieves 0.75 success with *tf*

weighting, C: 10, gamma: 0.01, kernel: rbf parameter settings; LR has the same classification success with SVC for both *tf* and *tf-idf* methods. However, considering the classifier's training time, LR trains the model in a shorter time than SVC. According to the results given in Table 4.1, SVC and Logistic regression classifiers have the highest classification success, whereas Naïve Bayes Multinomial is the second best, and the Random Forest is the worst one for the sentiment analysis.

Table 4.1. Experimental results of traditional classifiers using bag-of-words method for sentiment analysis

| classifier | weighting method | parameters | best cross-validation f-1 score | test f-1 score |
|-------------------------|------------------|--|---------------------------------|----------------|
| Naïve Bayes Multinomial | tf | alpha: 1 | 0.73 | 0.74 |
| | tf-idf | alpha: 10 | 0.73 | 0.74 |
| SVC | tf | C: 10 gamma: 0.01 kernel: rbf | 0.74 | 0.75 |
| | tf-idf | C: 1 gamma: 0.001 kernel: rbf | 0.74 | 0.74 |
| Logistic Regression | tf | C: 1 penalty: l2 | 0.74 | 0.75 |
| | tf-idf | C: 1 penalty: l2 | 0.74 | 0.75 |
| Random Forest | tf | criterion: entropy max_depth: 50 n_estimators: 300 | 0.70 | 0.70 |
| | tf-idf | criterion: entropy max_depth: 50 n_estimators: 300 | 0.70 | 0.70 |

Also, the results presented in Table 4.1 forms our baseline results for the sentiment classification task. In all subsequent experimental results, two sets of parameters are used for SVM called as SVC-1 and SVC-2, where SCV-1 is the SVC classifier with parameters C=10, and gamma=0.01 that give the best

performance for *tf* weighting. SCV-2 is the SVC classifier with parameters $C=1$ and $\gamma=0.001$ that give the best performance for *tf-idf* weighting.

In Table 4.2, the classification performances for sentiment analysis problem of the traditional classifiers are shown when using embedding methods to obtain word vectors. In this experiment, the performances of word vectors are compared by setting vector dimension to 100 and 300. In this experiment, 100 and 300-dimensional word vectors are obtained by applying Word2vec to Turkish Sentiment Dataset. Documents are represented by taking the average, sum, and variance of the Word2vec word vectors. The class labels for the test dataset is estimated with the models learned through training dataset. The *micro average F-1* score is used to measure models' successes.

Table 4.2. Experimental results of traditional classifiers using Word2vec embedding vectors trained from TSD

| | | 100 | | | | 300 | | | |
|-----------|-----|-------------|-------------|-------------|-----------|-------------|-------------|-------------|-------------|
| | | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| LR | avg | 0.63 | 0.60 | 0.62 | 0.62 | 0.63 | 0.60 | 0.65 | 0.63 |
| | sum | 0.63 | 0.63 | 0.63 | 0.63 | 0.65 | 0.63 | 0.65 | 0.66 |
| | var | 0.63 | 0.61 | 0.61 | 0.62 | 0.66 | 0.60 | 0.64 | 0.62 |
| RF | avg | 0.71 | 0.68 | 0.73 | 0.69 | 0.71 | 0.68 | 0.74 | 0.69 |
| | sum | 0.71 | 0.68 | 0.72 | 0.69 | 0.71 | 0.68 | 0.73 | 0.69 |
| | var | 0.69 | 0.68 | 0.69 | 0.68 | 0.69 | 0.68 | 0.69 | 0.68 |
| SVC- 1 | avg | 0.68 | 0.62 | 0.68 | 0.65 | 0.68 | 0.62 | 0.68 | 0.65 |
| | sum | 0.69 | 0.66 | 0.72 | 0.68 | 0.70 | 0.65 | 0.73 | 0.68 |
| | var | 0.64 | 0.59 | 0.61 | 0.61 | 0.65 | 0.58 | 0.63 | 0.58 |
| SVC- 2 | avg | 0.62 | 0.55 | 0.59 | 0.57 | 0.61 | 0.55 | 0.61 | 0.56 |
| | sum | 0.67 | 0.63 | 0.68 | 0.64 | 0.67 | 0.62 | 0.68 | 0.65 |
| | var | 0.61 | 0.52 | 0.51 | 0.51 | 0.54 | 0.51 | 0.51 | 0.51 |

In Table 4.2, LR, RF, SVC stand for Logistic regression, random forest, and support vector machine classifier, respectively. In the columns of this table CBOW, SG, HS, and NS mean Continuous Bag-of-Words, Skip Gram, Hierarchical Softmax, and Negative Sampling, respectively. According to Table 4.2, the highest success is obtained by random forest classifier when the average of

300-dimensional Word2vec vectors trained by using skip-gram architecture and hierarchical softmax is used as the document representation method. In addition, it is found that skip-gram architecture yields better results than CBOW and the hierarchical softmax algorithm yields better than negative sampling for both architectures. When these results are compared with the baseline results given in Table 4.1, it can be easily seen that using word embedding methods to represent texts for traditional classifiers reduces classification score except for the random forest classifier.

In Table 4.3 experimental results are presented for the sentiment analysis task when word vectors are trained from 20M tweets dataset. According to Table 4.3, the success of the negative sampling algorithm has increased with the increase of the dataset size in which the word vectors trained. The highest classification success rate is 74% with RF and SVC-1 classifiers. However, when looking at the overall table, it is observed that the RF classifier is more successful than the other classifiers.

Table 4.3. Experimental results of traditional classifiers using Word2vec embedding vectors trained from 20M tweets dataset

| | | 100 | | | | 300 | | | |
|-------|-----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| LR | avg | 0.66 | 0.69 | 0.68 | 0.69 | 0.67 | 0.70 | 0.69 | 0.70 |
| | sum | 0.66 | 0.69 | 0.68 | 0.69 | 0.67 | 0.70 | 0.70 | 0.70 |
| | var | 0.59 | 0.60 | 0.58 | 0.59 | 0.60 | 0.63 | 0.59 | 0.60 |
| RF | avg | 0.72 | 0.74 | 0.73 | 0.74 | 0.71 | 0.73 | 0.73 | 0.73 |
| | sum | 0.71 | 0.73 | 0.73 | 0.73 | 0.71 | 0.73 | 0.73 | 0.73 |
| | var | 0.66 | 0.67 | 0.66 | 0.66 | 0.66 | 0.68 | 0.66 | 0.66 |
| SVC-1 | avg | 0.71 | 0.73 | 0.69 | 0.71 | 0.73 | 0.74 | 0.71 | 0.73 |
| | sum | 0.68 | 0.65 | 0.72 | 0.73 | 0.67 | 0.65 | 0.73 | 0.73 |
| | var | 0.66 | 0.66 | 0.57 | 0.60 | 0.66 | 0.65 | 0.59 | 0.61 |
| SVC-2 | avg | 0.67 | 0.71 | 0.66 | 0.68 | 0.68 | 0.72 | 0.67 | 0.70 |
| | sum | 0.71 | 0.72 | 0.68 | 0.71 | 0.72 | 0.70 | 0.71 | 0.72 |
| | var | 0.60 | 0.67 | 0.51 | 0.51 | 0.61 | 0.69 | 0.51 | 0.54 |

It can be concluded more meaningful results for Word2vec embedding method when Table 4.2 and Table 4.3 are compared. First of all, negative sampling is found to be more successful when increasing the size of the data from which vectors trained. In addition to this, it can be said that the increase in the size of the vector dimension also increases the classification success slightly. In order to observe the effect of the dataset used for training the word vectors on the classification success, we should examine the classification *micro average F-1* scores that are greater than 0.70 in the two tables. In Table 4.3, there are 42 cases where the classification F1-score is greater than 0.70, whereas in Table 4.2 there are only 11 cases where the *micro average F-1* score is greater than 0.70. Based on this observation; it can be concluded that the size of the dataset in which the word vectors are trained, positively effects the classification success. Although using a very large dataset to train word vectors and using 300-dimensional word vectors, classification performance could not reach the baseline results given in Table 4.1. Therefore, for sentiment analysis task, using traditional classifiers with traditional text representation methods (*tf* and *tf-idf*) yields more successful class label assignments.

Table 4.4 and Table 4.5 illustrates the classification performances of traditional classifiers when documents are represented by using the vectors learned by the Doc2vec method that are trained by doing 10-iterations on TSD and 20M tweets datasets respectively. Doc2vec is an algorithm that includes class or tag information in its vector calculations. For this reason, when Table 4.5 is examined, it is observed that the classification performance of the document vectors that are trained from 20M tweets is very low from the baseline, as 20M tweets dataset does not contain class label information of the tweets. However, when using the document vectors that are learned from the TSD dataset, it is observed that classification accuracies of the traditional classifiers increase up to 14% over the baseline results as shown in Table 4.4. According to Table 4.4, DBOW-HS has the highest classification success for all classifiers and classification F1-measure is

increased up to 0.89 for LR, RF, and SVC-1 classifiers. There is no difference between using 100 or 300-dimensional document vectors. From these experiments, when traditional classifiers are used for sentiment analysis task, representing tweets by using DBOW which trained on the original dataset has the best classification performance.

Table 4.4. Experimental results of traditional classifiers using Doc2vec embedding vectors trained from TSD

| | | LR | RF | SVC- 1 | SVC-2 |
|-----|-----------|-------------|-------------|-------------|-------------|
| 100 | DBOW – HS | 0.89 | 0.89 | 0.89 | 0.87 |
| | DBOW – NS | 0.84 | 0.84 | 0.84 | 0.65 |
| | DM – HS | 0.81 | 0.83 | 0.81 | 0.77 |
| | DM – NS | 0.75 | 0.79 | 0.75 | 0.60 |
| 300 | DBOW – HS | 0.89 | 0.89 | 0.89 | 0.87 |
| | DBOW – NS | 0.84 | 0.84 | 0.84 | 0.65 |
| | DM – HS | 0.81 | 0.84 | 0.81 | 0.77 |
| | DM – NS | 0.75 | 0.78 | 0.75 | 0.59 |

Table 4.5. Experimental results of traditional classifiers using Doc2vec embedding vectors trained from 20M tweets

| | | LR | RF | SVC- 1 | SVC-2 |
|-----|-----------|-------------|-------------|-------------|-------------|
| 100 | DBOW – HS | 0.59 | 0.66 | 0.61 | 0.57 |
| | DBOW – NS | 0.59 | 0.64 | 0.59 | 0.51 |
| | DM – HS | 0.58 | 0.66 | 0.62 | 0.59 |
| | DM – NS | 0.57 | 0.60 | 0.58 | 0.56 |
| 300 | DBOW – HS | 0.62 | 0.67 | 0.62 | 0.59 |
| | DBOW – NS | 0.60 | 0.64 | 0.60 | 0.51 |
| | DM – HS | 0.63 | 0.66 | 0.66 | 0.61 |
| | DM – NS | 0.58 | 0.61 | 0.59 | 0.55 |

In the next experiment, the performance of using Fasttext to generate word vectors to represent documents for sentiment analysis task is measured. First, Fasttext vectors are trained by using the TSD dataset, then vectors for the tweets are computed by taking the average, sum, and variance of the vectors of the words contained in the tweets. After that, traditional classifiers are applied to make sentiment analysis. The results of this experiment are presented in Table 4.6. The same procedure is repeated but this time word vectors are learned from the large 20M tweets dataset. And the results of this experiment are summarized in Table 4.7.

Table 4.6. Experimental results of traditional classifiers using Fasttext embedding vectors trained from TSD

| | | 100 | | | | 300 | | | |
|-------|-----|-------------|-------------|-------------|-----------|-------------|-------------|-------------|-------------|
| | | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| LR | avg | 0.63 | 0.63 | 0.66 | 0.65 | 0.65 | 0.61 | 0.69 | 0.65 |
| | sum | 0.63 | 0.65 | 0.67 | 0.65 | 0.67 | 0.65 | 0.69 | 0.67 |
| | var | 0.61 | 0.60 | 0.61 | 0.62 | 0.64 | 0.61 | 0.64 | 0.63 |
| RF | avg | 0.71 | 0.70 | 0.73 | 0.71 | 0.71 | 0.69 | 0.73 | 0.72 |
| | sum | 0.71 | 0.70 | 0.73 | 0.71 | 0.71 | 0.70 | 0.73 | 0.71 |
| | var | 0.68 | 0.68 | 0.68 | 0.70 | 0.69 | 0.69 | 0.69 | 0.70 |
| SVC-1 | avg | 0.69 | 0.65 | 0.70 | 0.68 | 0.69 | 0.65 | 0.71 | 0.67 |
| | sum | 0.70 | 0.67 | 0.72 | 0.70 | 0.70 | 0.67 | 0.73 | 0.69 |
| | var | 0.66 | 0.61 | 0.60 | 0.62 | 0.66 | 0.60 | 0.63 | 0.61 |
| SVC-2 | avg | 0.61 | 0.58 | 0.64 | 0.62 | 0.63 | 0.58 | 0.66 | 0.61 |
| | sum | 0.68 | 0.64 | 0.70 | 0.68 | 0.68 | 0.64 | 0.71 | 0.67 |
| | var | 0.60 | 0.55 | 0.51 | 0.56 | 0.60 | 0.51 | 0.51 | 0.51 |

For this experiment; similar results are observed for both Fasttext and Word2vec when word vectors are trained from the smaller dataset, as shown in Table 4.6. The success of the hierarchical softmax algorithm is higher if the training dataset size is small. There are no significant differences between the vector sizes that are 100 and 300. In general, using the average of the word vectors to represent tweets has slightly better performance. RF and SVC-1 classifiers have the best performances for this tweet representation method. However, the

classification success of SVC-1 classifier is behind the baseline, while using Fasttext improves classification success of the RF classifier.

In Table 4.7, results are presented for sentiment analysis task when word vectors are generated by using Fasttext trained on 20M tweet dataset. As shown in the table, the classification success for the negative sampling algorithm increases with respect to training on the small dataset. According to this table, although using word vectors generated by Fasttext could not pass the baseline performance for the LR classifier, better results are obtained with RF and SVC classifiers with the average and sum of 100 and 300-dimensional word vectors obtained using skip-gram architecture and negative sampling algorithm.

Table 4.7. Experimental results of traditional classifiers using Fasttext embedding vectors trained from 20M tweets

| | | 100 | | | | 300 | | | |
|--------|-----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| LR | avg | 0.66 | 0.69 | 0.67 | 0.69 | 0.68 | 0.70 | 0.70 | 0.70 |
| | sum | 0.66 | 0.69 | 0.67 | 0.69 | 0.69 | 0.71 | 0.71 | 0.71 |
| | var | 0.60 | 0.62 | 0.58 | 0.59 | 0.61 | 0.64 | 0.60 | 0.61 |
| RF | avg | 0.73 | 0.74 | 0.74 | 0.75 | 0.72 | 0.73 | 0.73 | 0.74 |
| | sum | 0.73 | 0.74 | 0.74 | 0.75 | 0.72 | 0.73 | 0.74 | 0.74 |
| | var | 0.67 | 0.69 | 0.66 | 0.67 | 0.68 | 0.69 | 0.66 | 0.66 |
| SVC- 1 | avg | 0.73 | 0.74 | 0.69 | 0.72 | 0.74 | 0.74 | 0.72 | 0.73 |
| | sum | 0.68 | 0.65 | 0.72 | 0.74 | 0.67 | 0.65 | 0.74 | 0.75 |
| | var | 0.67 | 0.65 | 0.59 | 0.60 | 0.68 | 0.65 | 0.62 | 0.62 |
| SVC-2 | avg | 0.67 | 0.73 | 0.65 | 0.68 | 0.69 | 0.73 | 0.66 | 0.69 |
| | sum | 0.72 | 0.69 | 0.69 | 0.71 | 0.73 | 0.69 | 0.71 | 0.73 |
| | var | 0.63 | 0.68 | 0.51 | 0.51 | 0.63 | 0.69 | 0.51 | 0.51 |

When Table 4.6 and Table 4.7 are evaluated together, by using the larger dataset to learn Fasttext word vectors, it is possible to generate better word vectors and they positively affect the classification performance of the traditional classifiers.

In the next experiment, the performance of the Glove embedding method is analyzed for the sentiment analysis task when the traditional classifiers are used. The results of these experiments are presented in Table 4.8 and 4.9. If Table 4.8.

and Table 4.9 are evaluated together, representing tweets by using the word vectors that are generated by the Glove embedding vectors cannot improve the classification success of the traditional classifiers with respect to the baseline results. But for Glove embedding vectors, the RF classifier has the best classification performance, as in the other embedding methods. Also, if the two tables are compared, it could be said that as the size of the dataset where the vectors are trained increases, the classification success of the traditional classifiers increases as for the other embedding methods.

Table 4.8. Experimental results of traditional classifiers using Glove embedding vectors trained from TSD

| | | 100 | 300 |
|-------|-----|-------------|-------------|
| LR | avg | 0.62 | 0.62 |
| | sum | 0.63 | 0.65 |
| | var | 0.60 | 0.60 |
| RF | avg | 0.69 | 0.68 |
| | sum | 0.69 | 0.69 |
| | var | 0.67 | 0.68 |
| SVC-1 | avg | 0.62 | 0.62 |
| | sum | 0.66 | 0.65 |
| | var | 0.58 | 0.56 |
| SVC-2 | avg | 0.59 | 0.58 |
| | sum | 0.62 | 0.62 |
| | var | 0.51 | 0.51 |

Table 4.9. Experimental results of traditional classifiers using Glove embedding vectors trained from 20M tweets

| | | 100 | 300 |
|-------|-----|-------------|-------------|
| LR | avg | 0.63 | 0.65 |
| | sum | 0.64 | 0.66 |
| | var | 0.62 | 0.64 |
| RF | avg | 0.70 | 0.71 |
| | sum | 0.71 | 0.71 |
| | var | 0.68 | 0.69 |
| SVC-1 | avg | 0.65 | 0.66 |
| | sum | 0.68 | 0.68 |
| | var | 0.62 | 0.62 |
| SVC-2 | avg | 0.60 | 0.60 |
| | sum | 0.65 | 0.65 |
| | var | 0.56 | 0.55 |

In all the experiments performed in this step, Python version 3.6.7 programming language, Scikit-learn version 0.20.3 machine learning library for Python, Numpy version 1.16.3 library for mathematical operations, and Pandas version 0.24.2 library for data analysis and data processing, are used.

4.3.2. Performance of Deep Learning based Classifiers

GeForce GTX 1050 with 4 Gb memory, 418.56 version NVIDIA Driver, Ubuntu 18.04.2 LTS operating system, Cuda 10.1, Cuda toolkit 10.0.130, Cudnn 7.3.1, Keras 2.2.4, and Tensorflow 1.13.1 are used in the experiments performed in this section.

In Table 4.10, the classification performances can be seen on the test dataset when the CNN architecture with a convolution layer followed by pooling, flatten and dense layers are used together with embedding methods. According to this table, the highest classification success is achieved as *0.73 micro average F-1*

score with training Fasttext embedding method on 20M tweet using the skip-gram architecture. Performance of Word2vec embedding method is similar to Fasttext. However, Glove vectors have a lower performance with respect to the two methods and achieve the highest classification success rate of 0.68.

Table 4.10. Experimental results of CNN for TSD

| | 100 | | | | 300 | | | |
|-----------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| 20M Word2vec | 0.69 | 0.71 | 0.71 | 0.72 | 0.69 | 0.71 | 0.72 | 0.72 |
| TSD Word2vec | 0.69 | 0.64 | 0.71 | 0.65 | 0.71 | 0.63 | 0.70 | 0.66 |
| 20M Fasttext | 0.70 | 0.71 | 0.72 | 0.71 | 0.69 | 0.71 | 0.72 | 0.73 |
| TSD Fasttext | 0.69 | 0.66 | 0.71 | 0.68 | 0.68 | 0.65 | 0.71 | 0.68 |
| 20M Glove | 0.67 | | | | 0.68 | | | |
| TSD Glove | 0.65 | | | | 0.64 | | | |

The performance of the CNN3 architecture, which is created by sending the input to 3 different convolution layers, is summarized in Table 4.11. According to this table, the highest classification success is achieved as 0.73 with the Word2vec and Fasttext vectors, which are trained over 20M tweet dataset as in previous CNN architecture. There is no significant difference between the classification performances obtained for the 100 and 300-dimensional vectors if it is necessary to make a comparison between the dimensions of the embedding vectors. However, if a comparison is made between the Word2vec, Fasttext, and Glove embedding vectors, which have been trained for 10 iterations, Fasttext vectors have the highest classification success in both CNN architectures.

Table 4.11. Experimental results of CNN3 for TSD

| | 100 | | | | 300 | | | |
|-----------------|-------------|-------------|-----------|-------------|-------------|-------------|-------------|-------------|
| | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| 20M Word2vec | 0.70 | 0.72 | 0.69 | 0.71 | 0.69 | 0.71 | 0.71 | 0.71 |
| TSD Word2vec | 0.69 | 0.64 | 0.70 | 0.66 | 0.69 | 0.63 | 0.70 | 0.65 |
| 20M Fasttext | 0.71 | 0.73 | 0.71 | 0.72 | 0.70 | 0.72 | 0.72 | 0.73 |
| TSD Fasttext | 0.68 | 0.66 | 0.70 | 0.69 | 0.67 | 0.65 | 0.71 | 0.68 |
| 20M Glove | 0.67 | | | | 0.67 | | | |
| TSD Glove | 0.65 | | | | 0.65 | | | |

In Table 4.12, the classification performances can be seen for the network architecture that contains a single LSTM layer. This architecture has a performance of over 0.70 when the Word2vec and Fasttext vectors which are trained over 20M tweet data are used. For the first time, the Word2vec vectors, which are trained on the TSD, have 0.77 classification performance and pass the baseline which is determined as 0.75. As previously mentioned, if the amount of training data is small, the embedding vectors trained using the hierarchical softmax algorithm are found more successful than the vectors trained using the negative sampling algorithm.

Table 4.12. Experimental results of LSTM for TSD

| | 100 | | | | 300 | | | |
|-----------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------|
| | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| 20M Word2vec | 0.70 | 0.73 | 0.72 | 0.73 | 0.71 | 0.73 | 0.73 | 0.73 |
| TSD Word2vec | 0.73 | 0.65 | 0.76 | 0.67 | 0.76 | 0.65 | 0.77 | 0.66 |
| 20M Fasttext | 0.72 | 0.73 | 0.72 | 0.74 | 0.72 | 0.73 | 0.73 | 0.73 |
| TSD Fasttext | 0.70 | 0.68 | 0.72 | 0.70 | 0.71 | 0.67 | 0.72 | 0.70 |
| 20M Glove | 0.68 | | | | 0.68 | | | |
| TSD Glove | 0.66 | | | | 0.66 | | | |

As the next experiment, CNN and LSTM architectures are combined in two different ways and their classification performance are compared for the sentiment analysis task. The results of this experiment are presented in Table 4.13 and Table 4.14. Table 4.13. illustrates the results of the network architecture in which a convolution layer is followed by an LSTM layer. The highest classification success for this architecture is observed as 0.74 when 300-dimensional word vectors generated by Fasttext learned over 20M tweets. Table 4.14. shows the results for the architecture obtained by feeding the input to a separate convolution and an LSTM layer. The highest classification success for this architecture is observed as 0.72 when both Word2vec and Fasttext embedding methods are used with negative sampling and training is done over the 20M tweets dataset. Classification success of the first CNN-LSTM architecture is slightly better with respect to the second architecture. When these results are compared with the performances of using CNN and LSTM alone, it can be concluded that using LSTM alone has better classification success. Because the LSTM network uses historical knowledge, it can achieve more successful results in NLP problems.

Table 4.13. Experimental results of CNN-LSTM for TSD

| | 100 | | | | 300 | | | |
|-----------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| 20M Word2vec | 0.70 | 0.72 | 0.72 | 0.73 | 0.71 | 0.72 | 0.73 | 0.73 |
| TSD Word2vec | 0.71 | 0.64 | 0.72 | 0.66 | 0.73 | 0.64 | 0.73 | 0.67 |
| 20M Fasttext | 0.72 | 0.73 | 0.72 | 0.73 | 0.71 | 0.71 | 0.73 | 0.74 |
| TSD Fasttext | 0.69 | 0.67 | 0.71 | 0.69 | 0.70 | 0.66 | 0.71 | 0.69 |
| 20M Glove | 0.68 | | | | 0.69 | | | |
| TSD Glove | 0.66 | | | | 0.66 | | | |

Table 4.14. Experimental results of CNN-LSTM2 for TSD

| | 100 | | | | 300 | | | |
|-----------------|-------------|-------------|-----------|-------------|-------------|-------------|-------------|-------------|
| | CBOW- HS | CBOW- NS | SG- HS | SG- NS | CBOW- HS | CBOW- NS | SG- HS | SG- NS |
| 20M Word2vec | 0.68 | 0.71 | 0.70 | 0.72 | 0.68 | 0.72 | 0.71 | 0.71 |
| TSD Word2vec | 0.67 | 0.64 | 0.68 | 0.66 | 0.67 | 0.64 | 0.71 | 0.65 |
| 20M Fasttext | 0.70 | 0.71 | 0.70 | 0.72 | 0.69 | 0.72 | 0.72 | 0.72 |
| TSD Fasttext | 0.68 | 0.66 | 0.69 | 0.68 | 0.68 | 0.65 | 0.70 | 0.68 |
| 20M Glove | 0.67 | | | | 0.67 | | | |
| TSD Glove | 0.64 | | | | 0.64 | | | |

4.4. Results for Document Classification

4.4.1. Performance of Traditional Classifiers

In the previous section, results are presented for sentiment analysis task in which each document consists of short texts with lots of spelling and grammar errors. In this section, the same experiments are repeated for news classification task where text have longer texts without spelling and grammar errors to show that

whether the used methods are sensitive to document lengths and spelling errors or not.

For document classification, experiments are conducted separately on Cumhuriyet and Sabah datasets. However, as mentioned in the dataset section, Cumhuriyet and Sabah datasets are used together during the training of embedding methods since larger data from the same domain is not accessible. Also, in our experiments for document classification in this section, default parameters defined in the scikit-learn library are used for classifiers.

In Table 4.15, it can be seen the classification performance of Naive Bayes Multinomial, Logistic Regression, Random Forest and Support Vector Machines classifiers for the 9-class Cumhuriyet dataset when traditional document representation method (BOW) used with *tf* and *tf-idf* weighting schemes. These results form our baseline results. As shown in Table 4.15, the highest classification success is obtained by the LR classifier for both *tf* and *tf-idf* which is equal to 0.93.

Table 4.15. Experimental results of traditional classifiers using BOW method for Cumhuriyet dataset

| | Naïve Bayes Multinomial | | Logistic Regression | | Random Forest | | SVC | |
|------------------|-------------------------|--------|---------------------|-------------|------------------------------------|--------|----------------------|--------|
| parameters | alpha=1.0 | | C=1, penalty='l2' | | criterion='gini', n_estimators=100 | | C=1, kernel='linear' | |
| weighting method | tf | tf-idf | tf | tf-idf | tf | tf-idf | tf | tf-idf |
| F1 micro avg | 0.91 | 0.91 | 0.93 | 0.93 | 0.87 | 0.87 | 0.91 | 0.92 |
| F1 macro avg | 0.85 | 0.87 | 0.89 | 0.89 | 0.75 | 0.75 | 0.87 | 0.88 |
| F1 weighted avg | 0.91 | 0.92 | 0.93 | 0.93 | 0.86 | 0.86 | 0.91 | 0.92 |

Table 4.16 shows the performances of traditional classifiers for 4-class Sabah dataset when BOW model with *tf* and *tf-idf* weighting is used to represent

the documents. For the Sabah dataset, the highest success is achieved by LR as 0.89 in both representations.

Table 4.16. Experimental results of traditional classifiers using BOW method for Sabah dataset

| | Naïve Bayes Multinomial | | Logistic Regression | | Random Forest | | SVC | |
|------------------|-------------------------|--------|---------------------|-------------|------------------------------------|--------|----------------------|--------|
| parameters | alpha=1.0 | | C=1, penalty='l2' | | criterion='gini', n_estimators=100 | | C=1, kernel='linear' | |
| weighting method | tf | tf-idf | tf | tf-idf | tf | tf-idf | tf | tf-idf |
| F1 micro avg | 0.86 | 0.86 | 0.89 | 0.89 | 0.86 | 0.86 | 0.86 | 0.86 |
| F1 macro avg | 0.86 | 0.86 | 0.90 | 0.90 | 0.86 | 0.86 | 0.86 | 0.86 |
| F1 weighted avg | 0.86 | 0.86 | 0.89 | 0.89 | 0.86 | 0.86 | 0.86 | 0.86 |

The results observed in Table 4.15 and Table 4.16 form baseline for the Cumhuriyet and Sabah datasets. After this point, in the experiments with traditional classifiers; we proceeded with the LR classifier for which the best classification result for *tf* and *tf-idf* weighting are obtained, and the RF classifier, which is more advantageous than SVC in terms of computation time.

In Table 4.17, the classification performances of traditional classifiers on the Cumhuriyet dataset that is represented by using average, total, and variance of Word2vec vectors are presented. First of all, it is observed that the average and sums of vectors are more successful in the representation of texts than the variance, as it is observed in the sentiment analysis task. Based on this, if the results for average and sum are evaluated, the skip-gram architecture is more successful than the CBOW architecture. The highest classification success for 3 different vector sizes is achieved by skip-gram architecture and negative sampling algorithm. In

addition, the difference between embedding vector dimensions is not clearly visible in the sentiment analysis task as it is two-class dataset with short texts. However, it can be observed that this difference is clear in the document classification problem that has longer documents, and it can be said that using higher dimensions for the word vectors yields more successful classification results. The classification success is very similar for the Cumhuriyet dataset when using traditional BOW and Word2vec embedding methods to represent documents.

Table 4.17. Experimental results of traditional classifiers using Word2vec method for Cumhuriyet dataset

| | | LR | | | RF | | |
|-----|----------------------------------|-------------|-------------|------|------|------|------|
| | | avg | sum | var | avg | sum | var |
| 100 | CBOW – hierarchical softmax | 0.92 | 0.92 | 0.82 | 0.91 | 0.90 | 0.76 |
| | CBOW – negative sampling | 0.92 | 0.92 | 0.85 | 0.90 | 0.90 | 0.79 |
| | Skip-gram – hierarchical softmax | 0.92 | 0.92 | 0.86 | 0.91 | 0.90 | 0.86 |
| | Skip-gram – negative sampling | 0.92 | 0.92 | 0.86 | 0.91 | 0.90 | 0.85 |
| 200 | CBOW – hierarchical softmax | 0.93 | 0.92 | 0.86 | 0.91 | 0.90 | 0.76 |
| | CBOW – negative sampling | 0.93 | 0.92 | 0.88 | 0.90 | 0.90 | 0.80 |
| | Skip-gram – hierarchical softmax | 0.93 | 0.93 | 0.86 | 0.91 | 0.90 | 0.85 |
| | Skip-gram – negative sampling | 0.93 | 0.93 | 0.87 | 0.91 | 0.90 | 0.84 |
| 300 | CBOW – hierarchical softmax | 0.93 | 0.93 | 0.88 | 0.91 | 0.90 | 0.75 |
| | CBOW – negative sampling | 0.93 | 0.93 | 0.89 | 0.90 | 0.90 | 0.81 |
| | Skip-gram – hierarchical softmax | 0.93 | 0.93 | 0.86 | 0.91 | 0.90 | 0.83 |
| | Skip-gram – negative sampling | 0.93 | 0.93 | 0.88 | 0.91 | 0.90 | 0.83 |

The effect of the Word2vec vectors on the classification success for the Sabah dataset is summarized in Table 4.18. The highest classification performance for Sabah dataset is obtained as 0.89 with LR and 300-dimensional skip-gram vectors. Using higher dimensional Word2vec vectors slightly improves the classification success. Also, very similar success rates are achieved with the baseline results.

Table 4.18. Experimental results of traditional classifiers using Word2vec method for Sabah dataset

| | | LR | | | RF | | |
|-----|----------------------------------|-------------|------|------|------|------|------|
| | | avg | sum | var | avg | sum | var |
| 100 | CBOW – hierarchical softmax | 0.87 | 0.86 | 0.77 | 0.87 | 0.87 | 0.76 |
| | CBOW – negative sampling | 0.87 | 0.86 | 0.78 | 0.86 | 0.87 | 0.78 |
| | Skip-gram – hierarchical softmax | 0.88 | 0.87 | 0.79 | 0.87 | 0.87 | 0.80 |
| | Skip-gram – negative sampling | 0.88 | 0.87 | 0.77 | 0.88 | 0.87 | 0.78 |
| 200 | CBOW – hierarchical softmax | 0.88 | 0.87 | 0.79 | 0.87 | 0.87 | 0.77 |
| | CBOW – negative sampling | 0.88 | 0.87 | 0.81 | 0.87 | 0.87 | 0.78 |
| | Skip-gram – hierarchical softmax | 0.88 | 0.88 | 0.80 | 0.87 | 0.87 | 0.78 |
| | Skip-gram – negative sampling | 0.88 | 0.88 | 0.80 | 0.87 | 0.87 | 0.78 |
| 300 | CBOW – hierarchical softmax | 0.88 | 0.88 | 0.82 | 0.87 | 0.87 | 0.77 |
| | CBOW – negative sampling | 0.88 | 0.87 | 0.83 | 0.87 | 0.87 | 0.79 |
| | Skip-gram – hierarchical softmax | 0.89 | 0.88 | 0.81 | 0.87 | 0.87 | 0.77 |
| | Skip-gram – negative sampling | 0.89 | 0.88 | 0.82 | 0.87 | 0.87 | 0.78 |

In Table 4.19 and Table 4.20, performance analysis of Fasttext embedding method is presented. If Table 4.19 and Table 4.20 are evaluated together, Fasttext vectors also achieve similar classification success with Word2vec vectors. For both datasets, very similar classification performances with the baseline results are observed.

Table 4.19. Experimental results of traditional classifiers using Fasttext method for Cumhuriyet dataset

| | | LR | | | RF | | |
|-----|----------------------------------|-------------|-------------|------|------|------|------|
| | | avg | sum | var | avg | sum | var |
| 100 | CBOW – hierarchical softmax | 0.92 | 0.91 | 0.78 | 0.89 | 0.88 | 0.68 |
| | CBOW – negative sampling | 0.91 | 0.91 | 0.80 | 0.89 | 0.88 | 0.72 |
| | Skip-gram – hierarchical softmax | 0.92 | 0.92 | 0.86 | 0.91 | 0.90 | 0.86 |
| | Skip-gram – negative sampling | 0.92 | 0.92 | 0.87 | 0.91 | 0.90 | 0.85 |
| 200 | CBOW – hierarchical softmax | 0.92 | 0.92 | 0.84 | 0.89 | 0.88 | 0.70 |
| | CBOW – negative sampling | 0.92 | 0.92 | 0.86 | 0.89 | 0.88 | 0.74 |
| | Skip-gram – hierarchical softmax | 0.93 | 0.93 | 0.87 | 0.91 | 0.90 | 0.86 |
| | Skip-gram – negative sampling | 0.93 | 0.93 | 0.87 | 0.91 | 0.90 | 0.85 |
| 300 | CBOW – hierarchical softmax | 0.93 | 0.92 | 0.86 | 0.89 | 0.88 | 0.69 |
| | CBOW – negative sampling | 0.93 | 0.92 | 0.88 | 0.89 | 0.88 | 0.75 |
| | Skip-gram – hierarchical softmax | 0.93 | 0.93 | 0.87 | 0.91 | 0.90 | 0.86 |
| | Skip-gram – negative sampling | 0.93 | 0.93 | 0.89 | 0.92 | 0.90 | 0.85 |

Table 4.20. Experimental results of traditional classifiers using Fasttext method for Sabah dataset

| | | LR | | | RF | | |
|-----|----------------------------------|-------------|------|------|------|------|------|
| | | avg | sum | var | avg | sum | var |
| 100 | CBOW – hierarchical softmax | 0.86 | 0.86 | 0.73 | 0.86 | 0.85 | 0.70 |
| | CBOW – negative sampling | 0.86 | 0.85 | 0.73 | 0.86 | 0.86 | 0.72 |
| | Skip-gram – hierarchical softmax | 0.88 | 0.87 | 0.79 | 0.87 | 0.87 | 0.80 |
| | Skip-gram – negative sampling | 0.88 | 0.87 | 0.77 | 0.88 | 0.87 | 0.78 |
| 200 | CBOW – hierarchical softmax | 0.87 | 0.86 | 0.77 | 0.86 | 0.86 | 0.71 |
| | CBOW – negative sampling | 0.87 | 0.86 | 0.79 | 0.86 | 0.86 | 0.73 |
| | Skip-gram – hierarchical softmax | 0.88 | 0.88 | 0.82 | 0.87 | 0.87 | 0.80 |
| | Skip-gram – negative sampling | 0.88 | 0.88 | 0.80 | 0.87 | 0.87 | 0.79 |
| 300 | CBOW – hierarchical softmax | 0.88 | 0.87 | 0.79 | 0.86 | 0.85 | 0.70 |
| | CBOW – negative sampling | 0.87 | 0.87 | 0.80 | 0.86 | 0.86 | 0.73 |
| | Skip-gram – hierarchical softmax | 0.89 | 0.88 | 0.82 | 0.87 | 0.87 | 0.80 |
| | Skip-gram – negative sampling | 0.89 | 0.88 | 0.82 | 0.87 | 0.87 | 0.79 |

The experimental results of the Doc2vec vectors that trained as a result of 10 iterations on 640.585 documents belonging to 13 class are given in Table 4.21 and Table 4.22. Considering both of these tables, DBOW architecture can be said to be more successful than DM architecture for long texts such as news.

If the classification performances for the datasets are evaluated, the baseline performance for both datasets is reached when embedding methods are used to represent documents. However, only for Sabah dataset, the performance of the Doc2vec method is below the baseline.

Table 4.21. Experimental results of traditional classifiers using Doc2vec method for Cumhuriyet dataset

| | | LR | RF |
|-----|-----------------------------|-------------|------|
| 100 | DBOW – hierarchical softmax | 0.92 | 0.89 |
| | DBOW – negative sampling | 0.93 | 0.91 |
| | DM – hierarchical softmax | 0.86 | 0.79 |
| | DM – negative sampling | 0.74 | 0.73 |
| 200 | DBOW – hierarchical softmax | 0.93 | 0.85 |
| | DBOW – negative sampling | 0.93 | 0.90 |
| | DM – hierarchical softmax | 0.86 | 0.76 |
| | DM – negative sampling | 0.75 | 0.69 |
| 300 | DBOW – hierarchical softmax | 0.93 | 0.83 |
| | DBOW – negative sampling | 0.93 | 0.88 |
| | DM – hierarchical softmax | 0.86 | 0.72 |
| | DM – negative sampling | 0.76 | 0.67 |

Table 4.22. Experimental results of traditional classifiers using Doc2vec method for Sabah dataset

| | | LR | RF |
|-----|-----------------------------|-------------|------|
| 100 | DBOW – hierarchical softmax | 0.86 | 0.85 |
| | DBOW – negative sampling | 0.85 | 0.85 |
| | DM – hierarchical softmax | 0.69 | 0.66 |
| | DM – negative sampling | 0.57 | 0.67 |
| 200 | DBOW – hierarchical softmax | 0.86 | 0.82 |
| | DBOW – negative sampling | 0.86 | 0.85 |
| | DM – hierarchical softmax | 0.68 | 0.62 |
| | DM – negative sampling | 0.62 | 0.66 |
| 300 | DBOW – hierarchical softmax | 0.87 | 0.81 |
| | DBOW – negative sampling | 0.86 | 0.84 |
| | DM – hierarchical softmax | 0.67 | 0.58 |
| | DM – negative sampling | 0.64 | 0.65 |

In all the experiments performed in this step, Python version 3.6.7 programming language, Scikit-learn version 0.20.3 machine learning library, Numpy version 1.16.3 library for mathematical operations, and Pandas version 0.24.2 library for data analysis and data processing, are used.

4.4.2. Performance of Deep Learning based Classifiers

In this section, Amazon Web Services are used for our experiments. Because the GPU is needed to train the models, the g3.4xlarge instance of AWS's recommended GPU instances is used. Hardware features of this instance are 1 NVIDIA Tesla M60 GPU with 8Gb memory, 122 Gb memory, and 16 vCPU. AWS Deep Learning AMI (Ubuntu)-version 23.0 is installed on this instance for the working environment needed. The tensorflow_p36 environment on this AMI is used for experiments. The operating system, libraries, and NVIDIA driver version on AMI are Ubuntu 16.04, Tensorflow 1.13.1, Keras 2.2.4, and NVIDIA Driver 418.40.04. Figure 4.5 contains a screenshot of the GPU usage of the virtual machine used.

Experiments are done without any problems for the Cumhuriyet dataset with a total of 97,325 samples including 72,973 train and 24,325 test data instances. However, the Sabah dataset includes 314,987 train and 104,996 test instances, and totally 419,983 samples. For this reason, some memory problems occur when applying experiments for the Sabah dataset. Therefore, a subset of the documents in the Sabah dataset is generated by randomly chosen 25% of the documents in the Sabah dataset by taking the class distribution into account, and the deep learning-based experiments are applied on this subset which is called 25% Sabah dataset. In order to make a comparison, this subset of the dataset is represented by *tf* and *tf-idf* weighting and the baseline experiments are repeated for this subset.

The class distributions of the 25% Sabah dataset for the train and test sets are shown in Table 4.23. The baseline results obtained by using *tf* and *tf-idf* on this dataset can be seen in Table 4.24.


```

ubuntu@ip-172-31-35-234:~$ nvidia-smi
Mon May 20 08:42:36 2019

+-----+
| NVIDIA-SMI 418.40.04      Driver Version: 418.40.04      CUDA Version: 10.1      |
+-----+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0   Tesla M60             On          | 00000000:00:1E:0 Off |            0         |
| N/A   55C    P0     100W / 150W |  7455MiB /  7618MiB |   100%    Default   |
+-----+-----+

Processes:
GPU      PID    Type   Process name                      GPU Memory
Usage
+-----+-----+
|  0      2475    C     ..naconda3/envs/tensorflow_p36/bin/python  7444MiB
+-----+-----+

```

Figure 4.5. Screenshot for GPU usage on AWS

Table 4.23. Class distribution of 25% of Sabah dataset

| class | Number of samples in train data | Number of samples in test data | total |
|----------|------------------------------------|-----------------------------------|--------|
| yazarlar | 12769 | 4256 | 17025 |
| ekonomi | 16026 | 5342 | 21368 |
| yaşam | 22996 | 7666 | 30662 |
| gündem | 26956 | 8985 | 35941 |
| total | 78747 | 26249 | 104996 |

Table 4.24. Experimental results of traditional classifiers on 25% of Sabah dataset using BOW method

| | Naïve Bayes Multinomial | | Logistic Regression | | Random Forest | | SVC | |
|-----------------------|----------------------------|--------|------------------------|-------------|---------------------------------------|--------|-------------------------|--------|
| parameters | alpha=1.0 | | C=1, penalty='l2' | | criterion='gini', n_estimators=100 | | C=1, kernel='linear' | |
| weighting method | tf | tf-idf | tf | tf-idf | tf | tf-idf | tf | tf-idf |
| F1 micro avg | 0.85 | 0.86 | 0.88 | 0.88 | 0.84 | 0.84 | 0.85 | 0.86 |
| F1 macro avg | 0.86 | 0.86 | 0.88 | 0.88 | 0.84 | 0.84 | 0.86 | 0.86 |
| F1 weighted avg | 0.85 | 0.86 | 0.88 | 0.88 | 0.84 | 0.84 | 0.85 | 0.86 |

Table 4.25 and Table 4.26 summarize the experimental results performed using the Word2vec and Fasttext vectors that are trained with the 300-dimensional

skip-gram architecture and the negative sampling algorithm, which are the best results in our previous experiments for the Cumhuriyet and Sabah datasets. If Table 4.25 is evaluated for the Cumhuriyet dataset, the best result obtained for all architectures is 0.93. *Micro-average F-1* score, which is equal to the baseline, but not higher. However, if Table 4.26 is examined, we have slightly higher classification successes than the baseline performance that are obtained as 0.88 in all network architectures for Sabah dataset.

Table 4.25. Experimental results of deep learning-based classifiers on Cumhuriyet dataset

| Network architecture | Word2vec | Fasttext |
|----------------------|-------------|-------------|
| CNN3 | 0.93 | 0.92 |
| LSTM | 0.93 | 0.93 |
| CNN-LSTM | 0.93 | 0.93 |
| CNN-LSTM2 | 0.93 | 0.93 |

Table 4.26. Experimental results of deep learning-based classifiers on 25% Sabah dataset

| Network architecture | Word2vec | Fasttext |
|----------------------|-------------|-------------|
| CNN3 | 0.89 | 0.89 |
| LSTM | 0.90 | 0.89 |
| CNN-LSTM | 0.89 | 0.89 |
| CNN-LSTM2 | 0.89 | 0.90 |

For all the artificial neural network architectures that are created, the highest *F-1* scores is achieved ad 0.93 for the Cumhuriyet data. This value is equal to the baseline result. For the Sabah dataset, all the results achieved with the deep learning-based classification methods are slightly over the baseline which is equal to 0.88. It is observed that the successes of Word2vec and Fasttext vectors are approximately the same when Table 4.25 and Table 4.26 are evaluated together.

4.5. Comparison of the Used Methods

Document representations for short and long texts are obtained by using Word2vec, Fasttext, Doc2vec and Glove embedding methods, and traditional *tf* and *tf-idf* weighting methods. Similar classification scores are observed with embedding methods which were learned from 2 different size datasets and traditional weighting methods, for the sentiment analysis problem. The only exception in these experiments was to achieve a score of 14% above the baseline result by employing Doc2vec vectors trained using class information over the dataset itself. This can be explained by the fact that the number of samples of the trained dataset is relatively small and contains only two classes. For the problem of document classification, news documents that consist of longer texts are worked on. The same classification success is achieved as the embedding vectors trained for this problem and the traditional weighting methods.

According to these results, it is necessary to make a choice between embedding methods and traditional weighting methods; embedding methods may be preferred if embedding vectors are learned on a dataset obtained from the same domain as the data to be studied. If embedding vectors are not available and need to be learned; it is necessary to consider the cost of hardware and time, because it can take long hours for iterations to train on large datasets.

According to these results, it is necessary to make a choice between embedding methods and traditional weighting methods; embedding methods may be preferred if embedding vectors previously learned on a dataset from the same domain with the data to be studied are available. But if embedding vectors are not available and need to be learned; in this case, it is necessary to consider the cost of hardware and time, because it can take long hours for iterations to train on large datasets.

If a comparison of traditional machine learning methods with deep learning-based methods is made; by using the LSTM for sentiment analysis problem, 2% higher classification success is achieved. In other architectures used,

results achieved are similar to traditional methods. The same success is achieved as the traditional methods for the 9-class Cumhuriyet dataset with the deep learning-based methods used for document classification problem. Deep learning-based methods achieved 1-2% higher success for 4-class Sabah dataset.

When training these network structures, the embedding vectors that have previously trained are used. Considering the successes obtained by using the same vectors for traditional classifiers; it is observed that they have the same success with deep learning-based classifiers.

If an assessment is made, it can be said that deep learning-based classifiers are successful. However, this success depends not only on these classifiers but also on the representation methods used to train these network structures. However, for training deep learning-based methods, a large number of hyperparameters need to be adjusted compared to the traditional methods. There are no fixed hyperparameters specific to the problem. Therefore, the network should be observed during the training and the parameters should be adjusted according to these observations. In addition, training deep learning-based methods requires larger sized datasets. This increases the cost of hardware and time.

4.6. Comparison of Results with Studies using the Same Datasets

4.6.1. Comparison of Results for Sentiment Analysis

In this section, results of this thesis are compared with the studies using the same dataset with our study for sentiment classification task. In this thesis the data set which is shared by Hayran and Sert (2017) is used.

This dataset has been first used by Hayran and Sert (2017) who applied some preprocessing steps that are i) restricting the number of consecutive repeating letters to 2, ii) converting all letters to lowercase, iii) removing all face expressions, iv) removing usernames, links, punctuation marks, and single-character words. After this step, they obtain 16000 positive and 16000 negative tweets.

In the study of Hayran and Sert (2017) for the Word2vec word embedding method; the *min_count* parameter is taken as 2, 100-dimensional word vectors are trained by using CBOW and skip-gram architectures. They represent documents by using average, sum, variance, average-sum, average-variance, sum-variance and variance-average-sum of the Word2vec vectors. By using sum values, skip-gram and CBOW architectures are compared, and it is observed that skip-gram achieves higher classification success (78.27%). In this step, they applied SVM classifier with linear kernel and 2-fold cross-validation on the entire data. Then, the SVM classifier is applied to the texts represented by vectors trained by using skip-gram architecture by taking *min_count* parameters as 1 and 2. In this step, when *min_count* is equal to 1, higher classification success (78.27%) is achieved. In the last step of the study (Hayran and Sert, 2017); vectors are trained using *min_count*=1 and skip-gram architecture. Document representations are obtained from these vectors by the methods mentioned earlier. The SVM classifier is applied with 5-fold cross-validation over the whole data set.

According to Hayran and Sert (2017) the classification successes obtained as a result of the experiments are as follows; 78.31% for sum representation, 78.34% for average representation, 64.02% for variance representation, and 80.05% by using these 3 representation methods together. For all experiments in Hayran and Sert (2017) accuracy is chosen as evaluation metric.

In this thesis; first of all, the preprocessing steps are applied. As preprocessing we also restrict the number of consecutive repeating letters to 2, apply lowercase conversion, and eliminate all characters except the letters. At the end of these steps, 15369 positive and 15881 negative tweets are obtained.

During the experiments in this thesis; Word2vec, Doc2vec, Fasttext, and Glove embedding vectors are trained in 100 and 300 dimensions. During the training of these vectors, *min_count* is set to 1, as done in the previous study (Hayran and Sert, 2017). The window size parameter, which is not specified in Hayran and Sert (2017), is taken as 5. For the training of these embedding vectors,

the following parameters are not specified in the study (Hayran and Sert, 2017) are used; CBOW and skip-gram architectures are trained for 10 iterations by taking negative parameter as 10 with both hierarchical softmax and negative sampling algorithms. Both TSD and 20M tweet datasets are used during the training in our study. Instead of applying a classifier on the entire dataset; 70% of randomly selected data is used as training data and the remaining 30% is used as test data. The documents are represented by both bag-of-words model with *tf* and *tf-idf* weighting, and the embedding vectors. The documents represented by using average, sum and variance values of embedding vectors are classified by applying traditional classifiers and deep learning-based classifiers. NBM, RF, LR, and SVM classifiers are used as traditional classifiers and CNN, LSTM and two different combinations of CNN and LSTM are used as deep learning-based classifiers.

As a result of all our experiments during this thesis, the highest classification success is achieved as 75% by using *tf* and *tf-idf* weighting. As a result of experiments conducted with embedding vectors, the best classification success is achieved by using DBOW architecture of 100 and 300-dimensional Doc2vec vectors trained using hierarchical softmax algorithm and LR classifier as 89%. According to the results of our experiments; the highest classification success for the Word2vec embedding method is 77%. This score is reached with the 300-dimensional vectors trained by using skip-gram architecture and hierarchical softmax algorithm on the TSD, and an LSTM architecture as the classifier. The highest classification accuracy is 75% for the Fasttext embedding method, and this success is reached with 100 and 300-dimensional vectors trained by skip-gram and negative sampling on 20M tweets and RF and SVM classifiers. Finally, the highest classification success for Glove embedding is achieved with RF classifier and 300-dimensional vectors as 71%. *Micro average F-1* score values is used when comparing all classifiers' success in our study.

4.6.2. Comparison of Results for Document Classification

In this thesis, SuDer dataset shared by Şen and Yanıkoğlu (2018) is used for document classification problem, and our results are compared with the other studies that use the same dataset.

First, study of Şen and Yanıkoğlu (2018) that developed the dataset is examined; in the Cumhuriyet dataset, there are 14 classes including photos and videos; in the Sabah dataset, there are 4 classes. In the preprocessing phase; suffixes separated by apostrophes, single letter words and numbers, and stop-words are eliminated, lowercase conversion is applied. Also, stemming is done by using the Zemberek library in their study.

During the experiments of Şen and Yanıkoğlu (2018); the documents are represented by *tf-idf* weighting and the average of Word2vec embedding vectors. Word2vec vectors are trained by using the following parameters; skip-gram architecture, negative sampling algorithm, *window_size* = 20, *negative* = 5, *vector_size* = {100, 200, 400, 600}, *min_count* = 20, and *iteration* = 20. For the document representation, the most common words are used for *tf-idf* weighting, and the frequency values are taken as {1000, 5000, 10000, 20000, 50000}. For classification step, SVM with linear kernel, SDA and an artificial neural network architecture are used. The artificial neural network architecture consists of two hidden layers containing 50 neurons and ReLU activation function. Also, RMSprop is applied as the optimization algorithm and 0.01 as the learning rate.

The documents represented by using *tf-idf* weighting method are classified with SDA in Şen and Yanıkoğlu (2018). The highest classification accuracy for this experiment are 72.08% for the Sabah dataset and 47.94% for the Cumhuriyet dataset in their study. Highest classification accuracies with word vector representations and SVM are 86.89% for Sabah and 72.50% for Cumhuriyet. Highest classification accuracies with word vector representations and the artificial neural network architecture they described are 88.28% for Sabah and 74.31% for

Cumhuriyet datasets. Accuracy is used in Şen and Yanıkoğlu (2018) as evaluation metric.

In this thesis; for the document classification problem, experiments are conducted on 9 classes of Cumhuriyet dataset and all of Sabah dataset. From the Cumhuriyet dataset; video and photo classes with a small number of words, and classes they shared with Sabah dataset are eliminated. In the preprocessing step; characters other than letters and numbers are eliminated, the suffixes after the apostrophe are eliminated when a word is separated by an apostrophe from its suffix, and lowercase conversion is applied.

In our study, for document representation; *tf*, *tf-idf* weighting methods, and average, sum and variance values of Word2vec, Doc2vec, and Fasttext vectors learned from all dataset containing more than 600,000 documents in total are used. We train embedding vectors for all possible combinations of architectures and algorithms, and for vector sizes with 100, 200, and 300.

Experiments in this thesis are performed separately for two data sets. In documents represented by *tf* and *tf-idf* weighting and classified by traditional classifiers; the highest success for the Cumhuriyet dataset is 93%, and while it is 89% for the Sabah dataset. Highest classification accuracies with Word2vec embedding vectors and traditional classifiers are %93 as Cumhuriyet and %89 for Sabah datasets. When Fasttext embedding vectors and traditional classifiers are used, the highest classification accuracies are 93% for the Cumhuriyet and 89% for the Sabah datasets. Finally, with Doc2vec embedding and traditional classifiers, the classification accuracy is 93% for the Cumhuriyet and 87% for the Sabah datasets.

In our experiments with deep learning-based CNN, LSTM and combination of CNN and LSTM architectures, the entire Cumhuriyet dataset is used, but 25% of the Sabah dataset could be used because of the memory problems. For all the deep learning-based classifiers applied, the achievements are 93% for the Cumhuriyet and 90% for the Sabah datasets. All these classification success values are *micro average F-1* scores for each one of our experiments.



5. CONCLUSIONS

In this study, neural network-based word and document representation methods, deep neural network-based classifiers are used, and a comparison is made with traditional document representation and classification methods which have been frequently used for Turkish text classification problems. While doing this comparison both tweets that are short texts and frequently contain spelling errors, and news documents that usually consist of longer texts and rarely have spelling errors are used. Binary sentiment classification is studied on tweets, and multi-class document classification is studied on news document datasets.

When we evaluate our findings for binary sentiment classification, it can be concluded that:

- On the texts that are represented by using *tf* and *tf-idf* weighting, the highest classification success is obtained as 0.75 with the LR classifier.
- In our experiments for sentiment analysis with traditional classifiers when embedding methods are applied to represent texts by using 100 and 300 dimensional vectors that are trained over 10 iterations on two different sized datasets, the highest classification accuracies are observed as 0.74, 0.75, 0.89, and 0.72, for Word2vec, Fasttext, Doc2vec, and Glove, respectively.
- For deep learning-based models trained by using Word2vec, Fasttext, and Glove document representation methods, the highest classification success is achieved as 0.77 with a single LSTM layer and by using Word2vec vectors trained on TSD. The next highest success is 0.74 which is achieved with architecture that has a convolution layer and a subsequent LSTM layer using Fasttext vectors trained on 20M tweets.

When we evaluate the results obtained for the document classification task, the following conclusions are reached;

- For Cumhuriyet and Sabah newspaper datasets, the highest classification accuracies for the traditional classifiers are observed as 0.93 and 0.89, respectively, when BOW model with *tf* and *tf-idf* weighting is used to represent documents.
- We have the classification successes as 0.93 and 0.89 for the two datasets respectively, by using 100, 200 and 300 dimensional Word2vec, Doc2vec, and Fasttext vectors that are trained on more than 600,000 documents consist of the two datasets.
- The achievements of the vectors trained do not pass the performance of the traditional *tf*, *tf-idf* weighting methods.
- In our experiments for deep learning-based classifiers, nearly the same classification successes with the baseline results are observed for the Cumhuriyet dataset. Only for the Sabah dataset, better classification success is achieved for deep learning methods with respect to the baseline results that are 0.90 vs. 0.88.

According to the experimental results, it is found that neural network-based text representation methods for Turkish texts have similar results to the traditional methods. If there is publicly available previously learned word or document vectors for Turkish which have been trained on a large dataset from the same domain with the problem to be solved, using these embedding vectors could be preferred instead of *tf* and *tf-idf* weighting. In our experiments, Doc2vec embedding method has higher classification accuracy for some cases than the *tf* and *tf-idf* weighting methods. Deep learning-based classifiers have closer or slightly higher classification successes with the use of Word2vec, Fasttext and Glove embedding vectors with respect to the traditional classifiers. Therefore, deep learning-based classifiers can be seen as an alternative to the traditional methods used for Turkish texts.

REFERENCES

- Aghababaei, S. and Makrehchi, M. 2016. Mining Social Media Content for Crime Prediction. 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Omaha, NE, 526-531.
- Amasyalı, M.F., Tasköprü, H., and Çalışkan, K., 2018. Words, Meanings, Characters in Sentiment Analysis. Innovations in Intelligent Systems and Applications Conference (ASYU), Adana, 1-6.
- Ay Karakuş, B., Talo, M., Hallaç, İ. R., and Aydın, G., 2018. Evaluating deep learning models for sentiment classification. *Concurrency and Computation: Practice and Experience*, 30(21): e4783.
- Ayata, D., Saraçlar, M., and Özgür, A., 2017. Turkish tweet sentiment analysis with word embedding and machine learning. 25th Signal Processing and Communications Applications Conference (SIU), Antalya, 1-4.
- Bilgin, M., and Şentürk, İ. F., 2017. Sentiment analysis on Twitter data with semi-supervised Doc2Vec. International Conference on Computer Science and Engineering (UBMK), Antalya, 661-666.
- Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T., 2017. Enriching Word Vectors with Subword Information, *Transactions of the Association for Computational Linguistics*, 5: 135-146.
- Breiman, L., 2001. Random Forests. *Machine Learning*, 45(1):5-32.
- Chang, C., and Lin, C. 2001. LIBSVM: a Library for Support Vector Machines (Version 2.31).
- Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. ICML.
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P.P., 2011. Natural Language processing (almost) from Scratch, *Journal of Machine Learning Research*, 12: 2493-2537.

- Cutler, A., Cutler, D.R., and Stevens., J.R. 2012. Random Forests. In Ensemble Machine Learning. (Edited by Zhang C, and Ma Y.). Springer, US, 157–175.
- Çelenli, H. İ., 2018. Application of paragraph vectors to news and tweet data. 26th Signal Processing and Communications Applications Conference (SIU), Izmir, 1-4.
- Çelenli, H. İ., Öztürk, S. T., Şahin, G., Gerek, A., and Ganiz, M. C., 2018. Document Embedding Based on Supervised Methods for Turkish Text Classification. 3rd International Conference on Computer Science and Engineering (UBMK), Sarajevo, 477-482.
- Çoban, Ö., and Karabey, I., 2017. Music genre classification with word and document vectors. 25th Signal Processing and Communications Applications Conference (SIU), Antalya, 1-4.
- Davcheva, P. 2014. Identifying Sports Talents by Social Media Mining as a Marketing Instrument. 2014 Annual SRII Global Conference, San Jose, CA, 223-227.
- Elman, J.L. 1990. Finding Structure in Time. *Cognitive Science* 14:179-211.
- Goodfellow, I., Bengio, Y., and Courville, A. 2016. *Deep Learning*. MIT Press, 800.
- Hassan, A., and Mahmood, A., 2017. Deep Learning Approach for Sentiment Analysis of Short Texts, 3rd International Conference on Control, Automation and Robotics (ICCAR), 705-710.
- Hayran, A., and Sert, M., 2017. Sentiment analysis on microblog data based on word embedding and fusion techniques. 25th Signal Processing and Communications Applications Conference (SIU), Antalya, 1-4.
- Hochreiter, S., and Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation*, 9: 1735-1780.

- Hu, Z., Hu, J., Ding, W., and Zheng, X., 2015. Review Sentiment Analysis Based on Deep Learning, IEEE 12th International Conference on e-Business Engineering, Beijing, 87-94.
- Huang, Q., Chen, R., Zheng, X., and Dong, Z., 2017. Deep Sentiment Representation Based on CNN and LSTM, International Conference on Green Informatics (ICGI), 30-33.
- Jurafsky, D., and Martin, J.H., 2018. Speech and Language Processing (3rd edition draft), 558.
- Kalchbrenner, N., Grefenstette, E., and Blunsom, P., 2014. A Convolutional Neural Network for Modelling Sentences, ACL.
- Kim, Y., 2014. Convolutional Neural Networks for Sentence Classification, EMNLP.
- Le, Q.V., and Mikolov, T., 2014. Distributed Representations of Sentences and Documents, ICML.
- LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., & Jackel, L.D., 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1:541-551.
- Lee, K., Agrawal, A. and Choudhary, A. 2015. Mining social media streams to improve public health allergy surveillance. 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Paris, 815-822.
- McCallum, A., and Nigam, K., 1998. A Comparison of Event Models for Naive Bayes Text Classification.
- Mikolov, T., Chen, K., Corrado, G.S., and Dean, J., 2013a. Efficient Estimation of Word Representations in Vector Space, CoRR, abs/1301.3781.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., and Dean, J., 2013b. Distributed Representations of Words and Phrases and their Compositionality, NIPS.

- Olivera, G., Zita, B., and Sasa, B. 2013. Students' behavior on social media sites — A data mining approach. 2013 IEEE 11th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, 347-352.
- Onorati, T., Díaz, P., and Recio, B. C. 2016. Tweet Me and I'll Help You: Mapping Tweets for Emergency Operation Centers: The Case of Paris Attacks. 2016 International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, 31-34.
- Pennington, J., Socher, R., and Manning, C.D., 2014. Glove: Global Vectors for Word Representation, EMNLP.
- Severyn, A., and Moschitti, A., 2015. Twitter Sentiment Analysis with Deep Convolutional Neural Networks, SIGIR.
- Seyfioğlu, M. S., and Demirezen, M. U., 2017. A hierarchical approach for sentiment analysis and categorization of Turkish written customer relationship management data. Federated Conference on Computer Science and Information Systems (FedCSIS), Prague, 361-365.
- Şahin, G., 2017. Turkish document classification based on Word2Vec and SVM classifier. 25th Signal Processing and Communications Applications Conference (SIU), Antalya, 1-4.
- Şen, M. U., and Erdoğan, H., 2014. Learning word representations for Turkish. 22nd Signal Processing and Communications Applications Conference (SIU), Trabzon, 1742-1745.
- Şen, M. U., and Yanıkoğlu, B., 2018. Document classification of SuDer Turkish news corpora. 26th Signal Processing and Communications Applications Conference (SIU), Izmir, 1-4.
- Vateekul, P., and Koomsubha, T., 2016. A study of sentiment analysis using deep learning techniques on Thai Twitter data, 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, 1-6.

- Vidhya., K., and Aghila, G., 2010. Mining Process, Techniques and Tools: An Overview. *International Journal of Information Technology and Knowledge Management*, 2(2):613-622.
- Vo, Q., Nguyen, H., Le, B., and Nguyen, M., 2017. Multi-channel LSTM-CNN model for Vietnamese sentiment analysis, 9th International Conference on Knowledge and Systems Engineering (KSE), Hue, 24-29.
- Yang, S., and Xia, Z., 2016. A convolutional neural network method for Chinese document sentiment analyzing. 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 308-312.
- Yıldırım, S., and Yıldız, T., 2018a. A comparative analysis of text classification for Turkish language, *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 24(5):879-886.
- Yıldırım, S., and Yıldız, T., 2018b. A Comparison of Different Approaches to Document Representation in Turkish Language. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 22(2):569-576.
- Young, T., Hazarika, D., Poria, S., and Cambria, E. 2018. Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine*, 13:55-75.
- Zhang, L., Wang, S., and Liu, B., 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 8.
- Zhang, Y., and Wallace, B.C., 2017. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *IJCNLP*.



BIOGRAPHY

Funda Güven was born on December 2th, 1992 in Mersin. She was completed her elementary education at İbrahim Karaođlanođlu Primary School and secondary education at MTSO Anatolian High School. She has completed her university education at department of Computer Engineering of ukurova University in 2016.





APPENDIX



APPENDIX A.

Table A. 1. Literature review for English and other languages

| Writers | Dataset | Subject of Study | Text Representation Methods | Classification Methods |
|-----------------------------|---|--|------------------------------------|---|
| Kalchbrenner et al., 2014 | Stanford Sentiment Treebank, TREC, Twitter data | Binary and multi-class sentiment prediction, Question classification, Sentiment prediction | | DCNN |
| Kim, 2014 | MR, SST-1, SST-2, Subj, TREC, CR | Sentiment analysis, Question classification | Word2vec | CNN, CNN-rand, CNN-non-static, CNN-multichannel |
| Severyn and Moschitti, 2015 | Semeval-2015 | Phrase-level and message-level sentiment analysis | Word2vec | Proposed method |

| Writers | Dataset | Subject of Study | Text Representation Methods | Classification Methods |
|--------------------------|---|-----------------------------------|---|--|
| Hu et al., 2015 | electronic product reviews - Amazon, film reviews - Amazon and IMDB, hotel comments - TripAdvisor | Document-level sentiment analysis | Word frequency, contextual windows, POS tagging | HDNN, NB, SVM |
| Hassan and Mahmood, 2017 | Stanford Sentiment Treebank, IMDB | Sentiment analysis | Word2vec | Proposed method (ConvLstm) |
| Yang and Xia, 2016 | Chinese hotel comment corpus | Sentiment analysis | Word2vec | CNN, SVM, NBM |
| Huang et al., 2017 | Chinese Micro-blog data | Sentiment analysis | Word2vec | CNN, LSTM, CNN-LSTM (each single layer), proposed method - CNN-LSTM, SVM |

| Writers | Dataset | Subject of Study | Text Representation Methods | Classification Methods |
|------------------------------|--------------------------------------|-------------------------|------------------------------------|--|
| Vateekul and Koomsubha, 2016 | Thai Twitter data | Sentiment analysis | Word2vec | LSTM, DCNN |
| Vo et al., 2017 | VS and VLSP (Vietnamese text corpus) | Sentiment analysis | | SVM, CNN, LSTM, proposed method (multi-channel LSTM-CNN) |

APPENDIX B.

Table B. 1. Literature review for Turkish

| Writers | Dataset | Subject of Study | Text Representation Methods | Classification Methods |
|-------------------------|--|----------------------------|-----------------------------|------------------------|
| Şen and Erdoğan, 2014 | Wikipedia, Boğaziçi | Comparing word embedding | Word2vec | |
| Şahin, 2017 | 22729 Turkish documents of 7 different classes | Document classification | BOW, Word2vec | SVM |
| Çoban and Karabey, 2017 | 1250 lyrics of 5 classes | Music genre classification | BOW, Word2vec, Doc2vec | SVM |
| Hayran and Sert, 2017 | 16000 positive and 16000 negative tweets | Sentiment analysis | Word2vec | SVM |
| Ayata et al., 2018 | 5808 tweets of 4 classes and 158.885 tweets (20M tweets) | Sentiment analysis | Word2vec | SVM, RF |
| Çelenli, 2018 | 1150 news, 3000 tweets, 20M tweets | Document classification | BOW, Doc2vec | NBM, SVM, KNN, NC |

| Writers | Dataset | Subject of Study | Text Representation Methods | Classification Methods |
|-------------------------------|--|---|-----------------------------|------------------------|
| Şen and Yanıkoğlu, 2018 | 420.513 documents of 4 classes and 268.784 documents of 14 classes | Document classification | BOW (tf-idf), Word2vec | SVM, LDA, NN |
| Amasyalı et al., 2018 | 17389 tweets of 3 classes | Sentiment analysis | BOW, Fasttext | SVM, RF, CNN, LSTM |
| Bilgin and Şentürk, 2017 | 5187 Turkish tweets of 3 classes and 60591 English tweets of 3 classes | Sentiment analysis | Doc2vec | Linear Regression |
| Seyfioğlu and Demirezen, 2017 | 1071 labeled (406 positive, 664 negative) and 14000 unlabeled review | Sentiment analysis, document classification | Word2vec, Doc2vec, BOW | Xgboost |
| Çelenli et al., 2018 | 20M tweets, 1150 news, Hurriyet6c1k, Bilcol | Document classification | Doc2vec, BOW | SVM, KNN, CC, CFSVM |

| Writers | Dataset | Subject of Study | Text Representation Methods | Classification Methods |
|----------------------------|--|---|---|---------------------------------------|
| Ay Karakuş et al., 2018 | 44.617 movie reviews | Sentiment analysis | Word2vec | MLP, CNN, LSTM, BiLSTM, CNN-LSTM |
| Yıldırım and Yıldız, 2018a | TTC-3600 and T-4900 | Text classification | BOW, Doc2vec | NB, SVM, KNN, DT, ANN |
| Yıldırım and Yıldız, 2018b | 4900 documents of 7 classes | Text classification | BOW, Doc2vec, Word2vec, Glove | LR, SVM, KNN, CNN, LSTM, GRU, RNN |
| Güven (in this thesis) | Turkish Sentiment Dataset, 20M tweets, SuDer | Sentiment analysis, document classification | BOW, Word2vec, Doc2vec, Fasttext, Glove | NBM, SVC, RF, LR, CNN, LSTM, CNN-LSTM |