ANKARA YILDIRIM BEYAZIT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SENTIMENT ANALYSIS of TURKISH FINANCIAL TWEETS USING DEEP LEARNING MODELS for BIST 100 INDEX

Ph.D. Thesis by

Erkut MEMİŞ

Department of Computer Engineering

January, 2024

ANKARA

SENTIMENT ANALYSIS of TURKISH FINANCIAL TWEETS USING DEEP LEARNING MODELS for BIST 100 INDEX

A Thesis Submitted to the

Graduate School of Natural And Applied Sciences of

Ankara Yildirim Beyazit University

In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

in Computer Engineering, Department of Computer Engineering

by

Erkut MEMİŞ

January, 2024

ANKARA

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**SENTIMENT ANALYSIS of TURKISH FINANCIAL TWEETS USING DEEP LEARNING MODELS for BIST 100 INDEX**" completed by **ERKUT MEMİŞ** under supervision of **ASST. PROF. DR. MUSTAFA YENİAD** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Asst. Prof. Dr. Mustafa YENİAD

Supervisor

Assoc. Prof. Dr. Baha ŞEN

Thesis Committee Member

Assoc. Prof. Dr. Ömer KARAL

Thesis Committee Member

Assoc. Prof. Dr. Fecir DURAN

Examining Committee Member

Assoc. Prof. Dr. Cevat RAHEBİ

Examining Committee Member

Prof. Dr. Sadettin ORHAN

Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All information, documents and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

Date: 2024, 25 January **Signature:** _____

Name & Surname: Erkut MEMİŞ

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Asst. Prof. Dr. Mustafa YENİAD for continuous support of my studies.

Beside my advisor, I am deeply indebted to Asst. Prof. Dr. Hilal AKARKAMÇI and Prof. Dr. Fatih V. ÇELEBİ for their contributions and advices.

I also thank my dear wife Nilgün, my lovely children Defne, Duru, and N. Doruk for their supports and patience during my studies.

January, 2024 **Erkut MEMİŞ**

SENTIMENT ANALYSIS of TURKISH FINANCIAL TWEETS USING DEEP LEARNING MODELS for BIST 100 INDEX

ABSTRACT

Data mining is a process that aims to extract valuable information from a data set. Association mining is one of the field of data mining, which discovers all the relationships and hidden patterns among the elements in a dataset. Patterns are discovered using predefined constraints (e.g., minimum support and minimum confidence values). In the stock market, data mining provides valuable information for all investors in terms of their results.

In this study, daily data of 87 stocks of the BIST-100 stock market index in Turkey between 21.10.2013 and 10.19.2018 were used. The association rules in the BIST-100 stock market index were determined using historical data, Apriori Algorithm and two different concentration methods, and the association results were determined as keywords in the sentiment analysis of Turkish financial tweets.Based on the assumption that finance related tweets can be an important indicator for decision makers if analyzed and interpreted in terms of the stock market, financial tweets containing the keywords determined in the BIST-100 index were collected. After labeling the tweets as “Positive”, “Negative” and “Neutral”, binary and multi-class data sets were created and Neural Network, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU) and GRU-CNN models were used as classifiers.

At the end of the study, the best results were 83.02% for dual classes, and for multi-class data sets it was observed that 73.73% was obtained with the CNN model. The highest performance achieved by the word embedding method is 63.85% through the neural network model and in multi-class data; using the GRU-CNN model, it was realized as 80.56% in the binary data set.

Keywords: Deep learning, sentiment analysis, financial tweet, data mining.

BİST 100 ENDEKSİ İÇİN DERİN ÖĞRENME MODELLERİ KULLANILARAK TÜRKÇE FİNANSAL TWEETLERİN DUYGU ANALİZİ

ÖZ

Veri madenciliği, bir veri kümesindeki değerli bilgileri bulma sürecidir. Birliktelik madenciliği ise veri madenciliğinin alanlarından biri olup bir veri setindeki öğeler arasında bulunan tüm ilişkileri ve gizli kalıpları keşfeder. Desenler, önceden tanımlanmış kısıtlamalar (minimum destek ve minimum güven değerleri gibi) kullanılarak keşfedilir. Borsada veri madenciliği, sonuçları açısından tüm yatırımcılar için değerli bilgiler sunar.

Bu çalışmada Türkiye'deki BİST-100 borsa endeksinin 21.10.2013 ile 10.19.2018 tarihleri arasındaki 87 farklı hisse senedinin günlük verisi kullanılmıştır. BİST-100 borsa endeksindeki birliktelik kuralları tarihsel veriler, Apriori Algoritması ve iki farklı yoğunlaşma yöntemi kullanılarak belirlenmiş olup ilişkilendirme sonuçları, Türkçe finansal tweetlerinin duyarlılık analizinde anahtar kelime olarak belirlenmiştir. Finansla ilgili tweetler borsa açısından analiz edilip yorumlanırsa karar vericiler için önemli bir gösterge olabileceği varsayımından hareketle, BIST-100 endeksinde belirlenmiş olan anahtar kelimeleri içeren finansal tweetler toplanmıştır. Tweetler "Pozitif", "Negatif" ve "Nötr" olarak etiketlendikten sonra ikili ve çok sınıflı veri kümeleri oluşturulmuş; sınıflandırıcı olarak Sinir Ağı, Evrişimli Sinir Ağı (CNN), Uzun Kısa-Süreli Bellek (LSTM), Geçitli Tekrarlayan Birimler (GRU) ve GRU-CNN modelleri kullanılmıştır.

Çalışmanın sonunda en iyi sonuçların ikili sınıflar için %83.02; çok sınıflı veri kümeleri için ise %72.73 ve CNN modeliyle elde edildiği gözlemlenmiştir. Kelime gömme yöntemi kullanıldığında ise, en yüksek başarımların sinir ağı modeli aracılığıyla ve çok sınıflı verilerde %63.85; GRU-CNN modeli kullanılarak ikili veri setinde %80.56 düzeyinde gerçekleşmiştir.

Anahtar kelimeler: Derin öğrenme, duygu analizi, finansal tweet, veri madenciliği.

CONTENTS

NOMENCLATURE

Acronyms

ANN	Artificial Neural Network
BOW	Bag of Words
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
GRU	Gated Recurrent Units
IEEE	Institute of Electrical and Electronics Engineers
ISMSIT	International Symposium on Multidisciplinary Studies and Innovative Technologies
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
NER	Named Entity Recognition
NLP	Natural Language Processing
NN	Neural Network
POS	Part of Speech
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SRL	Semantic Role Labeling
SVM	Support Vector Machine
TF-IDF	Term Frequency – Inverse Document Frequency
WEKA	Waikato Environment for Knowledge Analysis

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Nowadays, the most popular data sharing field is social media; thereby social media sites accumulate huge amount of data. Twitter[1] is one of the commonly used social media data sources. Twitter has nearly 700 million users and 58 million tweets average per day[2]. People post messages that are called as tweets, which may include texts, videos, links etc. Because of its huge popularity and usage, analyzing tweets posted by users becomes more and more important. So, automatically detecting tweets' sentiments is an attractive research area for many researchers.

Sentiment analysis is the outcome of people's emotions, attitudes, opinions, sentiments etc. in their sharing's, which can be written or spoken. This concept especially focuses on polarity detection [1] that identifies negative and positive opinions in the text.

Sentiment analysis is carried out at three levels which are word or phrase level, sentence level and document level [2]. Generally, lexicon based, learning based and hybrid based approaches [3] are used to realize sentiment classification problem.

Tweets are in a way microblogs or short texts, so our sentiment analysis is performed at the sentence level. In our work, we used learning-based approaches to define sentiments in sentences. Financial news from tweets and sentiment analysis of these may contain important information or indicators for the financial or stock market. Although many studies have been conducted for English in the field of sentiment analysis and financial sentiment analysis, not many studies have been published in Turkish yet.

Before we collected the tweets, we studied on association rule mining to determine keywords related to BIST 100 index. We focused on BIST100 stock exchange, and 87 different BIST100 stocks daily data was used for dates between 10/21/2013 and 10/19/2018. We used Apriori algorithm to identify association rules on BIST100 index. There were some researches like these but to the best of our knowledge, there

[1] https://twitter.com
[2] http://www.statisticbrain.com/twitter-statistics

weren't any studies on relations between BIST100 stocks. We presented this part of studies with title "Association Rule Mining on the BIST100 Stock Exchange" in ISMSIT 2019 and published in IEEE Xplore as an proceeding [4]. These studies included with details in chapter 4.

Turkish Financial Tweets were collected with determined keywords from BIST 100 index using association rule mining [4] and the tweets were tagged as "POSITIVE", "NEGATIVE" and "NEUTRAL". Binary dataset including only positive and negative classes and multi-class dataset including positive, negative and neutral classes were created. Chapter 4 includes details of those processes.

Noisy or unclear sentences negatively affected the sentiment classification process. In order to prepare these tweets for the analysis we used pre-processing, which includes stop word removal, normalization processes etc. "ITU Turkish NLP Web Service API" was utilized for the Turkish text normalization process [5].

Deep learning algorithms and methods have provided great improvements on pattern recognition and image recognition fields. These improvements led to Natural Language Processing (NLP) researchers to be able to focus on deep learning methods. Using dense vector representation based on neural networks has achieved better results on NLP tasks. Success of word embedding [6, 7] and deep learning methods [8] caused a trend which is using deep learning algorithms on NLP tasks. In contrast, to the traditional machine learning based NLP systems, which are using handmade features, deep learning enables automatic feature representation learning. Handmade features has several bottlenecks [9]. We used word embedding and pre-trained word embedding with fastText [10] for feature representation in our work.

Neural Network, Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM), Gated Recurrent Units (GRU) and GRU-CNN models are used as a sentiment classifier in this study. The performances of these models were evaluated based on their accuracies.

CHAPTER 2

BACKGROUND

2.1 Association Rule Mining

Frequent patterns are patterns that occur frequently in a data set. Frequent patterns have an important role in association mining, data classification, clustering and other data mining techniques. A typical example of frequent data mining is market basket analysis. This process analyses customer buying habits by finding relations between items in customer's shopping basket. Market Basket Analysis is the earliest form of frequent pattern mining [11]. So significant amount of terminology used to describe both the data (e.g. transactions) and the output (e.g. itemsets) is borrowed from the supermarket analogy [12].

Association rules can be generated by using frequent item sets. Association mining was first introduced by Agrawal, Imielinski and Swami [13]. The research results show the hidden patterns in large datasets. The association rules algorithm reveals the relationships between items or features that occur frequently in databases. For instance, in market basket analysis case, if people buy item A and item B together, we can say that there is a relationship between item A and item B. The association rule mining commonly defined as follows [11].

Let $I = I_1, I_2, ..., I_m$ be an itemset. Let D, be a set of database transactions where each transaction T is a nonempty itemset such that $T \subseteq I$. Each transaction is associated with an identifier, called a TID. Let A be a set of items. A transaction T is said to contain A if $A \subseteq T$. An association rule is an implication of the form A⇒B, where A⊂I, B⊂I, A≠ ∅, B≠ ∅, and A∩B=∅. The rule A⇒B holds in the transaction set D with **support** s, where s is the percentage of transactions in D that contain A∪B (i.e., the union of sets A and B say, or, both A and B). This is taken to be the probability, P(A∪B). The rule A⇒B has **confidence** c in the transaction set D, where c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability,

P(B|A). That is;

$$Support(A \Rightarrow B) = P(A \cup B) \quad \textbf{(2.1)}$$

$$Confidence(A \Rightarrow B) = P(B \mid A) = \frac{Support(A \cup B)}{Support(A)} \quad \textbf{(2.2)}$$

Rules that support both a minimum support threshold (minsup) and minimum confidence threshold (minconf) are called **strong** [11].

Association rule mining consists two phases;

1. At first phase; all the frequent item sets are generated that provides minimum support threshold.
2. In the second phase; the association rules are generated from the frequent item sets that provides minimum confidence threshold [12].

The support and confidence measures are insufficient at filtering out uninteresting association rules. To tackle this weakness, a correlation measure can be used to augment the support-confidence pair for association rules. There are several correlation measures. **Lift** is one of these correlation measures. It is described as; The occurrence of itemset A is independent of the occurrence of itemset B if P(A∪B) = P(A)P(B); otherwise itemsets A and B are dependent and correlated as events. The **lift** between the occurrence A and B can be calculated by Equation **2.3**;

$$Lift(A,B) = P(B \mid A) = \frac{P(A \cup B)}{P(A)P(B)} = \frac{P(B \mid A)}{P(B)} = \frac{conf(A \Rightarrow B)}{sup(B)} \quad \textbf{(2.3)}$$

If its value is less than 1, then the occurrence of A is negatively correlated with the occurrence of B, meaning that the occurrence of one likely leads to the absence of the

other one. If its value greater than 1, then A and B are positively correlated, meaning that the occurrence of one implies the occurrence of other. If its value equal to 1, then A and B are independent and there is no correlation between them [11].

2.2 Apriori Algorithm

Apriori is an algorithm to find association rules. It was initially introduced by Agrawal and Srikant in 1994 [14]. It uses preceding information of frequent item sets. A repetitive process, level-wise search is used in Apriori. In this process, (k+1)-itemsets are calculated from k-itemsets. There are some key concepts used in this algorithm such as **Frequent Itemsets**, **Apriori Property**, **Join** and **Prune** operations. Apriori property, which says that each nonempty subsets of frequent item set must also be frequent, improves the algorithm efficiency. Apriori algorithm is two steps processes that are join and prune steps that are defined as follows [11].

The join step: L_{k-1} includes (k-1) itemsets that provide minimum support and confidence values. L_{k-1} is joined with itself to determine candidate k-itemsets, C_k. C_k is used to find L_k [11].

The prune step: L_k is formed from candidate k itemsets, C_k. Each candidate item is controlled to provide minimum support and confidence values to generate L_k. But C_k can be huge and this requires heavy computation. Apriori property is used to reduce the size of C_k [11].

2.3 Sentiment Analysis

Sentiment analysis, also called opinion mining, is the outcome of people's emotions, opinions, sentiments, attitudes etc. in their sharing which can be written or spoken. This concept especially focuses on polarity detection [1] that identifies positive and negative opinions in the text.

Although linguistics and natural language processing (NLP) have a long history, few researches had been done on sentiment analysis before the year 2000. Since then the field had become a very active research area. Some reasons for this are;

i. Wide range of application area for commercial usage,

ii. Many challenging research areas which had never been studied,

iii. Huge opinionated data [2].

With the explosive growth of social media, sentiment analyses become a key research topic, because people wants to learn others opinions on any topic or subject

2.3.1 Sentiment Analysis Levels

Sentiment analysis is performed at three levels; document level, sentence level and word/phrase level.

1. **Document Level:** This level analysis interested in whole document, and classifies its opinion as a positive, negative or neutral sentiment. It assumes that each document has opinions on a single topic. Supervised learning methods are the most used techniques for document level analysis, but unsupervised methods are also used [2].

   Document level sentiment classification is a kind of text classification problem. Traditional text classification generally classifies documents according to their topics, e.g., sports, finance, and politics. Topic related words are key elements for text classifications. As a kind of text classification, sentiment classification trusts sentiment words for classification process [2].

   Because it is a kind of text classification, existing supervised learning methods such as SVM (Support Vector Machines) can be applied to this problem [2].

   This level of classification has some drawbacks:

   i. In many application user wants to learn detailed information like sentiments for aspects of entities. Document level classification does not provide this information.

   ii. Document level classification cannot easily applied to blogs, discussions, because they have multiple entities [2].

2. **Sentence Level:** This type of analysis determines sentiment for each sentence as positive, negative or neutral [2].

3. **Entity and Aspect level:** Both of the document level and sentence level sentiment analysis have drawbacks, because they don't determine what exactly people like or don't like. Aspect level has more grained analysis. It was earlier called feature level (feature based opinion mining and summarization) [15]. It is interested in the opinion itself. It discovers that an opinion consists of sentiment (positive, negative or neutral) on a target (of opinion). For example "Although the fuel consumption is not good, I like this car". This sentence has a positive sound but it is not exactly positive sentence. In fact it is positive about the car but negative about its fuel consumption. Thus this kind of analysis is to determine sentiments on entities and/or their aspects. For example, the sentence, "The iPhone's picture quality is good but its sound is not good", has two aspects, picture quality and sound of iPhone (entity). The sentiment on iPhone picture quality is positive, but the sentiment on its sound is negative [2].

   Aspect level analysis is more difficult than both of sentence and document level analysis because it consists of several sub problems [2].

2.3.2 Sentiment Lexicons and Challenges

Sentiment or opinion words are the most important sentiment indicators for sentiment analysis process. These words are generally used to show positive or negative sentiments. For example good, excellent, and perfect are positive sentiment words, however bad, weak, and disgusting are negative sentiment words. Idioms and phrases are also important sentiment indicators. A sentiment words and phrases list is called a sentiment/opinion lexicon [2].

Sentiment lexicon is important for sentiment analysis but it is not enough to solve whole sentiment analysis problem by itself because of complexity. Some challenges or issues with sentiment lexicon are:

i. Positive or negative sentiment words may change their sentiment in different domains. For example "suck" generally has negative sentiment but sometimes it

also has positive sentiment, e.g., “This phone sucks.”, however it has also positive sentiment, e.g., “this vacuum cleaner sucks well.” [2].

ii. Some type of sentences (e.g., question and conditions) contains sentiment words but they may not have any sentiment. For example “which phone is good?” has a positive sentiment word but the sentence has not a sentiment [2].

iii. Sarcastic sentences are difficult to find sentiment. For example, “What a good player!, he did not play any match” [2].

iv. Sentences without sentiment words may have sentiment. For example “The car consumes a lot of fuel” has negative sentiment but it has not any sentiment word [2].

2.3.3 Sentiment Analysis and Approaches

Sentiment analysis approaches are roughly classified into **lexicon based approach**, **learning based approach** and **hybrid approach** [3].

The lexicon based approach benefits from predefined dictionaries of semantic words to determine sentiment by using simple basic mapping. This approach does not require training process. Moreover it is fast because of using predefined word dictionaries. However this advantage causes to the main drawback of this approach, which it is not adaptable to specific domain data.

Learning-based approaches apply machine learning algorithms to analyse the sentiment of the text. In this method, predefined sets are also required which contain positive and negative words. But it has machine learning algorithms which are much more complex than simple basic mapping used in lexical methods. In the machine learning algorithms, learning phase is carried out with training sets and after the learning phase, the system is ready to decide about new samples.

The hybrid approach combines both approaches. The various approaches and the most popular algorithms of sentiment analysis are showed in Figure 2.1 [3].

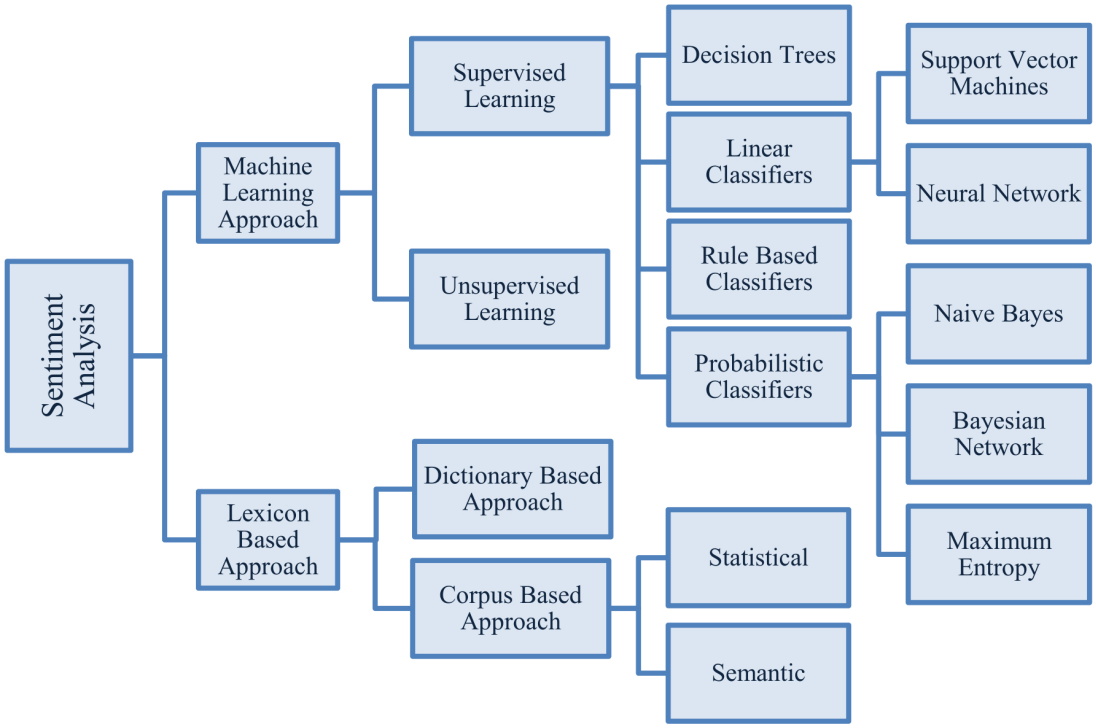


Figure 2.1 Sentiment analysis approaches and algorithms [3]

2.4 Feature Extraction

Machine learning algorithms cannot directly run on text data, the text must be converted into numerical values. This process is called "Feature Extraction". There are numerous types of feature extraction methods. Some popular feature extraction methods from text are Bag of Words (BoW) and Word Embedding. We used word embedding method in our work.

2.4.1 Bag of Words (BOW)

Each document is represented as vector d. each dimension of vector d consists a distinct term in the term spaces of document collection. We represent each vector d as;

$$d = (w_1, w_2, w_3, \ldots, w_n) \tag{2.4}$$

where w_i is the weight of i^th term of document d. **Boolean weighting** and **TF-IDF** are most commonly used weighting algorithms.

Boolean weighting has a binary representation for term weight. Its weight is consider

as 1 if document consists the term otherwise it is considered as 0. The equation of Boolean weighting is;

$$w_i = \begin{cases} 1, & \text{if } tf_i > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

where tf_i is the frequency of term i in the document [16].

TF-IDF (Term Frequency – Inverse Document Frequency) weighting equation is as follow;

$$w_i = tf_i * log(\frac{n}{n_i}) \tag{2.6}$$

where, tf_i is the frequency of term i in document d, n is the total number of documents, and n_i is the number of documents that includes term i [16].

BOW creates a vector for document with lots of zero values. This vector is called a sparse vector. Sparse vectors require more memory and computational resources when modelling and the vast number of positions or dimensions can make the modelling process very challenging for traditional algorithms [17].

One of the major disadvantages of using BOW is that it discards word order thereby ignoring the context and in turn meaning of words in the document. For natural language processing (NLP) maintaining the context of the words is of utmost importance. To solve this problem Word Embedding approach is used [18].

2.4.2 Word Embedding

It is a representation of text where words that have the same meaning have a similar representation. In other words it represents words in a coordinate system where related words, based on a corpus of relationships, are placed closer together [18, 19].

Text is considered a form of sequence data similar to time series data that you would have in weather data or financial data. In the previous BOW model, we have seen how to represent a whole sequence of words as a single feature vector. Word embedding represents each word as vector.

One-hot encoding is the simplest word embedding method to represent word as a vector [20].

It is simply done by taking a vector of the length of the vocabulary with an entry for each word in the corpus. In this way, you have for each word; given it has a spot in the vocabulary, a vector with zeros everywhere except for the corresponding spot for the word which is set to one. This can become a fairly large vector for each word and it does not give any additional information like the relationship between words [20].

This is often used when you have a categorical feature which you cannot represent as a numeric value but you still want to be able to use it in machine learning. One use case for this encoding is of course words in a text but it is most prominently used for categories. Such categories can be for example city, department, or other categories [20].

But Word Embedding methods represent words as dense word vectors (also called word embeddings) which are trained unlike the one-hot encoding which are hardcoded. This means that the word embeddings collect more information into fewer dimensions [20].

The word embeddings map the statistical structure of the language used in the corpus. Their aim is to map semantic meaning into a geometric space. This geometric space is then called the embedding space [20].

This would map semantically similar words close on the embedding space like numbers or colors. If the embedding captures the relationship between words well, things like vector arithmetic should become possible. A famous example in this field of study is the ability to map King - Man + Woman = Queen [20].

How can you get such a word embedding? You have two options for this. One way is to

train your word embeddings during the training of your neural network. The other way is by using pre-trained word embeddings which you can directly use in your model. There you have the option to either leave these word embeddings unchanged during training or you train them also [20].

Some of the well-known models of word embedding are Word2Vec [21], GloVe [22] and fastText [23]. We used word embedding and pre-trained word embedding with fastText in our study the feature vector size is 300.

2.4.2.1 Word2vec

Word2vec is a common used word embedding method that was proposed by Google researchers Mikolov et al. in 2013 [21]. They proposed training a simple neural network (one with only linear activations – no sigmoid, relu, or other non-linear activations) to learn embeddings for words based on the context in which they are used. Mikolov trained a network using a dictionary of the million most-frequently-used words in a training corpus consisting of the 6 billion words from Google News. An example vector space consisted of a set of 300 floating point values [24].

The most interesting thing about Mikolov's 300-dimensional vector space is that words with similar syntactic and semantic meanings tend to group near one another in various dimensions of the space. And, surprisingly, one can do simple vector math on the learned embedding vectors to explore these relationships. For example, if you train a network to learn a set of word embeddings, and you take the vector representing the word "King", subtract the vector for the word "Man" and add the vector for the word "Woman" you will find that the word closest to the resulting vector in the word2vec vector space turns out to be "Queen" [24].

Mikolov's group trained their network using two different methods: Continuous Bag of Words (CBOW) and the skip-gram model. Given a set of sentences (also called corpus) the model loops on the words of each sentence and either tries to use the current word w in order to predict its neighbors (i.e., its context), this approach is called **"Skip-Gram"**, or it uses each of these contexts to predict the current word w, in that case the method is called **"Continuous Bag Of Words" (CBOW)**. To limit the number of words in each

context, a parameter called “window size” is used [25].

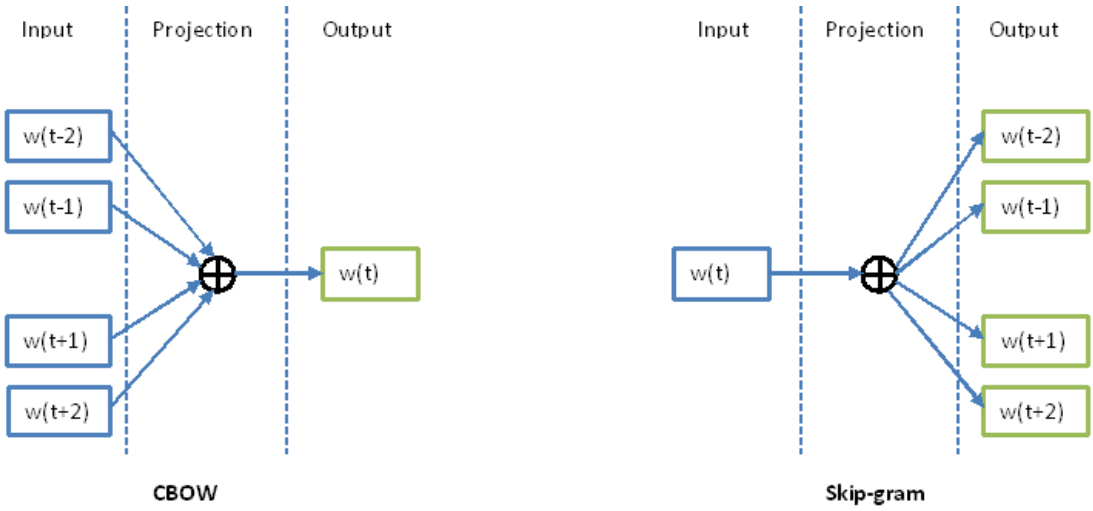


Figure 2.2 CBOW and skip-gram architectures [21]

The **CBOW** method tries to guess a particular word based on the 4 words preceding the target word and the four words following the target word (the window size of 4 words is arbitrary, another hyper parameter, but Mikolov found that a +/- 4 window size was found to produce good results). Positive training examples were produced from the Google News corpus; negative examples were generated from random sets of preceding and following words. Figure 2.2 shows a pictorial representation of the CBOW model [24].

As we can see, the input in the CBOW model in this example consists of the embeddings for the two words preceding the target word and the two words following the target word, and the desired output is then the predicted target word. This network is trained to produce high activations for examples from the corpus and low activations for random sequences of input words [24].

The **skip-gram** model turns this network on its head: rather than predicting the word that would lie between a set of preceding and succeeding words, the skip-gram model takes as input a single word and predicts, as output, the most likely set of words to precede and succeed the input word, as shown in Figure 2.2 [24].

It turns out that the skip-gram model actually does the better job at determining the “best” set of embeddings for a dictionary of words, where “best” means that the vector

encodings for words in the dictionary end up being located in the 300-dimensional vector space such that words with similar syntactic and semantic meanings are located near one another in this vector space [24].

2.4.2.2 fastText

English words usually have internal structures and formation methods. For example, we can deduce the relationship between “dog”, “dogs”, and “dogcatcher” by their spelling. All these words have the same root, “dog”, but they use different suffixes to change the meaning of the word. Moreover, this association can be extended to other words. For example, the relationship between “dog” and “dogs” is just like the relationship between “cat” and “cats”. The relationship between “boy” and “boyfriend” is just like the relationship between “girl” and “girlfriend”. This characteristic is not unique to English. In French and Spanish, a lot of verbs can have more than 40 different forms depending on the context. In Finnish, a noun may have more than 15 forms. In fact, morphology, which is an important branch of linguistics, studies the internal structure and formation of words [26].

Word2vec does not use morphology information. Both of the skip-gram model and continuous bag-of-words model use different vectors to represent words with different forms. For example, “dog” and “dogs” are represented by two different vectors, while the relationship between these two vectors is not directly represented in the model. In view of this, fastText [23] proposes a subword embedding method. Based on the skip-gram model in word2vec, it represents the central word vector as the sum of the subword vectors of the word. Subword embedding utilizes the principles of morphology, which usually improves the quality of representations of uncommon words [26].

2.5 Deep Learning Based Natural Language Processing

Natural language processing (NLP) is a theory-motivated range of computational techniques for the automatic analysis and representation of human language. NLP research has evolved from the era of punch cards and batch processing, in which the analysis of a sentence could take up to 7 minutes, to the era of Google and the likes of

it, in which millions of webpages can be processed in less than a second [27]. NLP enables computers to perform a wide range of natural language related tasks at all levels, ranging from parsing and part-of-speech (POS) tagging, to machine translation and dialogue systems [9].

Deep learning architectures and algorithms have already made impressive advances in fields such as computer vision and pattern recognition. Following this trend, recent NLP research is now increasingly focusing on the use of new deep learning methods. For decades, machine learning approaches targeting NLP problems have been based on shallow models (e.g., SVM and logistic regression) trained on very high dimensional and sparse features. In the last few years, neural networks based on dense vector representations have been producing superior results on various NLP tasks. This trend is sparked by the success of word embedding [6, 7] and deep learning methods [8]. Deep learning enables multi-level automatic feature representation learning. In contrast, traditional machine learning based NLP systems liaise heavily on hand-crafted features. Such hand-crafted features are time-consuming and often incomplete [9].

Collobert et al. [28] demonstrated that a simple deep learning framework outperforms most state-of-the-art approaches in several NLP tasks such as named-entity recognition (NER), semantic role labelling (SRL), and POS tagging. Since then, numerous complex deep learning based algorithms have been proposed to solve difficult NLP tasks. Some of the major deep learning related models and methods applied to natural language tasks are neural networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and recursive neural networks [9].

2.5.1 Neural Networks

Neural networks, or sometimes called artificial neural network (ANN) or feed forward neural network, are computational networks which were vaguely inspired by the neural networks in the human brain. They consist of neurons (also called nodes) which are connected like in the Figure 2.3 [20].

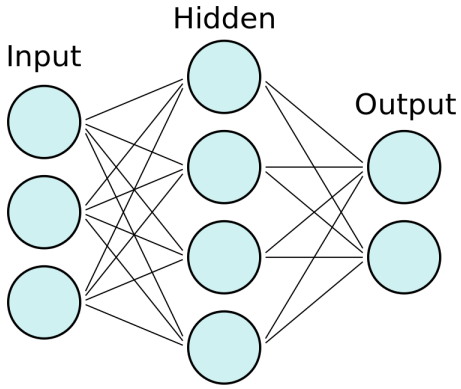


Figure 2.3 Neural network model

We start by having a layer of input neurons where we feed in our feature vectors and the values are then feed forward to a hidden layer. At each connection, we are feeding the value forward, while the value is multiplied by a weight and a bias is added to the value. This happens at every connection and at the end we reach an output layer with one or more output nodes [20]

For a binary classification we can use one node, otherwise we should use multiple nodes for each class. We can have as many hidden layers as we wish. In fact, a neural network with more than one hidden layer is considered a deep neural network. The formula from one layer to the next is as shown in Equation **2.7** [20].

$$o_j = f\left(\sum_i w_{i,j} a_i + b_i\right) \tag{**2.7**}$$

The layer with nodes "a" serves as input for the layer with nodes "o". In order to calculate the values for each output node, we have to multiply each input node by a weight "w" and add a bias "b" [20].

All of those have to be then summed and passed to a function f. This function is considered the activation function and there are various different functions that can be used depending on the layer or the problem. It is generally common to use a **rectified linear unit (ReLU)** for hidden layers, a **sigmoid** function for the output layer in a **binary classification** problem, or a **softmax** function for the output layer of **multi-class** classification problems [20].

Weights calculation is obviously the most important and also the most difficult part of neural networks. The algorithm starts by initializing the weights with random values and they are then trained with a method called **backpropagation**. This process is done by using optimization methods such as gradient descent, and Adam to reduce the error between expected and calculated output values. Adam is the most common optimization method. Loss function is used for error calculation. There are also different loss functions; binary cross entropy is one of those functions [20].

2.5.2 Convolutional Neural Networks (CNN)

A CNN is basically a neural-based approach which represents a feature function that is applied to constituting words or n-grams to extract higher-level features. The resulting abstract features have been effectively used for sentiment analysis, machine translation, and question answering, among other tasks. Collobert and Weston [29] were among the first researchers to apply CNN-based frameworks to NLP tasks. The goal of their method was to transform words into a vector representation via a look-up table, which resulted in a primitive word embedding approach that learn weights during the training of the network (see Figure 2.4) [9, 30].

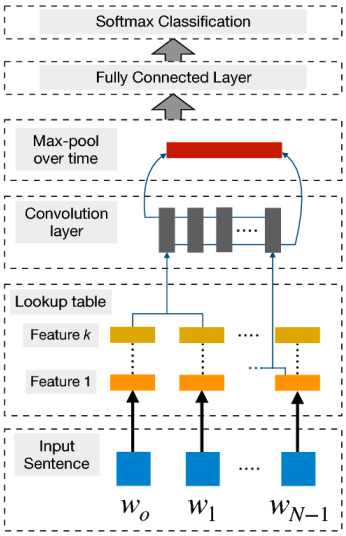


Figure 2.4 CNN framework used to perform word wise class prediction [9]

In order to perform sentence modelling with a basic CNN, sentences are first tokenized into words, which are further transformed into a word embedding matrix (i.e., input embedding layer) of d dimension. Then, convolutional filters are applied on this input embedding layer which consists of applying a filter of all possible window sizes to

produce what's called a feature map. This is then followed by a max-pooling operation which applies a max operation on each filter to obtain a fixed length output and reduce the dimensions of the output. And that procedure produces the final sentence representation [30]. Figure 2.5 shows CNN modelling on text.

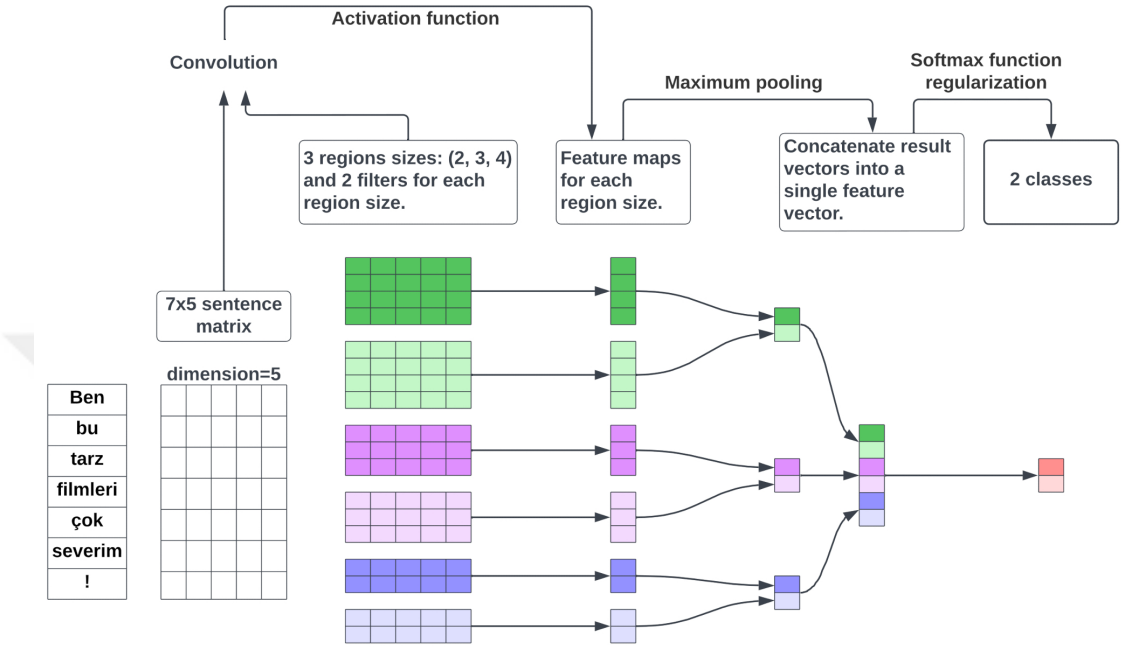


Figure 2.5 CNN modeling on text [9]

2.5.3 Recurrent Neural Networks (RNN)

RNNs [31] are specialized neural-based approaches that are effective at processing sequential information. An RNN recursively applies a computation to every instance of an input sequence conditioned on the previous computed results. These sequences are typically represented by a fixed-size vector of tokens which are fed sequentially (one by one) to a recurrent unit. Figure 2.6 illustrates a simple RNN structure [30].

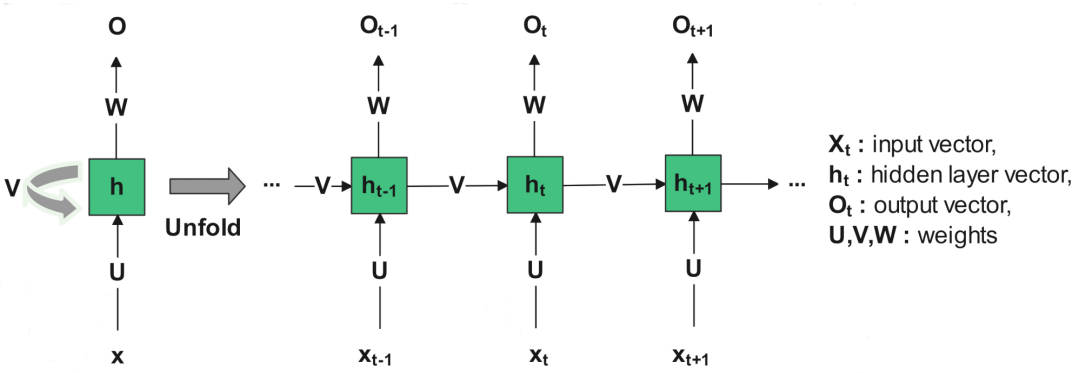


Figure 2.6 Simple recurrent neural network structure [9]

Simple RNN; in the context of NLP, RNNs are primarily based on Elman network [31] and they are originally three layer networks. Above figure illustrates a more general RNN which is unfolded across time to accommodate a whole sequence. In the figure, x_t is taken as the input to the network at time step t and s_t represents the hidden state at the same time step [9].

Calculation of s_t is based on the Equation **2.8**:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{2.8}$$

Thus, s_t is calculated based on the current input and the previous time step's hidden state. The function f is taken to be a non-linear transformation such as tanh, ReLU and U, V, W account for weights that are shared across time. In the context of NLP, x_t typically comprises of one-hot encodings or embedding. At times, they can also be abstract representations of textual content. ot illustrates the output of the network which is also often subjected to non-linearity, especially when the network contains further layers downstream [9].

The hidden state of the RNN is typically considered to be its most crucial element. As stated before, it can be considered as the network's memory element that accumulates information from other time steps. In practice, however, these simple RNN networks suffer from the infamous vanishing gradient problem, which makes it really hard to learn and tune the parameters of the earlier layers in the network [9].

This problem has led to the development of various RNN derivative models like long short-term memory (LSTM) and gated recurrent units (GRU) [9].

The main strength of an RNN is the capacity to memorize the results of previous computations and use that information in the current computation. This makes RNN models suitable to model context dependencies in inputs of arbitrary length so as to create a proper composition of the input. RNNs have been used to study various NLP tasks such as machine translation, image captioning, and language modelling, among others [30].

As it compares with a CNN model, an RNN model can be similarly effective or even better at specific natural language tasks but not necessarily superior. This is because they model very different aspects of the data, which only makes them effective depending on the semantics required by the task at hand [30].

2.5.4 Long Short-Term Memory (LSTM)

LSTM [32, 33], LSTM has "forget gates" in addition to the simple RNN architecture to handle vanishing and exploding problems. Figure 2.7 shows the LSTM structure.

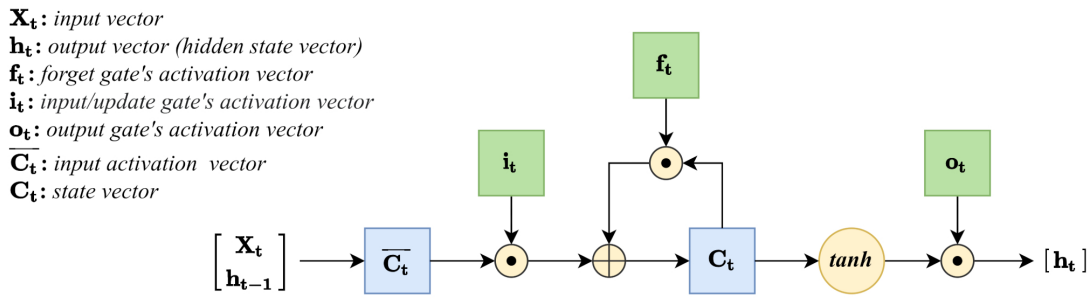


Figure 2.7 Long short-term memory [9]

Unlike the simple RNN, LSTM back propagates errors through a limitless number of time steps [9].

2.5.5 Gated Recurrent Unit (GRU)

GRU is an another RNN derivative model. It has lesser complexity but has a similar performance to LSTM. GRU adds reset and update gates to simple RNN. Figure 2.8 shows a gated recurrent unit.

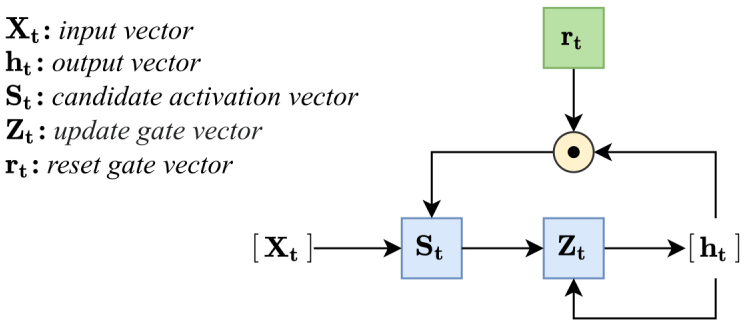


Figure 2.8 Gated recurrent unit [9]

CHAPTER 3

LITERATURE REVIEW

Data mining on stock market data is one of the attracted research subject due to its financial benefits. Finding hidden patterns and associations in the stock exchange data can help us to understand market behaviors and movements of stocks. In the past, generally two analysis methods (technical analysis and fundamental analysis) were used to predict movements in stock market, but associations between all companies in the stock exchange were considered in a limited extend. Technical analysis is based on historical stock market prices and time series analysis techniques are used in this method. The second method, fundamental analysis trusts company information (company value, loans, profits, investments etc.) and economic values (interest rate, inflation, unemployment etc.). Recently, there are many studies based on association mining on stock exchange.

Association rule mining exposes hidden patterns on a data set items by using frequently occurred items with some constraints. Association rules show regularities in a database. Association mining was first introduced by Agrawal, Imielinski and Swami [13] and it is used in different areas to find hidden patterns. Stock markets are one applied area of the association mining. And researches show that most the world's stock markets are integrated and associated each other and also companies in a stock market have relations each other. Wu and Su studied on four major international stock markets and they show that there are significant dynamic relations between of them [34]. They also find that the returns of large markets lead to the returns of small markets. Almohaimeed used an Apriori type algorithm to identify the association rules between S&P 500 and the stock prices [35].

Ariya has studied stock forecasting using association rule mining. Ariya selected eight stock groups which are financial, agro-industry, consumer product, service, properties & construction industry group, resource, technology and industry. Each group has four randomly selected sample stocks [36]. She used Apriori algorithm to discover associations between items in databases. She presented the way to prepare data for

running on Apriori algorithm and the result showed unknown relationship of stocks.

Kumar and Kalia studied association rule mining on the stocks data set of thirteen years period i.e. from 1996 to December 2008 of NSE stock exchange that amounts to 3252 days was used [37]. First item sets are mined for a given minimum support and based on these item sets obtained; the association rules are computed for a given minimum confidence. The generated patterns help investors to build their portfolio and using these patterns investment strategies may be planned.

Karpio, Lukaseiewicz, Orlowski and Zabrowski studied mining associations on the Warsaw Stock Exchange [38]. Their study implements the association rules using a data mining approach to explore co-movement between stock items listed on the Warsaw Stock Exchange. To explore associations they used Apriori algorithm.

Şimşek and Özdemir have studied relation between Turkish twitter messages and stock market index. They examined a Turkish twitter data set using data mining techniques. They created emotional corpus and evaluated the happiness and unhappiness level of tweets. They show that stock market and tweet data relation is about approximately 45% [39].

Yıldırım and Yüksel studied investigation of relationship between social media and stock price daily movement direction: sentiment analysis implementation. In their study, they select a telecommunication company and whose shares are processed in Borsa Istanbul. For a selected period, daily opening and closing prices of this company have collected and daily raise and fall of the stock price has been classified accordingly. Sentiment analysis has been conducted for the same period and daily polarity values have been obtained. In their study, public sentiment analysis and stock price movement direction is conducted. They used Spearman's rank correlation test. According the test result, a negative and moderation correlation exists between daily stock price and the public sentiments in tweets [40].

Öztürk and Çiftçi have studied to predict exchange rate movements with twitter content. They used daily exchange rate of USD/TR and twitter sentiments have been collected

with the keywords #USD/TR, USD/TR, Dollar, #Dollar. If exchange rate increases, its value selected as 1 for a given date, or its value selected 0. Collected tweets are sorted 3 different categories depending on their sentiments: Buy, Sell, and Neutral. They find that there is a significant relationship between Twitter sentiment and USD/TR movement [41].

Savaş and Can have studied Euro-Dollar Parity and Effects of The Real Exchange Rates on The Index of IMKB 100 [42]. In their study, the relation between the stock prices that are traded in IMKB with Euro-Dollar Parity and Real Effective Exchange Rate Index are examined by the Multiplier Linear Regression and this relation is also investigated by the Granger Causality Test. Monthly data set are used including from the January 2000 to July 2009.

Kendirli and Çankaya have studied, the causality between of the dollar exchange rate and Istanbul Stock Exchange National 30 Index (BIST-30) was investigated with "Granger Causality Test" [43].

CHAPTER 4

MATERIALS AND METHODS

In this study, tweets from BIST-100 stock exchange index were analyzed as a financial indicator. Some companies from BIST 100 stock exchange index were determined as keywords using association rule mining and tweets were collected using these keywords.The collected tweets were tagged as “Positive”, “Negative” and “Neutral”. Then binary and multi-class datasets were created. As a classifier, Neural Network, Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM), Gated Recurrent Units (GRU) and GRU-CNN models were used.

4.1 Association Rule Mining on BIST 100 Index

In this part, we tried to find relations between BIST 100 stock exchange index companies. We focused on BIST100 stock exchange, used 87 different BIST100 stocks daily data for dates between 21.10.2013 and 19.10.2018. We used Apriori algorithm to identify association rules on BIST100 stock exchange index. These relations are valuable information for all of the investors. Also, when we predict direction of BIST100 from Turkish financial tweets, we can use one company tweets for another company if there is relation between them. There are some researches like these but to the best of our knowledge, there is not any studies on relation between BIST100 stocks.

4.1.1 BIST100 Stock Market Data

The stock items listed on the BIST100 Stock Exchange were analyzed. 87 different BIST100 stock items’ daily data were collected for dates between 10/21/2013 and 10/19/2018 from yahoo finance[1]. These data includes Date, Open, High, Low, Close, Adj. Close, Volume columns.Sample stock item data can be shown in Table 4.1.

[1]https://finance.yahoo.com/

Table 4.1 Sample stock item data

Date	Open	High	Low	Close	Adj Close	Volume
21.10.2013	29.799999	29.9	29.200001	29.200001	25.479435	303459
22.10.2013	29.4	29.6	29.200001	29.299999	25.566694	126933
23.10.2013	29.4	29.5	29.299999	29.4	25.653952	165327
24.10.2013	29.4	29.4	28.9	29.299999	25.566694	134552
25.10.2013	29.200001	29.200001	28.9	29	25.30492	114069
28.10.2013	29.1	29.200001	28.799999	28.799999	25.130402	100740

For the purpose of analysis, we have defined “positive”, “negative” and “same” tags using items close values. Close values of items fluctuate over days. The difference between close values within the range of [-1%; 1%] are called “same”. If difference is bigger than 1% of item close value is called “positive” otherwise it is called “negative”. Using these rules, we created new files that include date and related stocks names columns and value of columns includes “same”, “positive” and “negative” tags (We wrote a python program for this purpose). The new file includes all stock items with transformed values can be shown in Table 4.2.

Table 4.2 All stock items transformed data

Date	Stock1	Stock2	Stock3	...	Stock87
21.10.2013	same	same	same	...	same
22.10.2013	negative	positive	positive	...	positive
23.10.2013	negative	positive	negative	...	negative
24.10.2013	positive	positive	negative	...	positive
25.10.2013	positive	positive	same	...	positive
28.10.2013	negative	positive	positive	...	negative
...	...	...	...	...	...

Listing A.1 in Appendix A shows our python code that transforms stock items data files to the tag file.

4.1.2 Generating Association Rules from Frequent Item sets

Strong association rules that satisfy minimum support and minimum confidence are generated from frequent itemsets.

4.1.3 Methodology for Association Rule Generation

In this part, we have studied on a question that is finding association rules on BIST100 companies using historical data and Apriori algorithm. This research only focuses on close prices of stocks. 5 years daily data of 87 different BIST100 stocks were collected for date between 21.10.2013 and 19.10.2018 from "finance.yahoo.com". These data includes Date, Open, High, Low, Close, Adj. Close, Volume columns. We transformed this data as we explained in subsection 4.1.1 - BIST100 Stock Market Data. Finally we used Apriori algorithm to find association rules in selected stocks set. We implemented a python program for data transformation and we used WEKA to find association rules [4].

Two different methods were used to find association rules. At first method, we used all 87 stocks to find association rules and we removed best 2 stocks from stocks set. We iterate this process until to find 14 rules. We called this method as **"decreasing"**. At second method we create different stocks sets based on sectors such as, banks, energy, automobile, industry etc. We found association rules on these sets. We called this method as **"sectoral"** [4].

The research procedure for decreasing method is shown in Figure 4.1. And research procedure for sectoral method is shown in Figure 4.2.

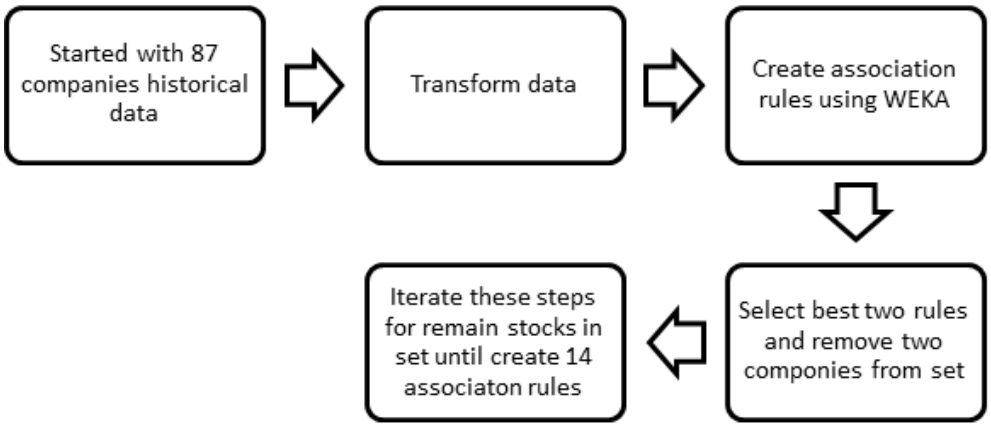


Figure 4.1 Proposed decreasing method [4]

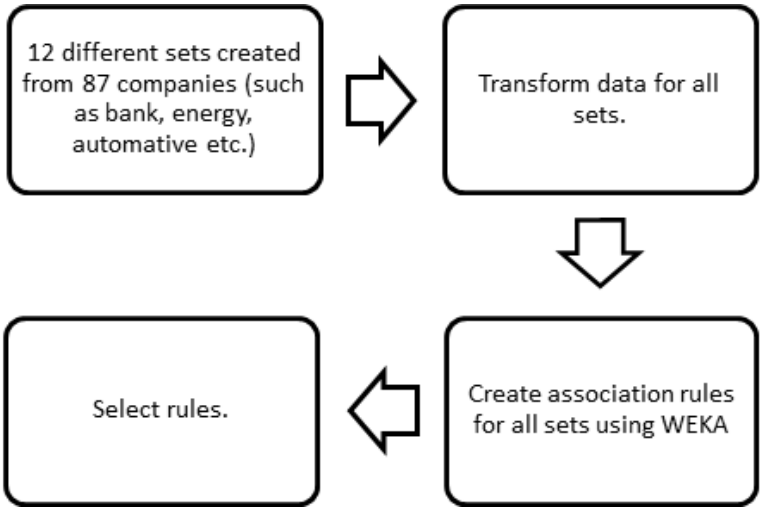


Figure 4.2 Proposed sectoral method [4]

4.1.4 Results and Discussions of Association Rule Generation Methods

In decreasing method, after data preparation that includes 87 companies stock data, Apriori algorithm was run in WEKA. Prepared data created from all 87 companies is shown in Table 4.3 [4].

Table 4.3 Prepared stock data from 87 companies historical stock data close prices

Date	AEFES	AFYON	AKBNK	...	YKBNK	ZOREN
21.10.2013	same	same	same	...	same	same
22.10.2013	negative	positive	positive	...	positive	positive
23.10.2013	negative	positive	negative	...	negative	positive
24.10.2013	positive	positive	negative	...	same	positive
25.10.2013	positive	positive	same	...	positive	negative
28.10.2013	negative	positive	positive	...	same	positive
29.10.2013	same	same	same	...	same	same
30.10.2013	same	positive	negative	...	negative	negative
31.10.2013	negative	positive	negative	...	negative	negative
1.11.2013	negative	positive	negative	...	negative	positive
4.11.2013	negative	negative	negative	...	negative	negative
...	...	...	...	...	...	...
17.10.2018	same	positive	positive	...	positive	same
18.10.2018	positive	negative	negative	...	negative	negative
19.10.2018	positive	negative	negative	...	negative	negative

When we run Apriori algorithm on this data, we figure out association rules as shown in Table 4.4 [4].

Table 4.4 Generated association rules for 87 companies

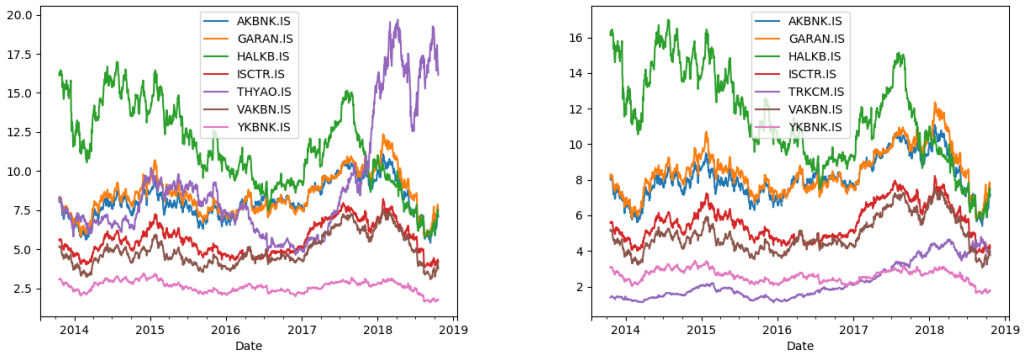
No	Weka Apriori -N 10 -T 0 -C 0.98 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1, Insts: 1305, Attrs: 87, Minimum support: 0.2 (261 instances), Minimum confidence: 0.9
1	AKBNK.IS=positive HALKB.IS=positive ISCTR.IS=positive THYAO.IS=positive VAKBN.IS=positive YKBNK.IS=positive 295 ==> GARAN.IS=positive 292 confidence: 0.99
2	AKBNK.IS=positive HALKB.IS=positive ISCTR.IS=positive PGSUS.IS=positive VAKBN.IS=positive YKBNK.IS=positive 282 ==> GARAN.IS=positive 279 confidence: 0.99
3	AKBNK.IS=positive ECILC.IS=positive HALKB.IS=positive ISCTR.IS=positive VAKBN.IS=positive YKBNK.IS=positive 269 ==> GARAN.IS=positive 266 confidence: 0.99
4	AKBNK.IS=negative ISCTR.IS=negative PGSUS.IS=negative SAHOL.IS=negative VAKBN.IS=negative YKBNK.IS=negative 265 ==> GARAN.IS=negative 262 confidence: 0.99
5	GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive TATGD.IS=positive YKBNK.IS=positive 283 ==> VAKBN.IS=positive 279 confidence: 0.99
6	AKBNK.IS=positive ALARK.IS=positive HALKB.IS=positive ISCTR.IS=positive VAKBN.IS=positive YKBNK.IS=positive 280 ==> GARAN.IS=positive 276 confidence: 0.99
7	AKBNK.IS=positive GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive TSKB.IS=positive 273 ==> VAKBN.IS=positive 269 confidence: 0.99
8	GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive TSKB.IS=positive YKBNK.IS=positive 272 ==> VAKBN.IS=positive 268 confidence: 0.99
9	AKBNK.IS=positive GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive TRKCM.IS=positive YKBNK.IS=positive 270 ==> VAKBN.IS=positive 266 confidence: 0.99
10	AKBNK.IS=positive ISCTR.IS=positive SAHOL.IS=positive THYAO.IS=positive VAKBN.IS=positive YKBNK.IS=positive 270 ==> GARAN.IS=positive 266 confidence: 0.99

We selected first, fourth and ninth rules form this table. And we removed two result companies that are GARANTI and VAKIFBANK from dataset. We select these rules because of theirs confidence level and providing companies diversity. We iterate this process until to reach 17 rules. We did not set minimum support level. Minimum support took 0.15 and 0.2 values. However, we set minimum confidence level that took 0.85, 0.90, 0.95 and 0.98 values and the created rules are as shown in Table 4.5 [4].

Table 4.5 Association rules using decrementing method

No	Association Rules
1	AKBNK.IS=positive HALKB.IS=positive ISCTR.IS=positive THYAO.IS=positive VAKBN.IS=positive YKBNK.IS=positive 295 ==> GARAN.IS=positive 292 confidence: 0.99
2	AKBNK.IS=negative ISCTR.IS=negative PGSUS.IS=negative SAHOL.IS=negative VAKBN.IS=negative YKBNK.IS=negative 265 ==> GARAN.IS=negative 262 confidence: 0.99
3	AKBNK.IS=positive GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive TRKCM.IS=positive YKBNK.IS=positive 270 ==> VAKBN.IS=positive 266 confidence: 0.99
4	HALKB.IS=positive ISCTR.IS=positive KCHOL.IS=positive PGSUS.IS=positive SAHOL.IS=positive THYAO.IS=positive YKBNK.IS=positive 203 ==> AKBNK.IS=positive 202 confidence: 1
5	AKBNK.IS=positive HALKB.IS=positive KARTN.IS=positive KCHOL.IS=positive THYAO.IS=positive YKBNK.IS=positive 204 ==> ISCTR.IS=positive 202 confidence: 0.99
6	BRSAN.IS=positive DEVA.IS=positive HALKB.IS=positive TATGD.IS=positive 207 ==> YKBNK.IS=positive 200 confidence: 0.97
7	AFYON.IS=negative HALKB.IS=negative IPEKE.IS=negative TATGD.IS=negative 204 ==> KOZAA.IS=negative 197 confidence: 0.97
8	BRSAN.IS=positive GOLTS.IS=positive KCHOL.IS=positive PGSUS.IS=positive 211 ==> THYAO.IS=positive 199 confidence: 0.94
9	ALARK.IS=negative HALKB.IS=negative KCHOL.IS=negative TTKOM.IS=negative 214 ==> SAHOL.IS=negative 201 confidence: 0.94
10	ALARK.IS=positive GOODY.IS=positive KARTN.IS=positive MGROS.IS=positive 211 ==> HALKB.IS=positive 196 confidence: 0.93
11	DEVA.IS=negative GOLTS.IS=negative KARTN.IS=negative PGSUS.IS=negative 220 ==> AFYON.IS=negative 203 confidence: 0.92
12	ALARK.IS=positive ECILC.IS=positive MGROS.IS=positive PETKM.IS=positive 215 ==> GOLTS.IS=positive 196 confidence: 0.91
13	DEVA.IS=negative ECZYT.IS=negative GOLTS.IS=negative GUBRF.IS=negative 222 ==> ECILC.IS=negative 200 confidence: 0.9
14	AKSA.IS=negative PETKM.IS=negative ZOREN.IS=negative 230 ==> TMSN.IS=negative 202 confidence: 0.88
15	BRSAN.IS=negative ISGYO.IS=negative ZOREN.IS=negative 227 ==> VESTL.IS=negative 198 confidence: 0.87
16	AKSEN.IS=negative ANACM.IS=negative GOODY.IS=negative 232 ==> PGSUS.IS=negative 202 confidence: 0.87
17	ANACM.IS=positive ARCLK.IS=positive PGSUS.IS=positive 234 ==> SISE.IS=positive 202 confidence: 0.86

Some graphics for stock prices for companies in decrementing method rules are as presented in Figure 4.3a and Figure 4.3b [4].



(a) Historical data graphics for stocks at rule1

(b) Historical data graphics for stocks at rule3

Figure 4.3 Historical data graphics with decreasing method [4]

In second method, we called it sectoral, we grouped stocks based on sectors (banks, energy companies, holdings, construction, automotive etc.) and also we created some groups that include dominant companies on BIST100. Our first group is based on banks and prepared data from these companies is shown in Table 4.6 [4].

Table 4.6 Prepared stock data from seven banks historical stock data close prices

Date	AKBNK	ALBRK	GARAN	HALKB	ISCTR	VAKBN	YKBNK
21.10.2013	same	same	same	same	same	same	same
22.10.2013	pos	pos	pos	pos	pos	pos	pos
23.10.2013	neg	neg	neg	neg	neg	neg	neg
24.10.2013	neg	pos	neg	neg	pos	same	same
25.10.2013	same	pos	pos	pos	pos	neg	pos
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
15.10.2018	pos	pos	pos	pos	pos	pos	pos
16.10.2018	neg	neg	neg	same	pos	pos	pos
17.10.2018	pos	pos	pos	pos	pos	pos	pos
18.10.2018	neg	neg	neg	neg	neg	neg	neg
19.10.2018	neg	neg	neg	same	neg	neg	neg

When we run Apriori algorithm on this data, we figure out association rules as shown in Table 4.7 [4].

Table 4.7 Generated association rules for banks

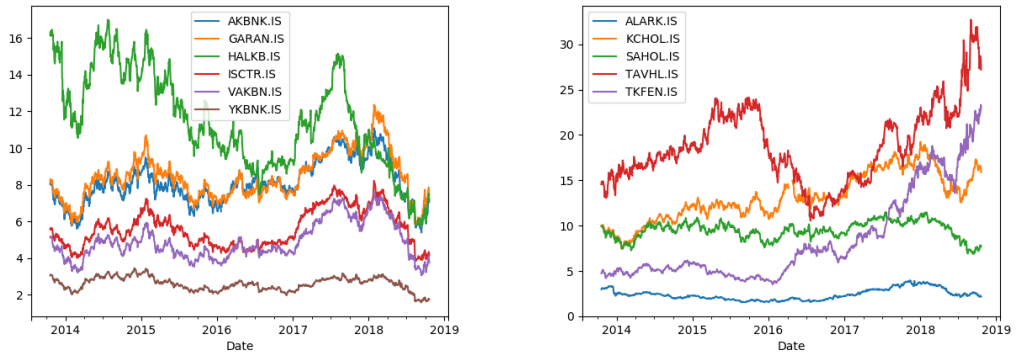
No	Weka Apriori -N 10 -T 0 -C 0.95 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1 Insts: 1305, Attrs: 7, Minimum support: 0.25 (326 instances), Minimum confidence: 0.95
1	AKBNK.IS=positive GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive YKBNK.IS=positive 347 ==> VAKBN.IS=positive 336 confidence: 0.97
2	AKBNK.IS=negative GARAN.IS=negative HALKB.IS=negative VAKBN.IS=negative 345 ==> YKBNK.IS=negative 334 confidence: 0.97
3	AKBNK.IS=positive HALKB.IS=positive ISCTR.IS=positive VAKBN.IS=positive YKBNK.IS=positive 348 ==> GARAN.IS=positive 336 confidence: 0.97
4	GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive YKBNK.IS=positive 370 ==> VAKBN.IS=positive 357 confidence: 0.96
5	AKBNK.IS=positive GARAN.IS=positive HALKB.IS=positive YKBNK.IS=positive 369 ==> VAKBN.IS=positive 355 confidence: 0.96
6	AKBNK.IS=positive GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive 373 ==> VAKBN.IS=positive 358 confidence: 0.96
7	AKBNK.IS=negative GARAN.IS=negative ISCTR.IS=negative VAKBN.IS=negative 368 ==> YKBNK.IS=negative 353 confidence: 0.96
8	AKBNK.IS=negative HALKB.IS=negative ISCTR.IS=negative VAKBN.IS=negative 341 ==> YKBNK.IS=negative 327 confidence: 0.96
9	AKBNK.IS=negative GARAN.IS=negative HALKB.IS=negative ISCTR.IS=negative 343 ==> YKBNK.IS=negative 328 confidence: 0.96
10	AKBNK.IS=positive HALKB.IS=positive ISCTR.IS=positive YKBNK.IS=positive 364 ==> VAKBN.IS=positive 348 confidence: 0.96

We selected first rule from this table. We made same process on other groups and we select some rules from these groups. We did not set minimum support level. Minimum support took 0.1, 0.15 and 0.2 values. However we set minimum confidence level that took 0.70, 0.80, 0.88, 0.90, 0.93, 0.95 and 0.97 values. Finally we created 15 rules that shown in Table 4.8 [4].

Table 4.8 Association rules using sectoral method

No	Association Rules
1	AKBNK.IS=positive GARAN.IS=positive HALKB.IS=positive ISCTR.IS=positive YKBNK.IS=positive 347 ==> VAKBN.IS=positive 336 confidence: 0.97
2	AKSEN.IS=negative ANELE.IS=negative GEREL.IS=negative ODAS.IS=negative 157 ==> ZOREN.IS=negative 134 confidence: 0.85
3	CCOLA.IS=positive TATGD.IS=positive ULKER.IS=positive 183 ==> MGROS.IS=positive 140 confidence: 0.77
4	ALARK.IS=positive KCHOL.IS=positive TAVHL.IS=positive TKFEN.IS=positive 157 ==> SAHOL.IS=positive 144 confidence: 0.92
5	AFYON.IS=negative KRDMD.IS=negative TKFEN.IS=negative 234 ==> GOLTS.IS=negative 197 confidence: 0.84
6	DOAS.IS=negative FROTO.IS=negative GOODY.IS=negative TTRAK.IS=negative 158 ==> TOASO.IS=negative 133 confidence: 0.84
7	ARCLK.IS=positive PGSUS.IS=positive 325 ==> THYAO.IS=positive 272 confidence: 0.84
8	EREGL.IS=positive GOLTS.IS=positive GUBRF.IS=positive MGROS.IS=positive SAHOL.IS=positive SISE.IS=positive THYAO.IS=positive 135 ==> VAKBN.IS=positive 133 confidence: 0.99
9	ECILC.IS=positive EREGL.IS=positive GOLTS.IS=positive MGROS.IS=positive SAHOL.IS=positive SISE.IS=positive THYAO.IS=positive 134 ==> VAKBN.IS=positive 132 confidence: 0.99
10	ECILC.IS=positive KARTN.IS=positive MGROS.IS=positive OTKAR.IS=positive SISE.IS=positive 145 ==> GOLTS.IS=positive 141 confidence: 0.97
11	EREGL.IS=positive GUBRF.IS=positive SISE.IS=positive THYAO.IS=positive ZOREN.IS=positive 143 ==> SAHOL.IS=positive 137 confidence: 0.96
12	EREGL.IS=positive GUBRF.IS=positive KARTN.IS=positive SAHOL.IS=positive TOASO.IS=positive 141 ==> THYAO.IS=positive 135 confidence: 0.96
13	EREGL.IS=positive KARTN.IS=positive TOASO.IS=positive ZOREN.IS=positive 145 ==> THYAO.IS=positive 137 confidence: 0.94
14	DGKLB.IS=negative ECILC.IS=negative GUBRF.IS=negative KARTN.IS=negative MGROS.IS=negative 152 ==> ZOREN.IS=negative 136 confidence: 0.89
15	DGKLB.IS=negative GUBRF.IS=negative KARTN.IS=negative MGROS.IS=negative ZOREN.IS=negative 154 ==> ECILC.IS=negative 136 confidence: 0.88

Some graphics for stock prices for companies in sectoral method rules are as seen in Figure 4.4a and Figure 4.4b [4].



(a) Historical data graphics for stocks at rule1

(b) Historical data graphics for stocks at rule4

Figure 4.4 Historical data graphics with sectoral method [4]

The main objective of this phase is to figure out association rules between BIST100 stocks using Apriori algorithm. We used two methods to select stocks. In first method we started with all 87 stocks, found association rules between them and removed two stocks at the best two rules and go on to find new association rules. In second method we divided stocks on sectoral base sets, and determined association rules on these sets [4].

When the results are examined, we showed that there are strong relationships between stocks in the dataset. We found that sectoral based stocks generally move together or they have same envelope for their historical price data. And also stocks at rules that have higher confidence than 0.90 move together or they have same envelope for their historical price data. We consider that association rule mining is beneficial to find hidden patterns in large financial dataset. We wrote a proceeding taking advantage of our study results and we sent it to ISMSIT2019. It was accepted and we presented our proceeding at ISMSIT2019, and also it was published in IEEE Explore [4].

Association rules were created with one minimum support level and minimum confidence level. In future works, lift values may be used to define association rules. We used positive, negative and same tags to define movement direction for all stocks data. Different tags can be implemented for stocks movement direction. We couldn't add the BIST100 index historical data into association rule mining process because lack of BIST100 index historical data. BIST 100 index historical data can be added to association rule mining process in future works. It is beneficial and also necessary for

feature selection process.

Using association rules, we determined the keywords we will use in the tweet crawl process as '#AFYON', '#AKBNK', '#AKSA', '#AKSEN', '#ALARK', '#ANACM', '#ARCLK', '#BRSAN', '#DEVA', '#ECILC', '#ECZYT', '#GARAN', '#GOLTS', '#GOODY', '#GUBRF', '#HALKB', '#IPEKE', '#ISCTR', '#ISGYO', '#KARTN', '#KCHOL', '#KOZAA', '#MGROS', '#PETKM', '#PGSUS', '#SAHOL', '#SISE', '#TATGD', '#THYAO', '#TMSN', '#TRKCM', '#TTKOM', '#VAKBN', '#VESTL', '#YKBNK', '#ZOREN'.

4.2 Twitter: Crawl, Create Datasets and Tagging

In this phase of our works; we focused on tweet crawling, create datasets and web based tweet tagging program development. We wrote a python code that uses TwitterAPI and collects financial tweets for given keywords on real time. We created our own datasets ,which are fed by our tweet crawling program, on MongoDB and MySQL databases. And we wrote a web based tweet tagging program to tag tweets as positive, negative, neutral, and irrelevant, using java.

4.2.1 Twitter Data Crawler

We wrote a tweet collection program in this phase. It collects tweets in real time for given keywords and stores these tweets into MySQL database. The keywords are selected from generated association rules by us. Listing B.1 in Appendix B contains our python code.

4.2.2 Tweet Dataset

We implemented a datasets that includes tweet id, tweet text, tweet creation date, topic used for sentiment analysis, and sentiment label (positive, negative, neutral, or irrelevant) columns. We created a **thesis schema** on MySQL database and two tables that are **"TweetSentiments"** and **"Twitter"**.

Twitter table has four columns. Those are **"tweet_id"**, **"screen_name"**, **"created_at"** and **"tekst"**. TweetSentiments table has also four columns that are

"id", **"topic"**, **"sentiment"** and **"tweet_id"**. These data tables are as shown in Table 4.9 and Table 4.10.

Table 4.9 Twitter data table sample

tweet_id	screen_name	created_at	tekst
109833092147069	Has******	20.02.2019 21:17	RT @EYMBorsa: #KOZAA nında 2018 yıl sonu karının 450-550 mılyon tl arası olmasını beklıyoruz.bakmayın sız vuk 35 mılyona her nekadar yaba...
109833088464470	AER*****	20.02.2019 21:17	RT @borsatilkisi: düşüşü bitmiş hisseler formülü tarama sonucu ytd #alka #kozaa #mipaz https://t.co/qTPO6GZwGG
109833088124314	Hak*****	20.02.2019 21:17	@sputnik_TR #xu100 #bist #borsa #krdmd #garan #ekgyo #asels #thyao Borsa düşecek bahane arıyordu iyi oldu.
109833087332427	Bod*****	20.02.2019 21:17	#THYAO(Alıntı Ytd.) https://t.co/OL9qHGHlrB
109471533907678	Egeli****	10.02.2019 21:50	RT @af_38: Haftalık takip listem; #alctl 6.50 #kuyas 1.37 #tuclk 2.28 #tknosa 3.12 #klgyo 1.98 #usak 1.08 #Glyho 3.81 #dgate...
109471486598024	Bek****	10.02.2019 21:48	#XU100 #GARAN #HALKB #BIST #ULKER #THYAO #AKBNK #TAVHL #TCELL #ISGSY #ALARK #ULKER #YATAS #OTKAR #USDTRY #EURTRY #GBPTRY #KOZAA #THYAO #AKBNK #YKBNK #ISCTR #ARCLK #SAHOL #TCELL #ODAS #XBANK #XU030 #ALARK TARAMA LİSTEM YTD https://t.co/rH79Ffqpw5
109471378582947	Sbal***	10.02.2019 21:44	RT @acoskun05: 1 aylık #petkm takası çok hoş ilerisi için ümit vadediyor net alıcı fonlar ve yabancılar. https://t.co/1rQWNx8KVB
109471373158873	Durh***	10.02.2019 21:44	RT @borsatilkisi: @kara2173 #kozaa hissede kar satışlarından sonra toparlanma çabaları görülüyor dirençler 6,00-6,20-6,50 destekler 5,88-5,...
109471350040707	Durh***	10.02.2019 21:43	RT @borsatilkisi: @ahmetpekcicekli #zoren 1,33 orta bolinger destek noktasından tekrar yükselmeye baslamıs kalıcı yükselişler için 1,37 nin...

Table 4.10 Tweetsentiments data table sample

id	topic	sentiment	tweet_id
23504	#KOZAA	POSITIVE	109833092147069
23503	#KOZAA	POSITIVE	109833088464470
23502	#THYAO	NEGATIVE	109833088124314
23501	#GARAN	NEGATIVE	109833088124314
23500	#THYAO	IRRELEVANT	109833087332427
23499	#AFYON	POSITIVE	109471533907678
23498	#YKBNK	IRRELEVANT	109471486598024
23497	#THYAO	IRRELEVANT	109471486598024
23496	#SAHOL	IRRELEVANT	109471486598024
23495	#KOZAA	IRRELEVANT	109471486598024

Twitter table includes tweets that are collected by using TwitterAPI and TweetSentiments table includes collected tweets topic and sentiment information. Initially, sentiment information is empty, but it is filled manually by our tagging program up to tweet sentiments. This datasets will be used for sentiment analysis at next phases. We created a twitterdb database on MongoDB. It includes collected tweets in a JSON format.

4.2.3 Tweet Tagging

We need tweet tagging to achieve tweet sentiment analysis. We implemented a web based java program for tweet tagging. These tagged tweets are used as train and test datasets for sentiment analysis. Some screen images of this program shown as Figure 4.5, Figure 4.6, Figure 4.7, Figure 4.8 and Figure 4.9.

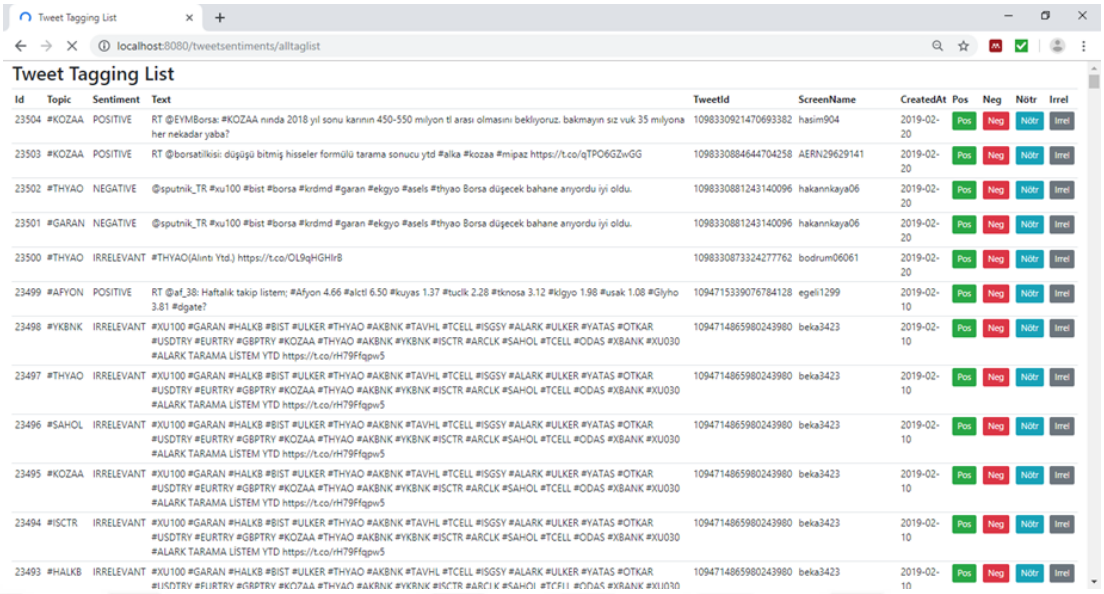


Figure 4.5 tweetsentiments/alltaglist page

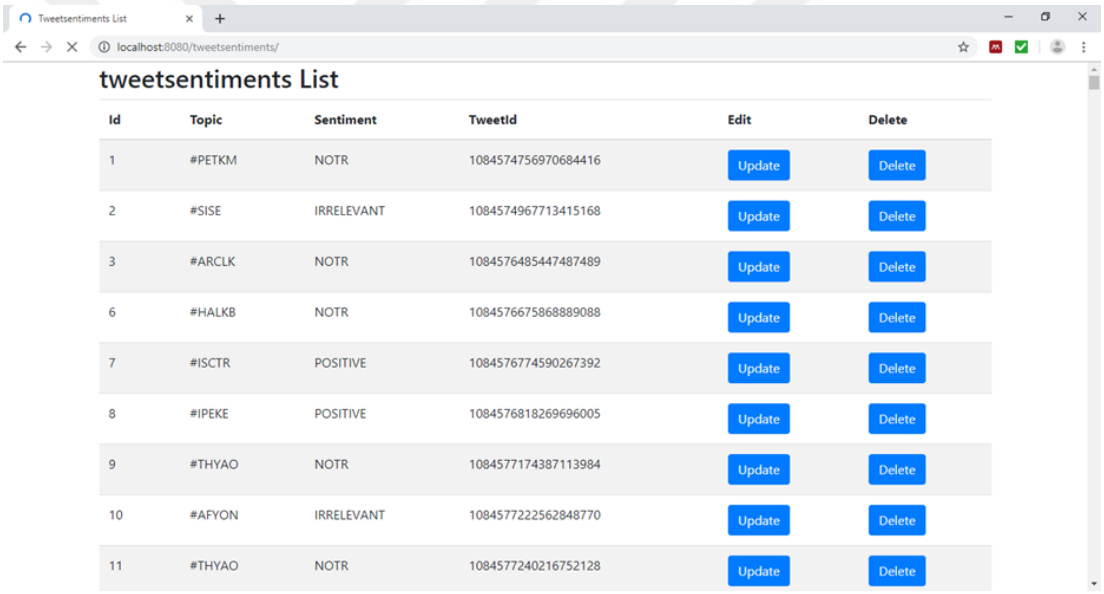


Figure 4.6 tweetsentiments/ page

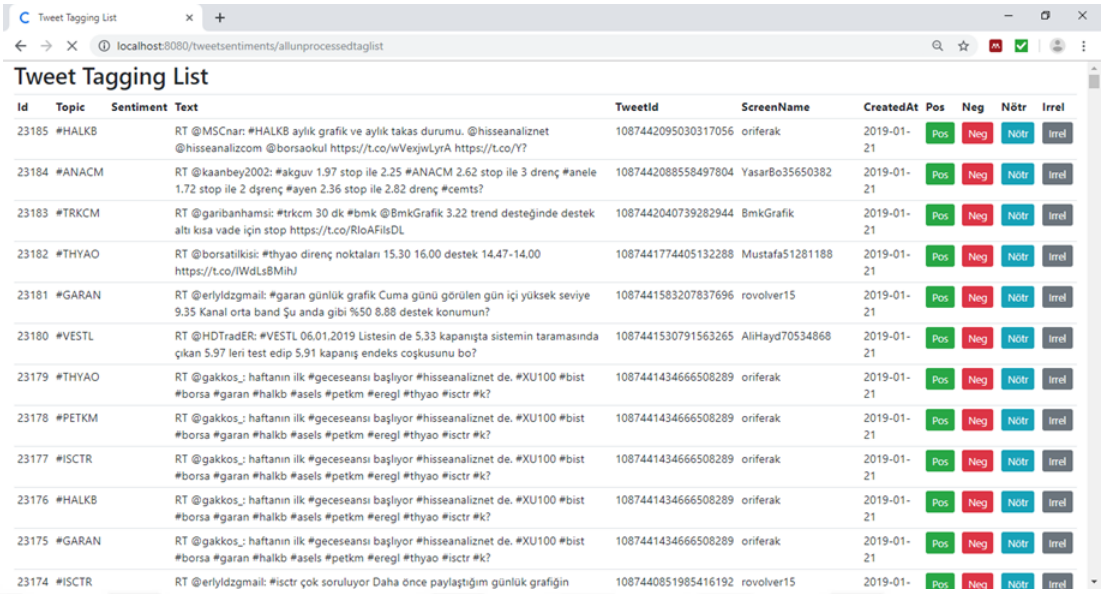


Figure 4.7 tweetsentiments/allunprocessedtaglist page

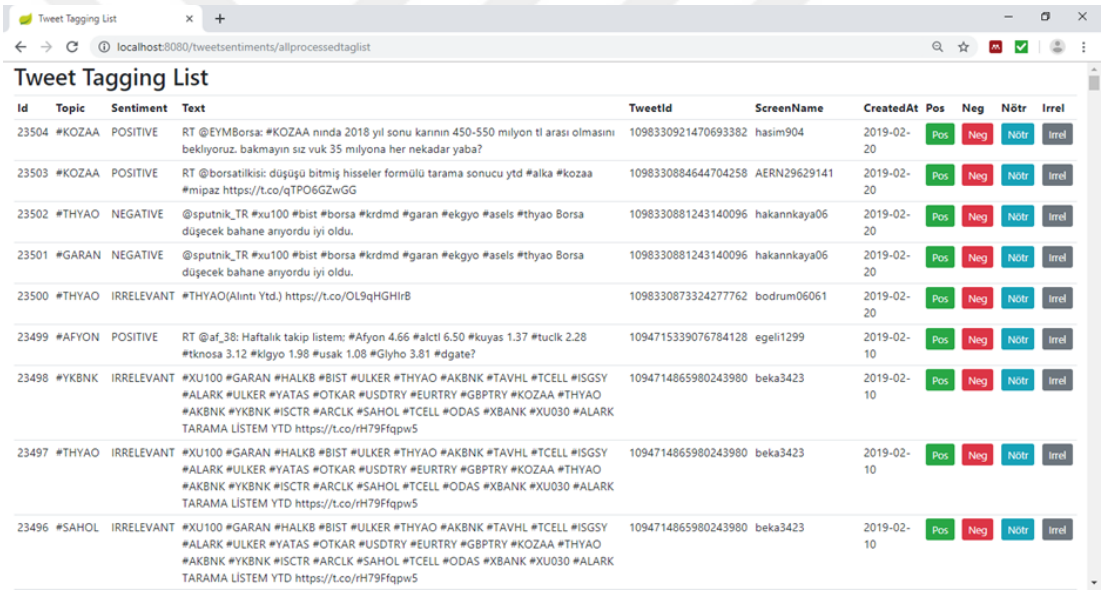


Figure 4.8 tweetsentiments/allprocessedtaglist page

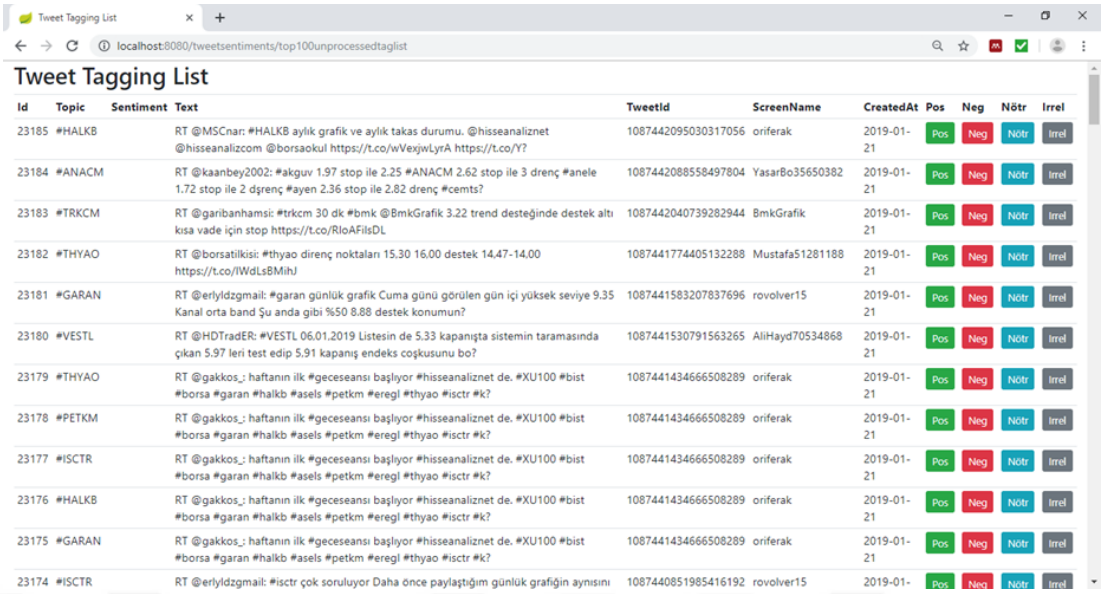


Figure 4.9 tweetsentiments/top100unprocessedtaglist page

With these pages, we can add, update or change tweet tags. We tagged all the collected tweets with these pages. This process provided us a new dataset for sentiment analysis on financial tweets.

4.2.4 Result and Discussions of Twitter: Crawl, Create Datasets and Tagging

In this section, we developed the tweet crawler, tweet tagging process, and datasets required for tweet classification and feature extraction.

4.3 Deep Learning Based Sentiment Analysis on Turkish Financial Twitter Data

Tweet sentiment analysis is an important part of our work. So in this phase studies we want to apply some deep learning methods to our data sets which are created and tagged by ourselves. And we report the performances of these methods.

We used Neural Network, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and GRU-CNN algorithms together with word embedding and fastText's pre-trained word embedding. These models were used for binary and multi-class classifications. While the **softmax** function was used in the output layer for **multi-class** classification, the **sigmoid** function was used in the output layer for **binary** classification. For all models, 5-fold cross validation is used

for training and testing processes. We wrote some python codes for deep learning and text pre-processing.

In this phase, we worked on a newly created and tagged by us Turkish tweet dataset that includes 2313 tweets. The dataset has 992 POSITIVE, 629 NEGATIVE and 691 NOTR labelled tweets. We created two datasets: binary ("0-NEGATIVE", "1-POSITIVE") and multi-class ("NEGATIVE", "POSITIVE" and "NEUTRAL") datasets. Datasets data distributions are shown in Figure 4.10 and sample labeled tweet are shown in Table 4.11.

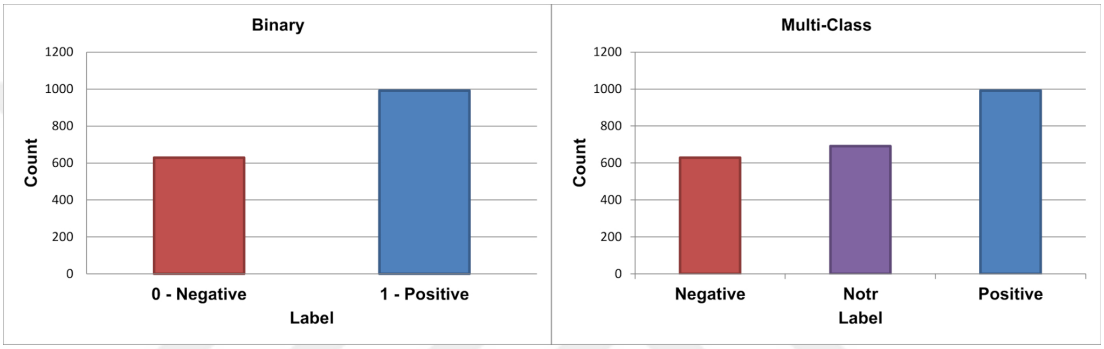


Figure 4.10 Binary and multi-class datasets

Table 4.11 Labeled tweets samples

Text	Label
#Halkb için bunu yazdığımızda hisse 805 lerdeydi aksam kapanişi 781 yaparak yanıltmadı..Garanyatırımla yemliyorlar milleti.	NEGATIVE
#Halkb için bunu yazdığımızda hisse 805 lerdeydi aksam kapanişi 781 yaparak yanıltmadı..Garanyatırımla yemliyorlar milleti. Diğerlerindede yem olarak kullanıları yazmayayı taktik değiştirmesinler.#gerel #Garan #Akbnk #Thyao #Petkm #Ttkom #Sasa	NEGATIVE
#klnma #iztar #avod #tactr #adese #vakfn #oyayo #thyao #krdmd #akbnk #garnt #bmelk -panik yapanlar dokuluyo ve malını alıyorlar panikciler gitsin yön yukarı #memsa -nedendir anlamam bundada bi panik havası dostlarım sakin olun patron olsanız bu fiyattanmi satarsınız..	NEGATIVE
#klnma #iztar #avod #tactr #adese #vakfn #oyayo #thyao #krdmd #akbnk #garnt #bmelk -panikcileri dokuyo siz sakin olun ve bekleyin #memsa-Ne dediysek o satan kuduruyo satın satın diye nara atiyo kimseye kanmayın. #sekur 3 5 derken bi atak yapar..	NEGATIVE
#kozaa mavi kanal üzerine atamadı , alt destek bandını bekliyorum ya da ara destek çalıştırıp trend verirse olabilir.	NEGATIVE

4.3.1 Tweet Pre-processing Phase

Before using tweets our neural network models, it needs text pre-processing operation. Tweet pre-processing includes;

- Removing unnecessary parts of tweets (external links and user names (signified with @sign), URL (http://...), stop words, #tags, retweets (starts with “RT”), punctuations, unnecessary whitespaces etc.),
- Transforming characters to lowercase,
- Removing numbers,
- And correction of spelling/writing errors (normalization) or restore popular abbreviations to their corresponding originals (e.g., mrb to merhaba). ITU Turkish NLP Web Service API [5] is used for normalization process.

We developed a tweet pre-processing program with python, which processes the tweets as shown in Figure 4.11.

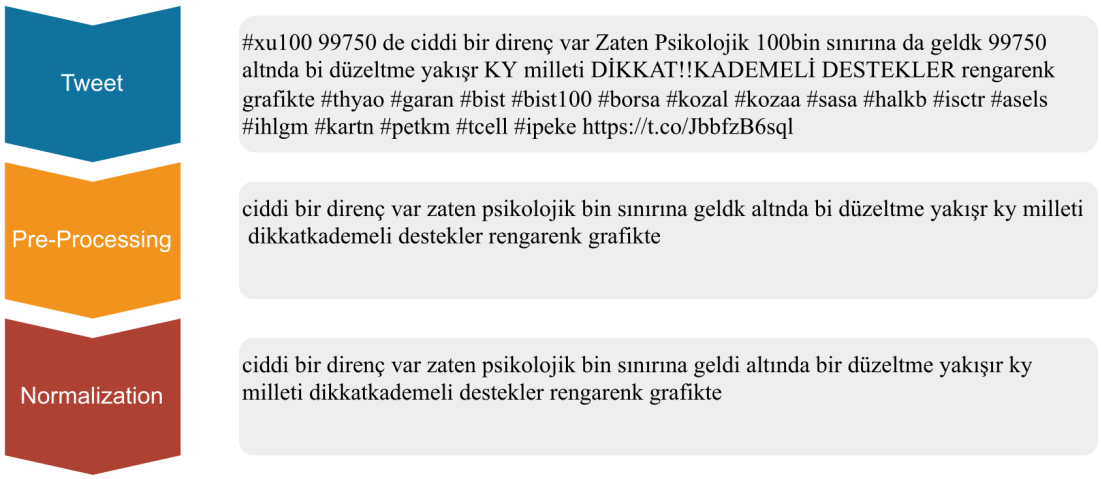


Figure 4.11 Pre-processing and normalization steps of tweets

After pre-processing and normalization we also manually checked the normalization process results and we created a tap separated text file as shown in Table 4.12.

Table 4.12 Sample tweet data set after pre-processing, normalization and manually checked.

Text	Label
bunu yazdığımızda hisselerdeydi akşam kapanışı yaparak yanıltmadigaranyatırımla yemliyorlar milleti	0
bunu yazdığımızda hisselerdeydi akşam kapanışı yaparak yanıltmadigaranyatırımla yemliyorlar milleti diğerlerinde de yem olarak kullanılanı yazmamayı taktik değiştirmesinler	0
panik yapanlar dökülüyor malını alıyorlar panikçiler gitsin yön yukarı nedendir anlamam bunda dabi panik havası dostlarım sakin olun patron olsanız fiyattan mı satarsınız	0
panikçileri döküyor sakin olun bekleyin dediysek satan kuduruyor satın satın nara atıyor kimseye kanmayın derken bir atak yapar	0
mavi kanal üzerine atamadı alt destek bandını bekliyorum ara destek çalıştırıp trend verirse olabilir	0

4.3.2 Classifications

In these phase we used Simple Neural Network, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and GRU-CNN algorithms together with word embedding and fastText's pre-trained word embedding and observed theirs train and test results.

We provide word embedding for our models' input. We can train our word embedding, during the train of our models or we can use pre-trained word embedding in our models.

Net2Vis [44] was used for our deep learning models visualizations.

4.3.2.1 Simple Neural Network Model

A binary neural network model using word embedding was designed with the Python code. Listing C.1 in Appendix C contains this code snippet. For multi-class binary neural network only difference is in the output layer which includes three neurons.

Figure 4.12 shows our simple neural network model for binary classification [44].

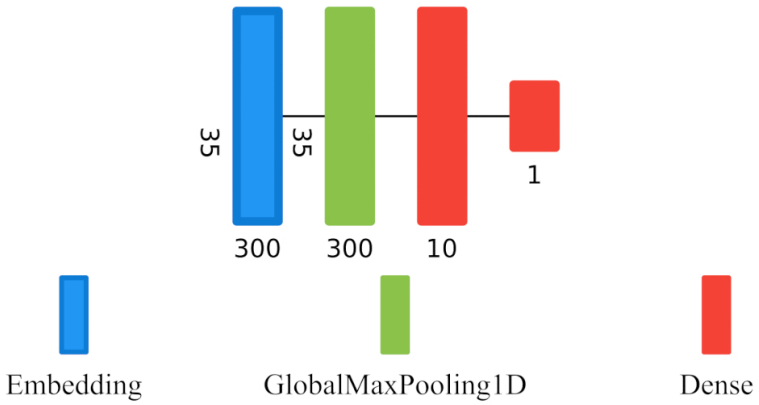


Figure 4.12 Binary neural network model

5-fold cross validation is used for training and testing process. Our training and testing accuracy results with word embedding for binary classification on binary dataset are as shown in Table 4.13;

Table 4.13 Training and testing accuracies for binary neural network model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Word embedding	Train	100
Binary	Word embedding	Test	80.25

The loss and accuracy graphs for our training and testing data are as shown in Figure 4.13.

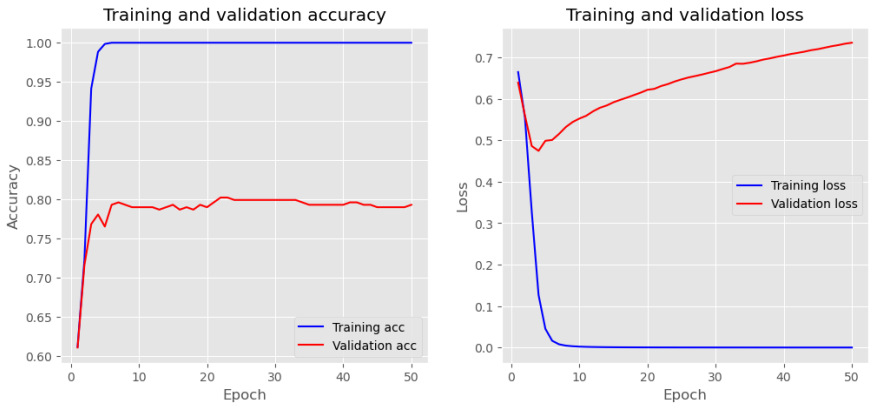
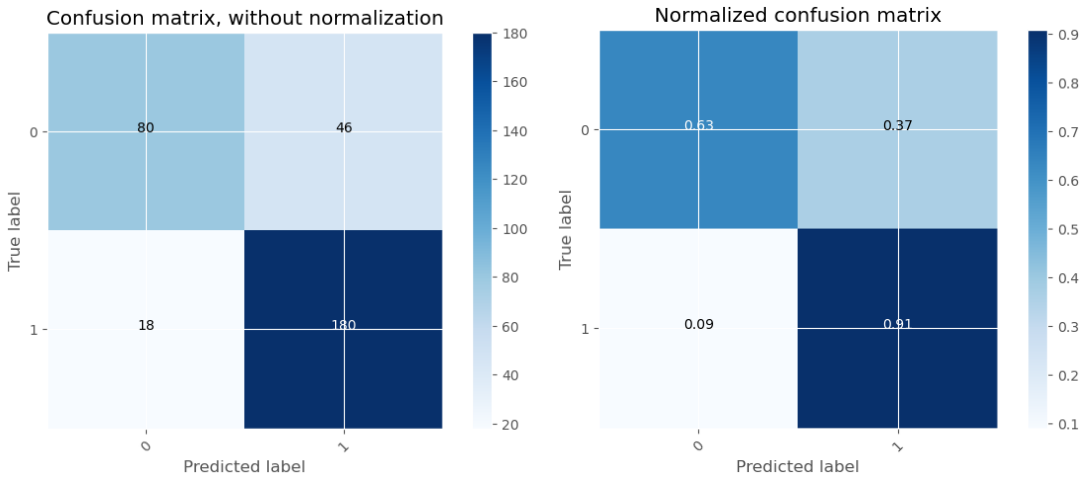


Figure 4.13 The loss and accuracy graphs for binary neural network model with word embedding

And the confusion matrices for test results is as shown in Figure 4.14.



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.14 The confusion matrices of test results for binary neural network model with word embedding

When we used fastText's pre-trained word embedding, our results are shown in below Table 4.14, Figure 4.15 and Figure 4.16.

Table 4.14 Training and testing accuracies for binary neural network model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Pre-trained word embedding	Train	100
Binary	Pre-trained word embedding	Test	79.32

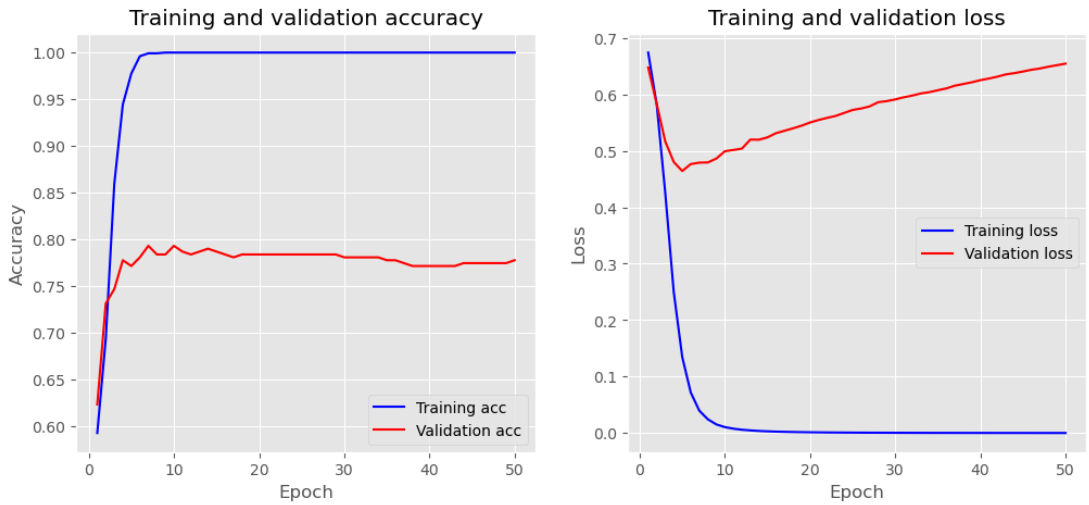
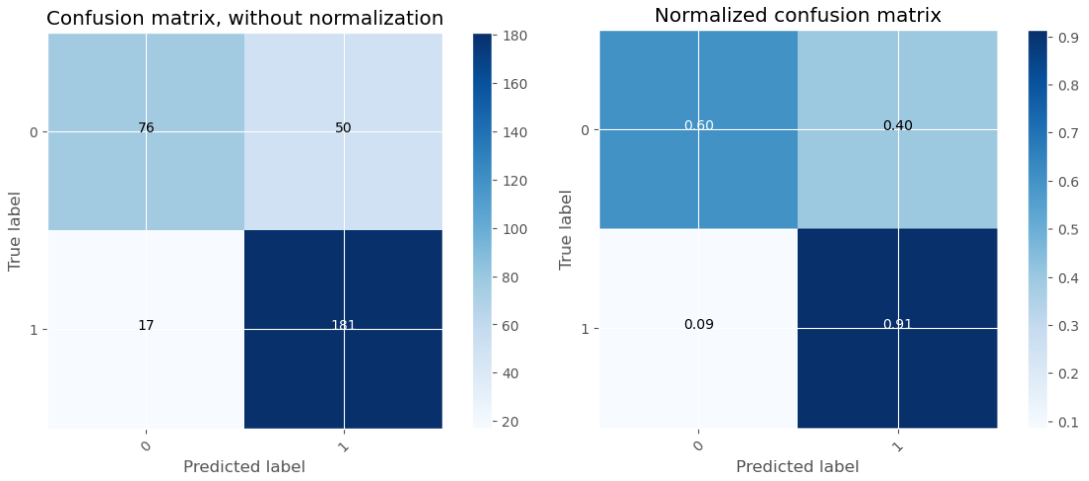


Figure 4.15 The loss and accuracy graphs for binary neural network model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.16 The confusion matrices of test results for binary neural network model with pre-trained word embedding

When we used multi-classes neural network model, our training and testing accuracy results with word embedding for multi-classes classification on multi-classes data set are shown in Table 4.15, Figure 4.17 and Figure 4.18;

Table 4.15 Training and testing accuracies for multi-classes neural network model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Word embedding	Train	99.57
Multi-classes	Word embedding	Test	63.85

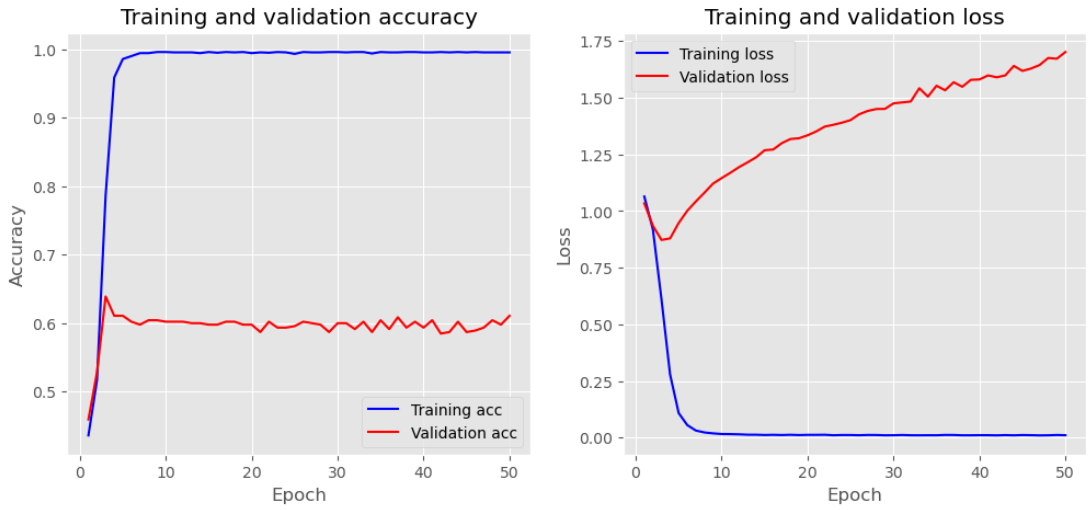
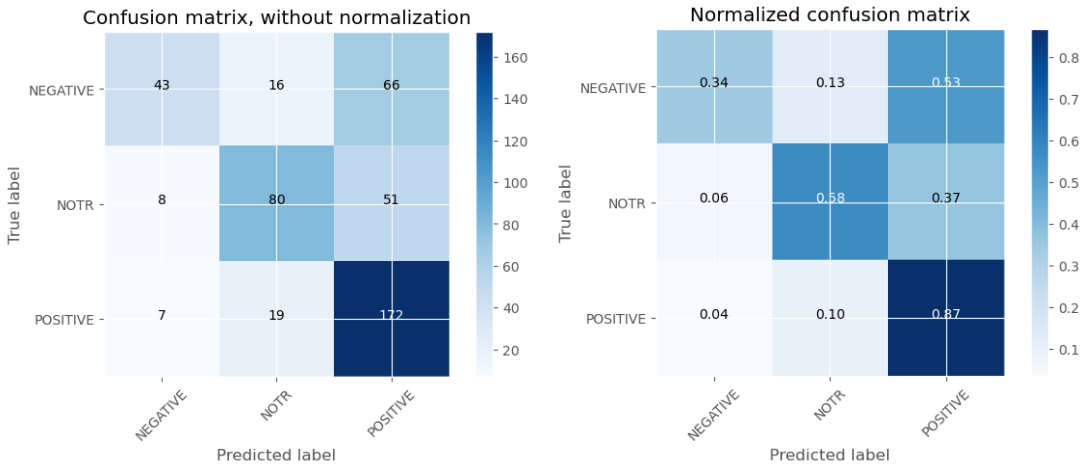


Figure 4.17 The loss and accuracy graphs for multi-classes neural network model with word embedding



(a) Confusion matrix **(b)** Normalized confusion matrix

Figure 4.18 The confusion matrices of test results for multi-classes neural network model with word embedding

When we used fastText's pre-trained word embedding, our results are shown in Table 4.16, Figure 4.19 and Figure 4.20.

Table 4.16 Training and testing accuracies for multi-classes neural network model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Pre-trained word embedding	Train	99.51
Multi-classes	Pre-trained Word embedding	Test	65.23

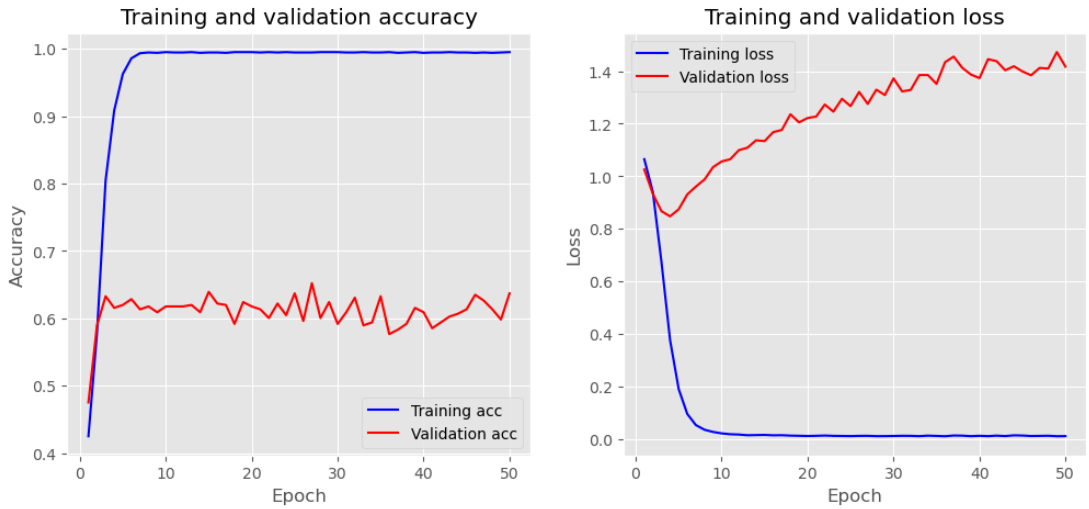
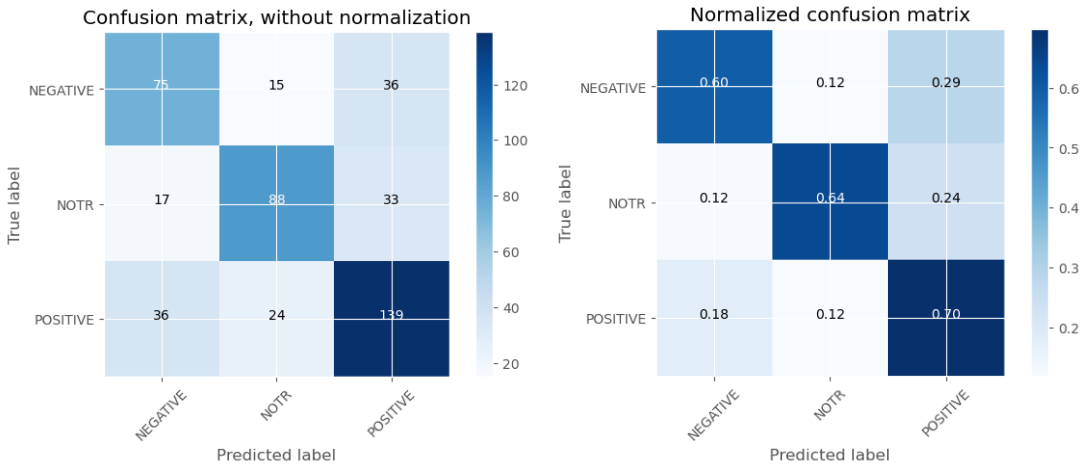


Figure 4.19 The loss and accuracy graphs for multi-classes neural network model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.20 The confusion matrices of test results for multi-classes neural network model with pre-trained word embedding

4.3.2.2 Convolutional Neural Network (CNN) Model

A convolutional neural network model using word embedding was designed with the python code snippet which is included in Listing C.2 in Appendix C. For multi-class CNN model has only one difference at its the output layer which includes three neurons.

Figure 4.21 shows our convolutional neural network model for binary classification.

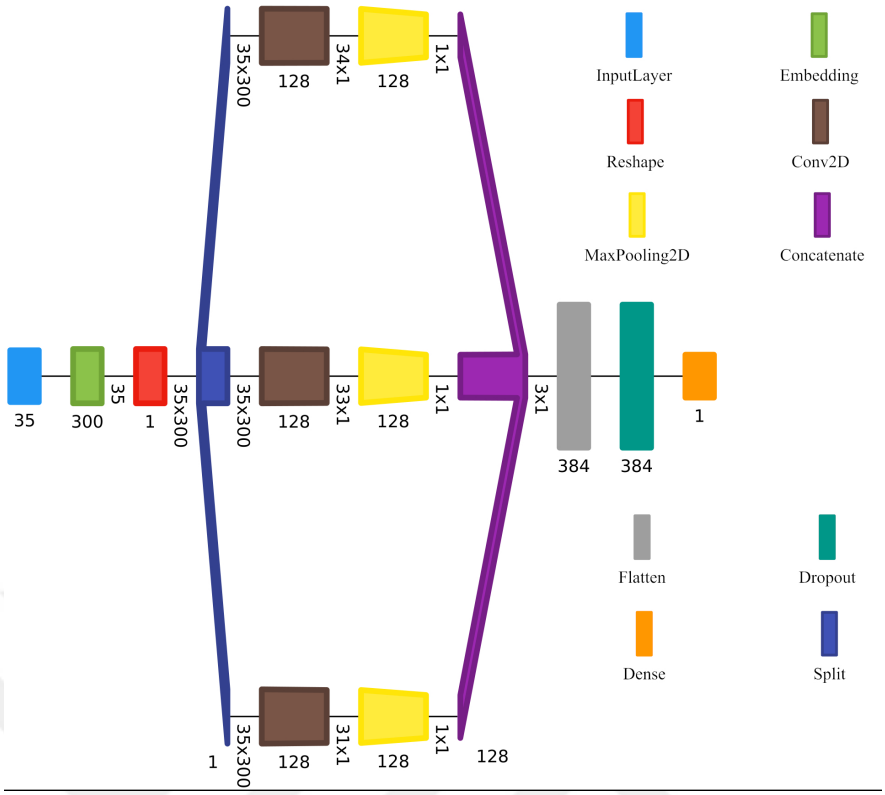


Figure 4.21 Binary convolutional neural network model

5-fold cross validation is used for training and testing process. Our training and testing accuracy results with word embedding for binary classification on binary dataset are as shown in Table 4.17, Figure 4.22 and Figure 4.23;

Table 4.17 Training and testing accuracies for binary convolutional neural network model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Word embedding	Train	100
Binary	Word embedding	Test	77.23

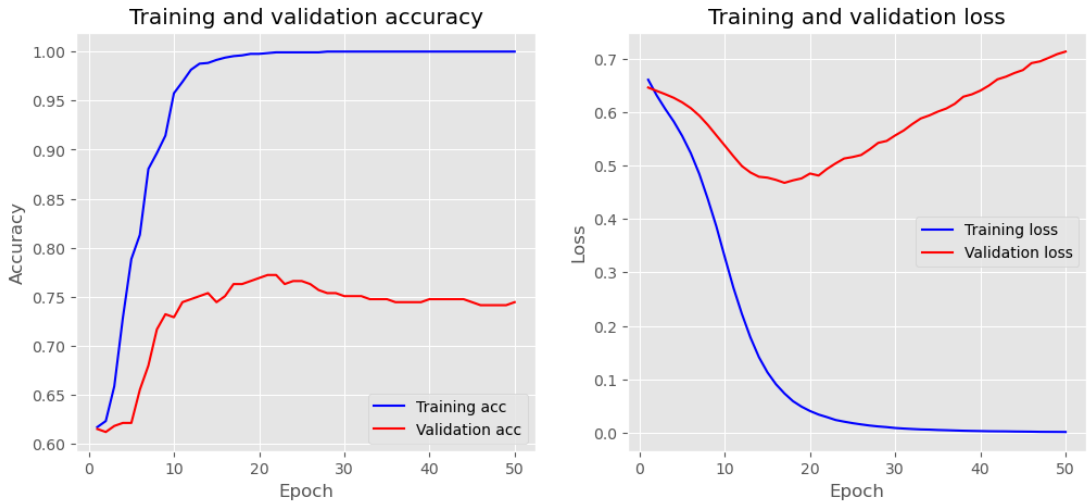
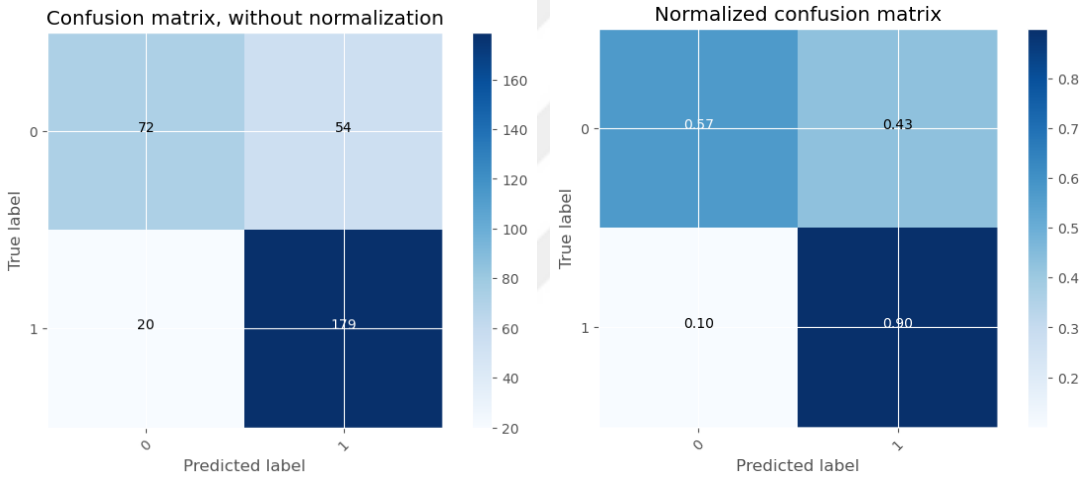


Figure 4.22 The loss and accuracy graphs for binary convolutional neural network model with word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.23 The confusion matrices of test results for binary convolutional neural network model with word embedding

When we used fastText's pre-trained word embedding, our results are as shown in Table 4.18, Figure 4.24 and Figure 4.25.

Table 4.18 Training and testing accuracies for binary convolutional neural network model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Pre-trained word embedding	Train	100
Binary	Pre-trained word embedding	Test	83.02

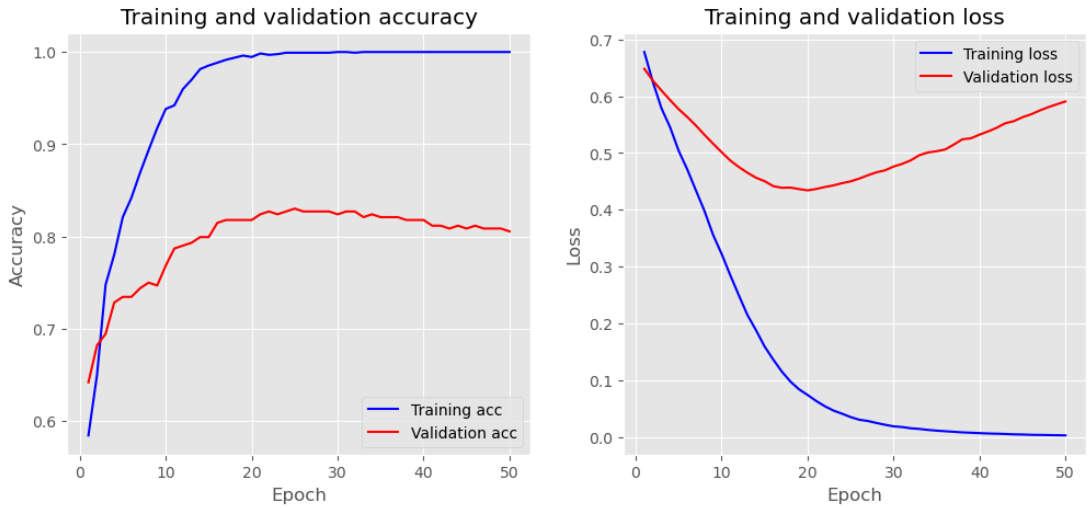
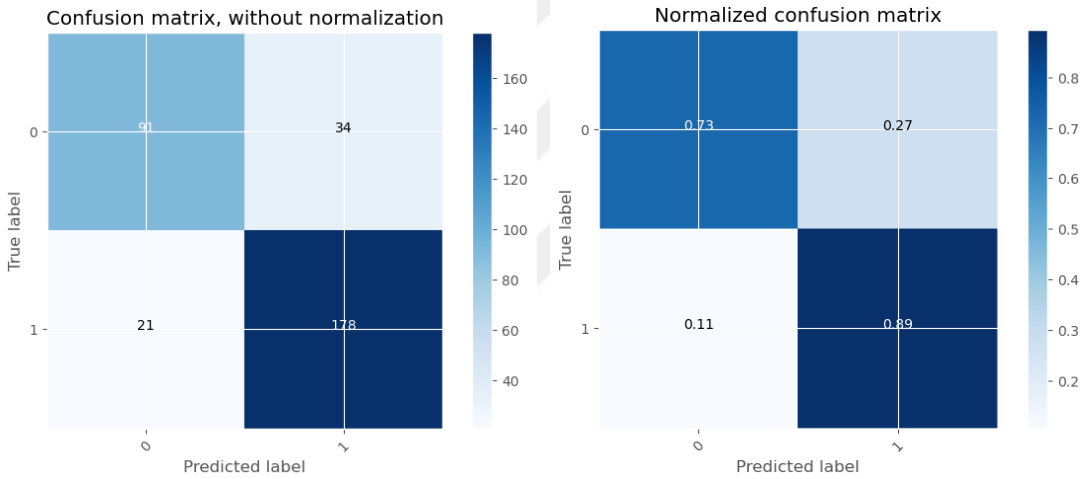


Figure 4.24 The loss and accuracy graphs for binary convolutional neural network model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.25 The confusion matrices of test results for binary convolutional neural network model with pre-trained word embedding

When we used multi-classes convolutional neural network model, our training and testing accuracy results with word embedding for multi-classes classification on multi-classes data set are shown in Table 4.19, Figure 4.26 and Figure 4.27;

Table 4.19 Training and testing accuracies for multi-classes convolutional neural network model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Word embedding	Train	99.73
Multi-classes	Word embedding	Test	63.71

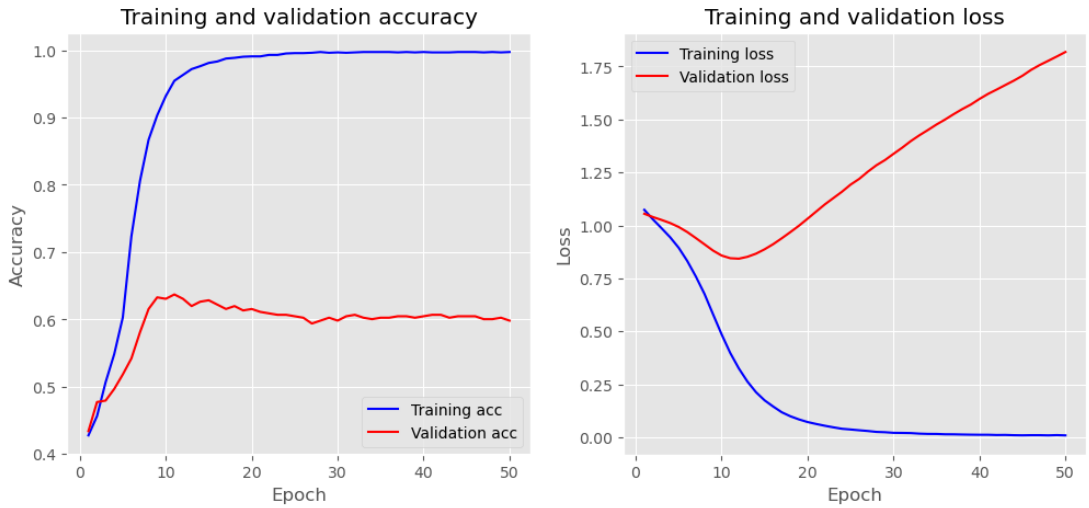
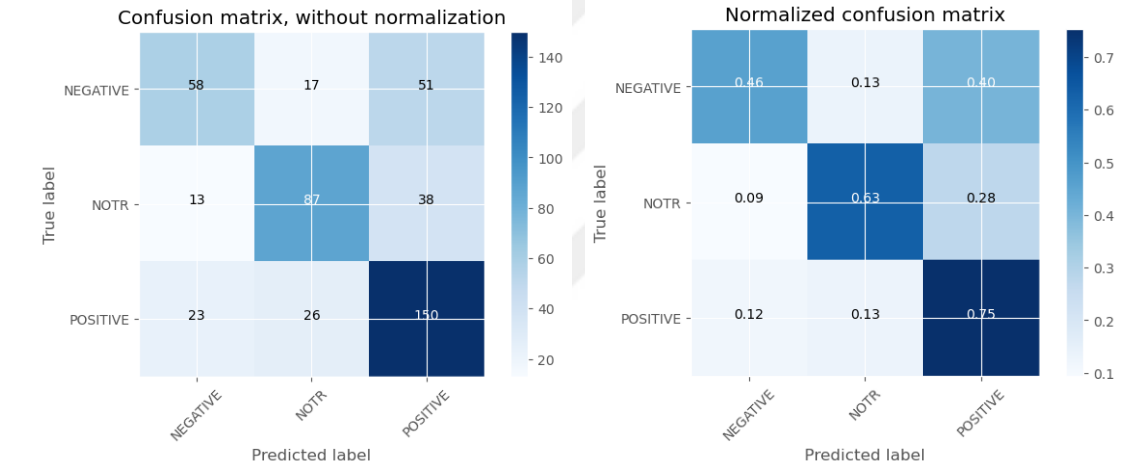


Figure 4.26 The loss and accuracy graphs for multi-classes convolutional neural network model with word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

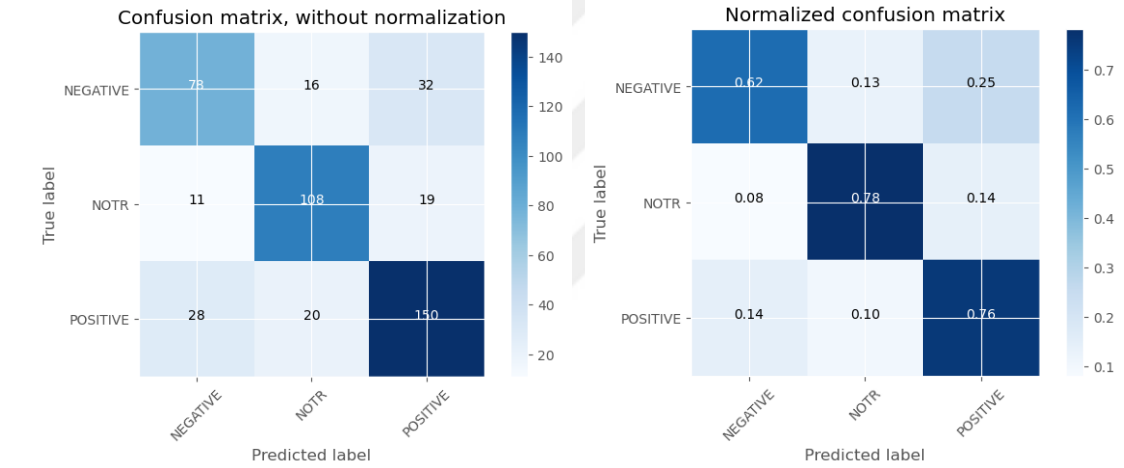
Figure 4.27 The confusion matrices of test results for multi-classes convolutional neural network model with word embedding

When we used fastText's pre-trained word embedding, our results are as shown in Table 4.20, Figure 4.28 and Figure 4.29.

Table 4.20 Training and testing accuracies for multi-classes convolutional neural network model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Pre-trained word embedding	Train	99.73
Multi-classes	Pre-trained word embedding	Test	72.73

Figure 4.28 The loss and accuracy graphs for multi-classes convolutional neural network model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.29 The confusion matrices of test results for multi-classes convolutional neural network model with pre-trained word embedding

4.3.2.3 Long Short-Term Memory (LSTM) Model

A long short-term memory model using word embedding was designed with python code snippet which is included in Listing C.3 in Appendix C. For multi-class LSTM model has only one difference at its the output layer which includes three neurons.

Figure 4.30 shows our long short-term memory model for binary classification.

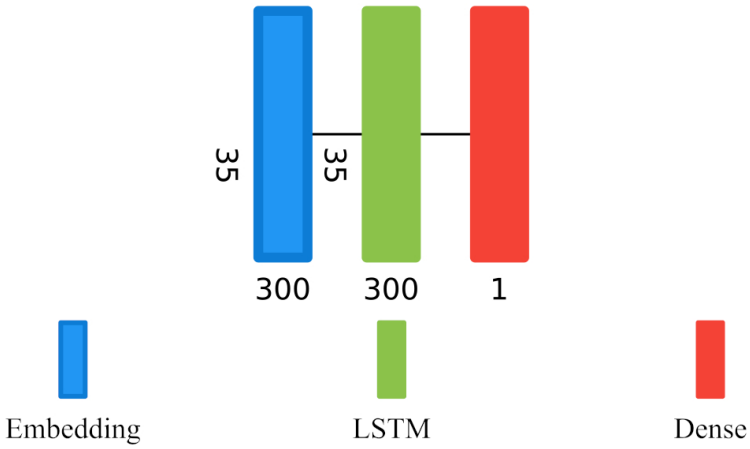


Figure 4.30 Binary long short-term memory model

5-fold cross validation is used for training and testing process. Our training and testing accuracy results with word embedding for binary classification on binary dataset are as shown in Table 4.21, Figure 4.31 and Figure 4.32;

Table 4.21 Training and testing accuracies for binary long short-term memory model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Word embedding	Train	98.30
Binary	Word embedding	Test	74.69

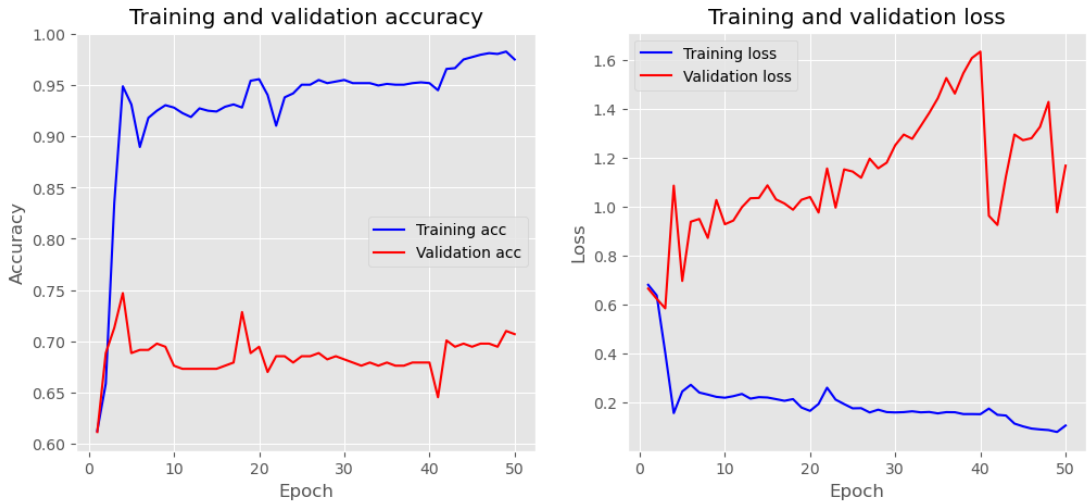
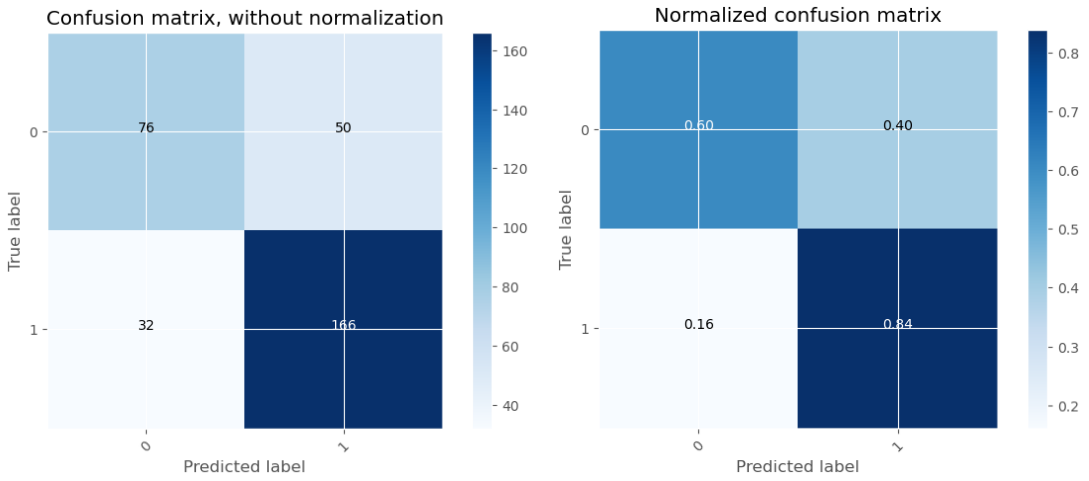


Figure 4.31 The loss and accuracy graphs for binary long short-term memory model with word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.32 The confusion matrices of test results for binary long short-term memory model with word embedding

When we used fastText's pre-trained word embedding, our results are as shown in Table 4.22, Figure 4.33 and Figure 4.34.

Table 4.22 Training and testing accuracies for binary long short-term memory model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Pre-trained word embedding	Train	100
Binary	Pre-trained word embedding	Test	79.32

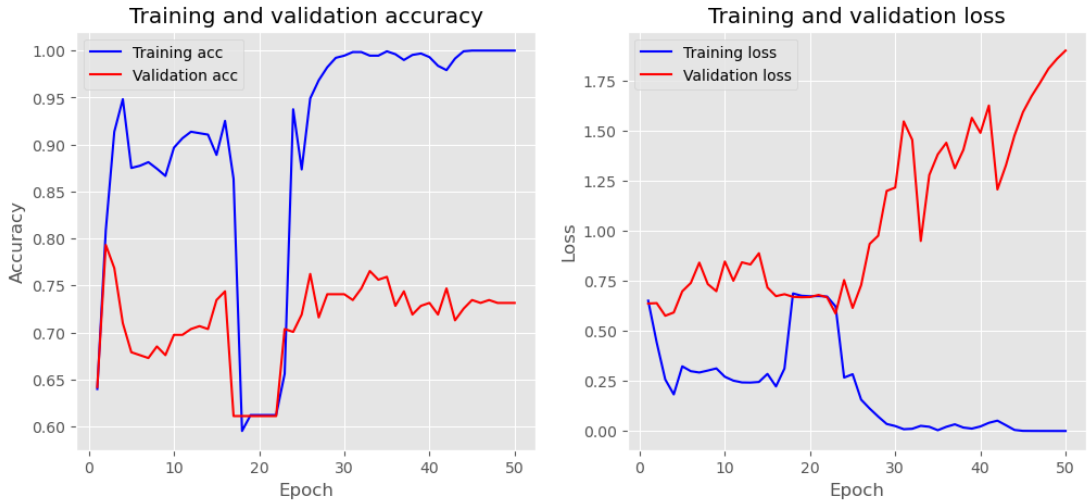
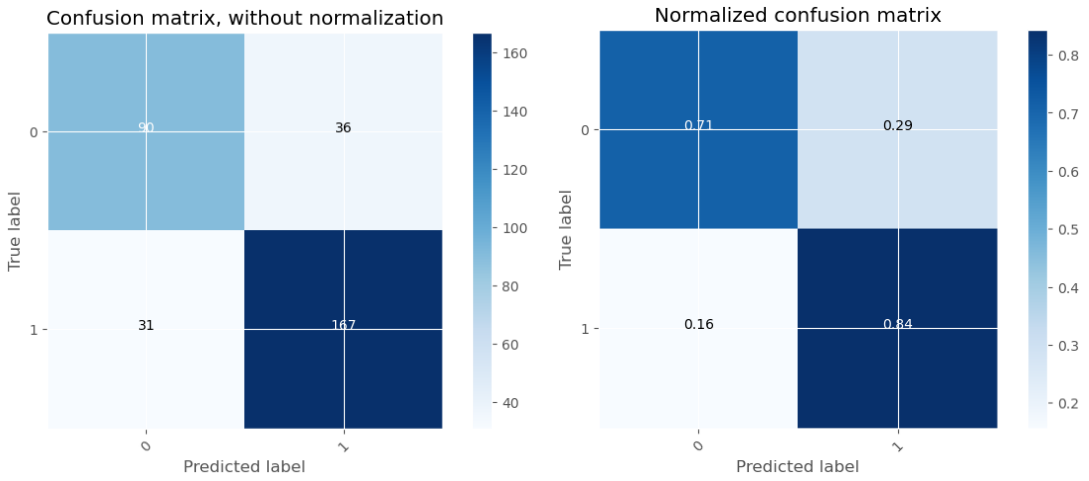


Figure 4.33 The loss and accuracy graphs for binary long short-term memory model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.34 The confusion matrices of test results for binary long short-term memory model with pre-trained word embedding

When we used multi-classes long short-term memory model, our training and testing accuracy results with word embedding for multi-classes classification on multi-classes data set are shown in Table 4.23, Figure 4.35 and Figure 4.36;

Table 4.23 Training and testing accuracies for multi-classes long short-term memory model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Word embedding	Train	99.24
Multi-classes	Word embedding	Test	59.52

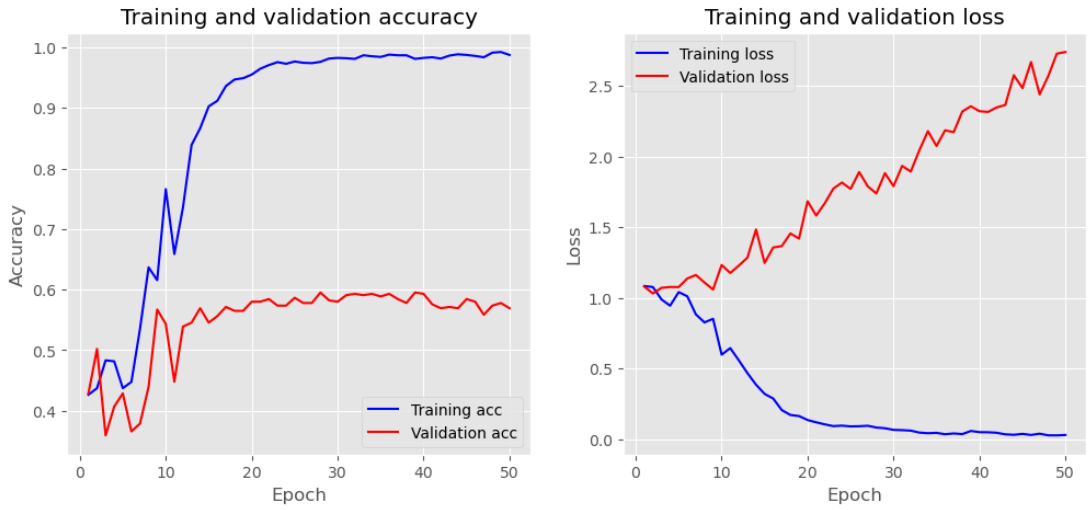
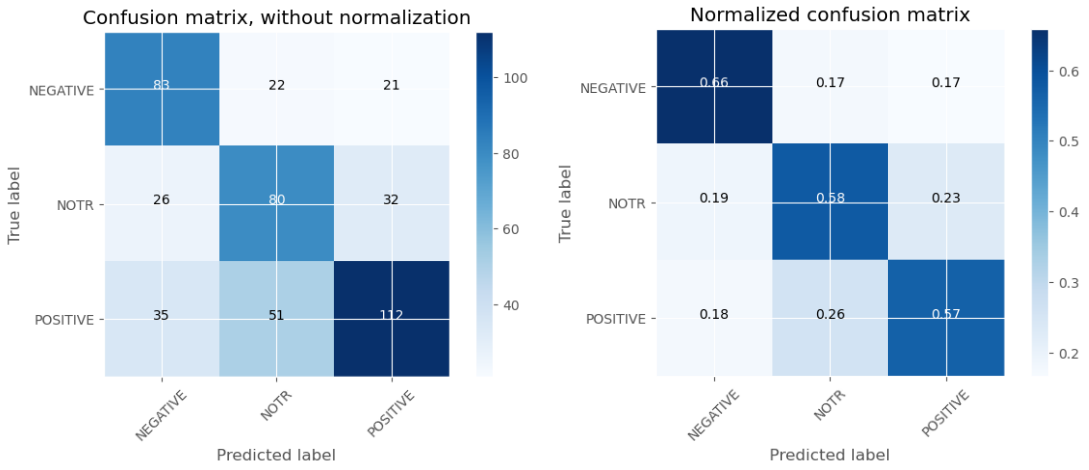


Figure 4.35 The loss and accuracy graphs for multi-classes long short-term memory model with word embedding



(a) Confusion matrix **(b)** Normalized confusion matrix

Figure 4.36 The confusion matrices of test results for multi-classes long short-term memory model with word embedding

When we used fastText's pre-trained word embedding, our results are as shown in Table 4.24, Figure 4.37 and Figure 4.38.

Table 4.24 Training and testing accuracies for multi-classes long short-term memory model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Pre-trained word embedding	Train	99.24
Multi-classes	Pre-trained word embedding	Test	61.69

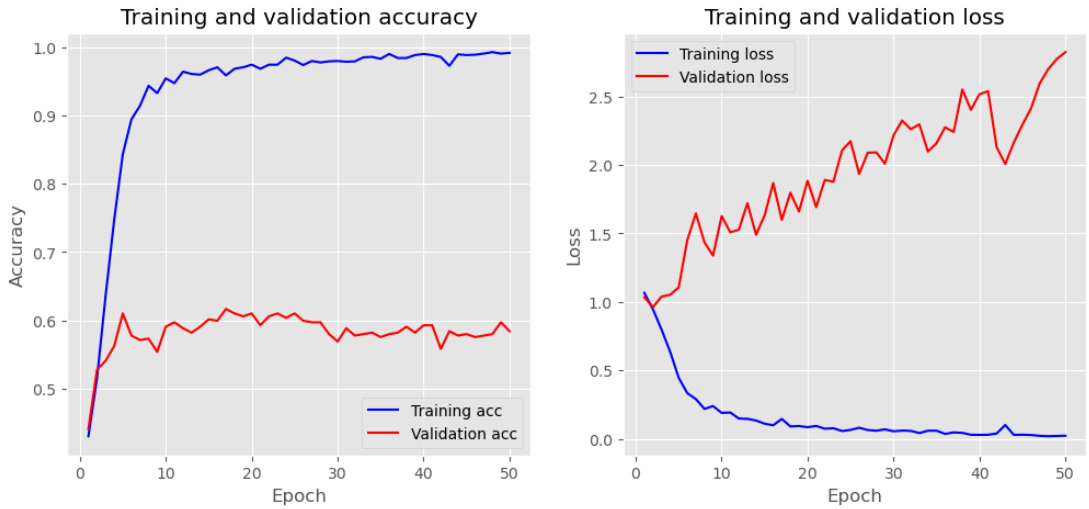
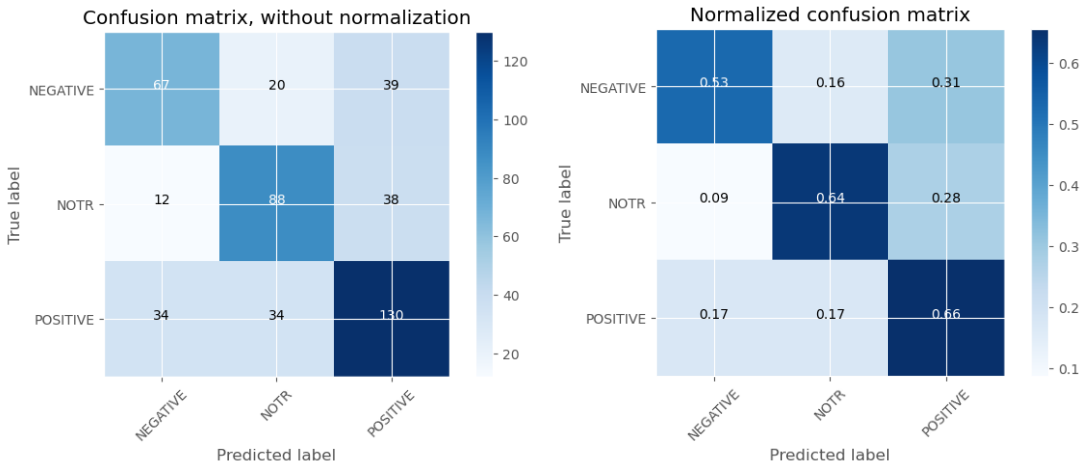


Figure 4.37 The loss and accuracy graphs for multi-classes long short-term memory model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.38 The confusion matrices of test results for multi-classes long short-term memory model with pre-trained word embedding

4.3.2.4 Gated Recurrent Units (GRU) Model

A gated recurrent units model using word embedding was designed with the below Python code snippet which is included in Listing C.4 in Appendix C. For multi-class GRU model has only one difference at its the output layer which includes three neurons.

Figure 4.39 shows our gated recurrent units model for binary classification.

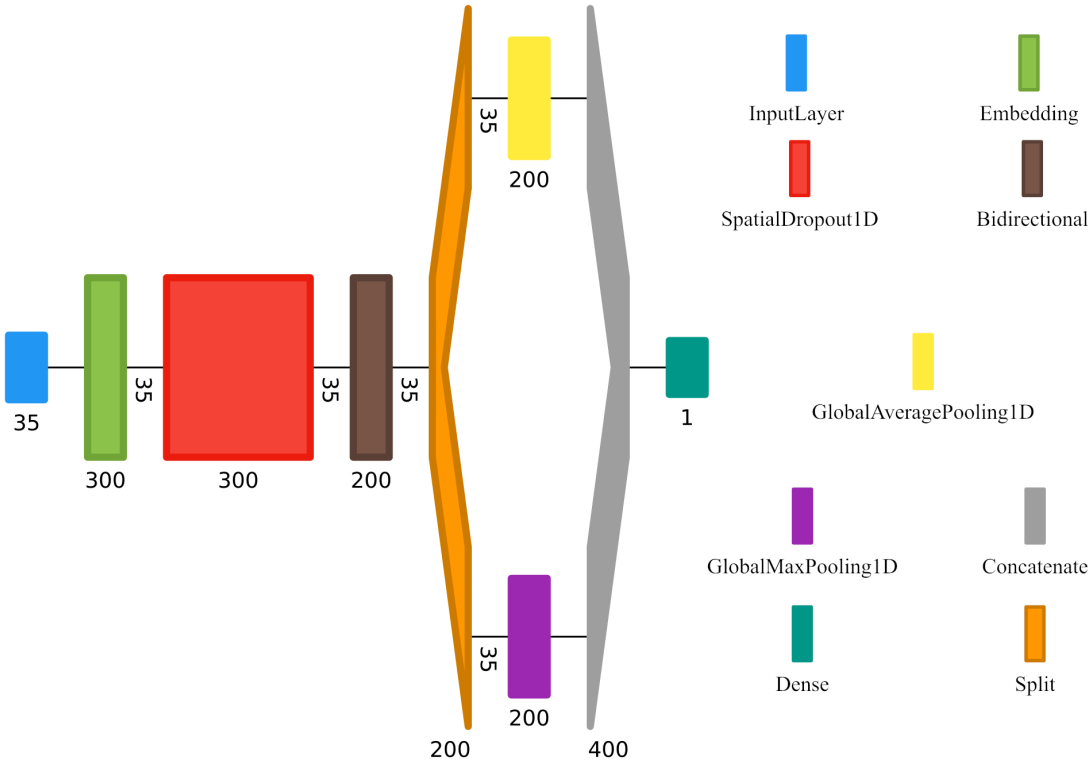


Figure 4.39 Binary gated recurrent units model

5-fold cross validation is used for training and testing process. Our training and testing accuracy results with word embedding for binary classification on binary dataset are as shown in Table 4.25, Figure 4.40 and Figure 4.41;

Table 4.25 Training and testing accuracies for binary gated recurrent units model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Word embedding	Train	100
Binary	Word embedding	Test	80.25

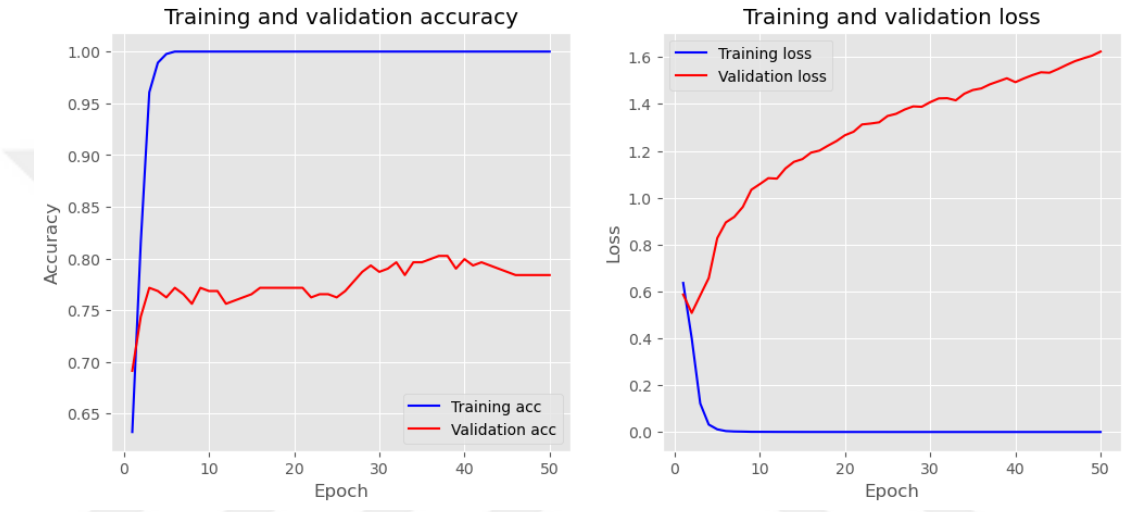
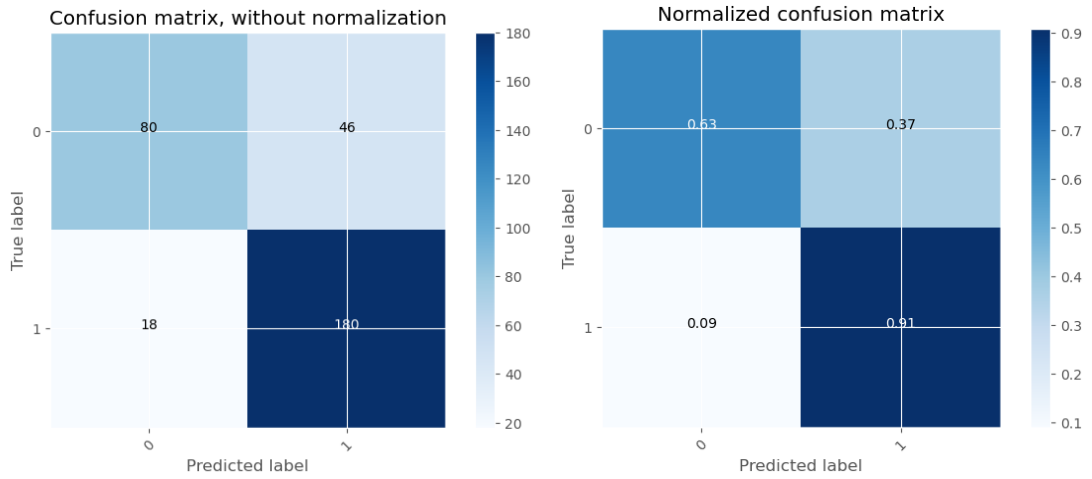


Figure 4.40 The loss and accuracy graphs for binary gated recurrent units model with word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.41 The confusion matrices of test results for binary gated recurrent units model with word embedding

When we used fastText's pre-trained word embedding, our results are as shown in Table 4.26, Figure 4.42 and Figure 4.43.

Table 4.26 Training and testing accuracies for binary gated recurrent units model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Pre-trained word embedding	Train	100
Binary	Pre-trained word embedding	Test	80.31

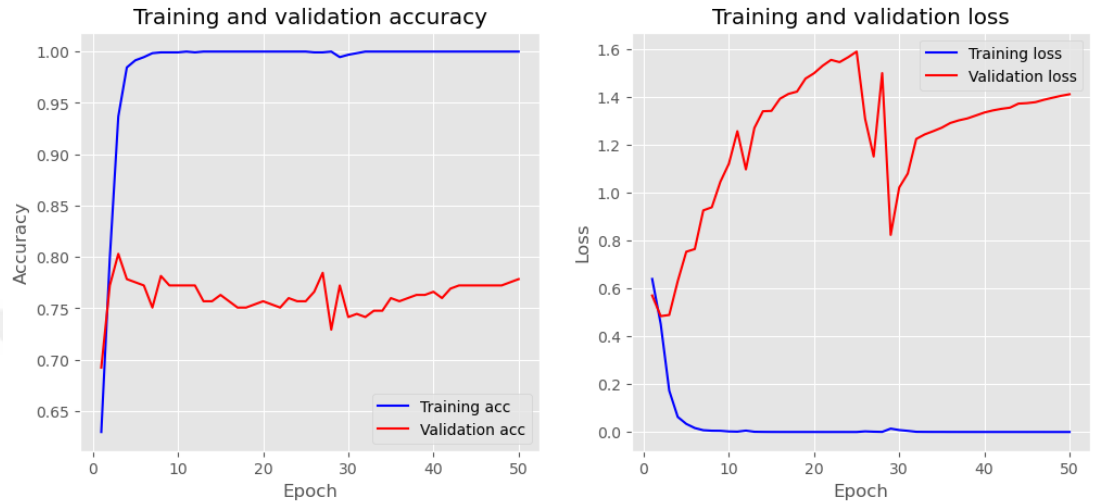
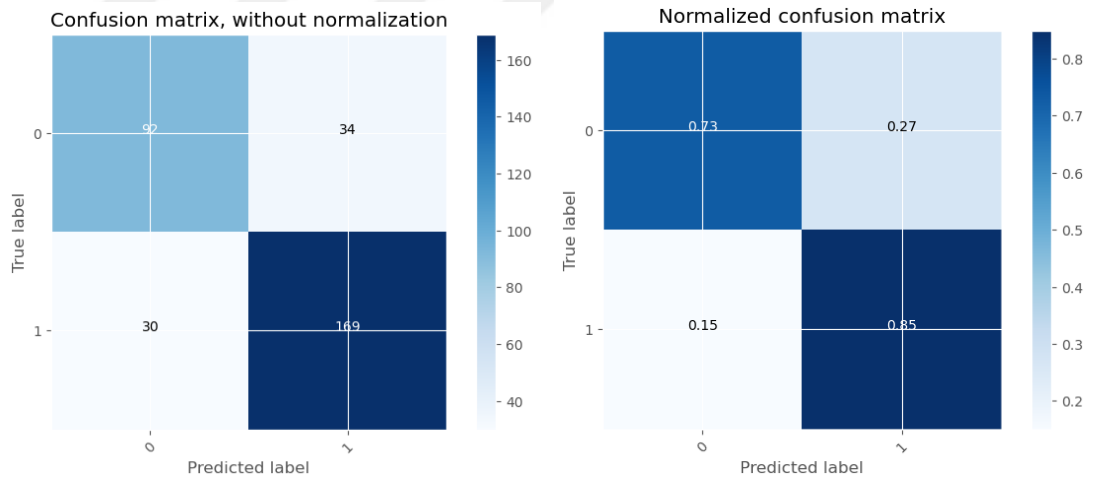


Figure 4.42 The loss and accuracy graphs for binary gated recurrent units model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.43 The confusion matrices of test results for binary gated rucurrent units model with pre-trained word embedding

When we used multi-classes gated recurrent units model, our training and testing accuracy results with word embedding for multi-classes classification on multi-classes data set are shown in Table 4.27, Figure 4.44 and Figure 4.45;

Table 4.27 Training and testing accuracies for multi-classes gated recurrent units model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Word embedding	Train	99.62
Multi-classes	Word embedding	Test	62.99

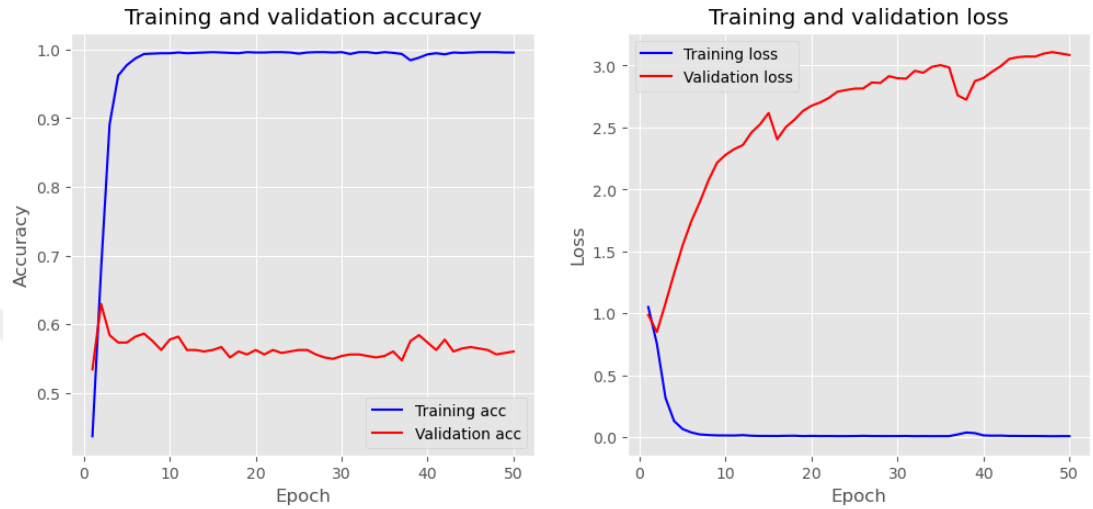
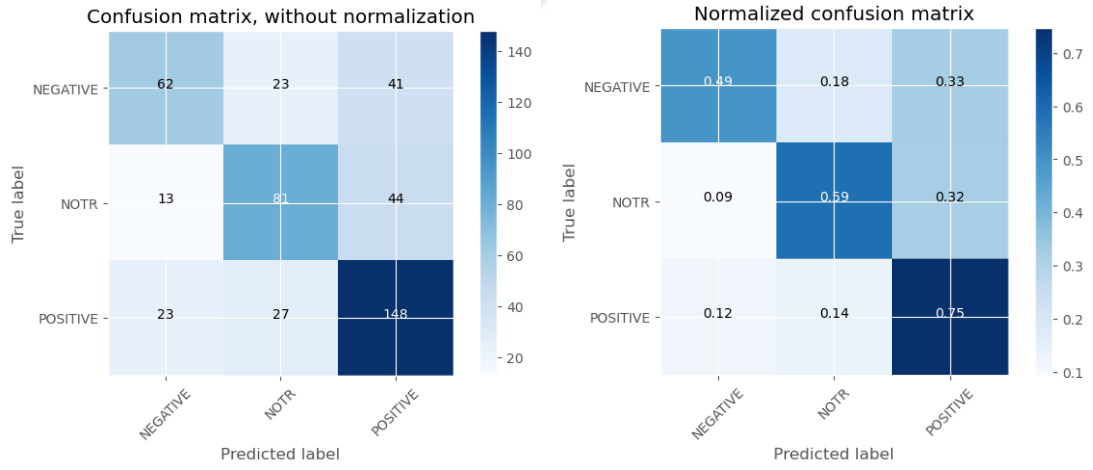


Figure 4.44 The loss and accuracy graphs for multi-classes gated recurrent units model with word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.45 The confusion matrices of test results for multi-classes gated recurrent units model with word embedding

When we used fastText's pre-trained word embedding, our results are as shown in Table 4.28, Figure 4.46 and Figure 4.47.

Table 4.28 Training and testing accuracies for multi-classes gated recurrent units model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Pre-trained word embedding	Train	99.68
Multi-classes	Pre-trained word embedding	Test	64.07

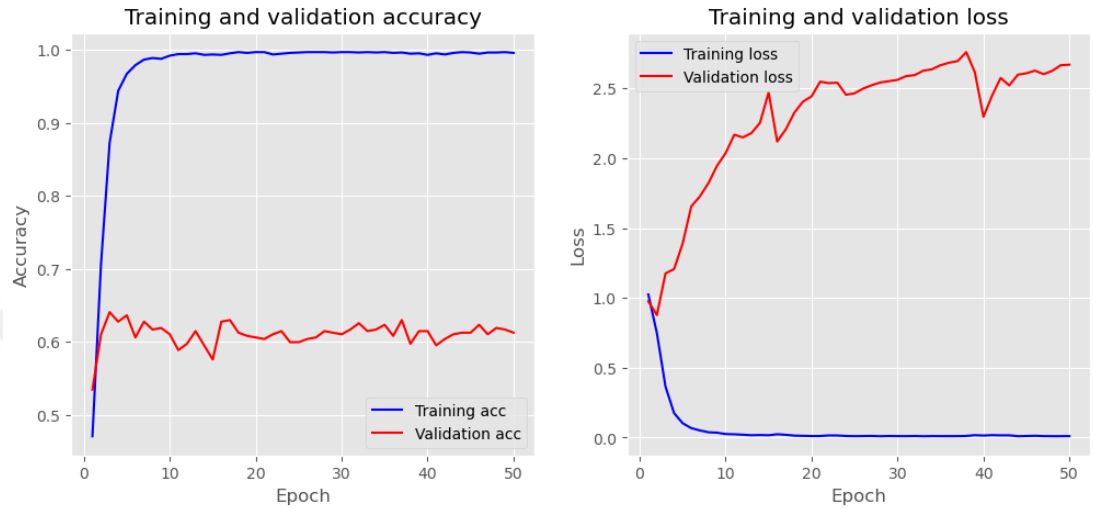
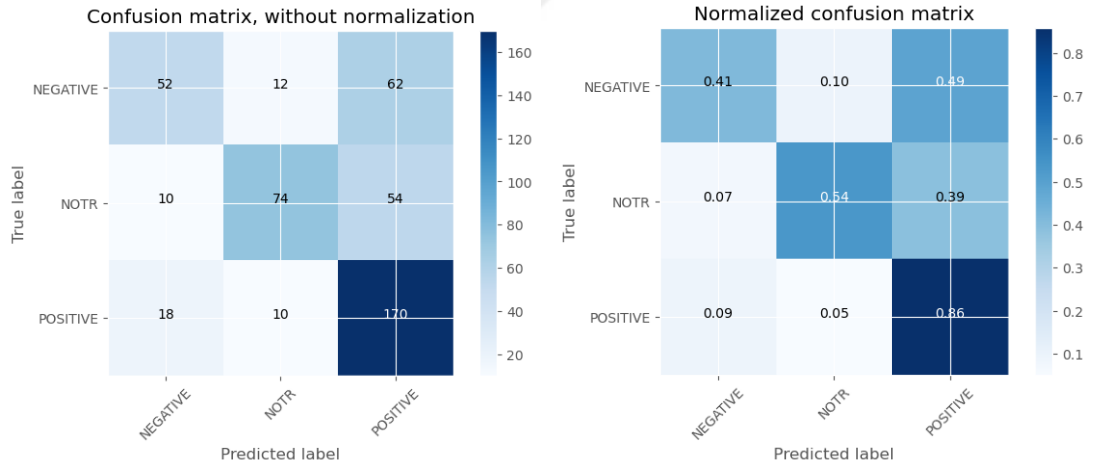


Figure 4.46 The loss and accuracy graphs for multi-classes gated recurrent units model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.47 The confusion matrices of test results for multi-classes gated recurrent units model with pre-trained word embedding

4.3.2.5 GRU – CNN Model

A GRU-CNN model using word embedding was designed with the below Python code snippet which is included in Listing C.5 in Appendix C. For multi-class GRU-CNN model has only one difference at its the output layer which includes three neurons.

Figure 4.48 shows our gru-cnn model for binary classification.

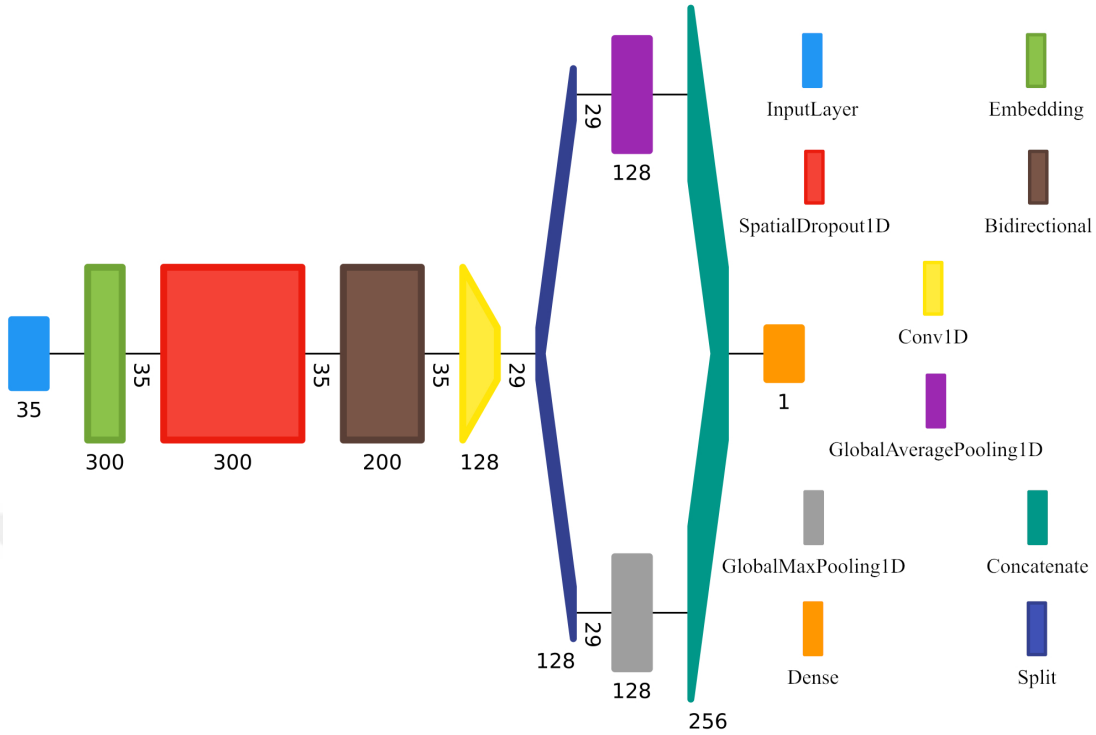


Figure 4.48 Binary gru-cnn model

5-fold cross validation is used for training and testing process. Our training and testing accuracy results with word embedding for binary classification on binary dataset are as shown in Table 4.29, Figure 4.49 and Figure 4.50;

Table 4.29 Training and testing accuracies for binary gru-cnn model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Word embedding	Train	100
Binary	Word embedding	Test	80.56

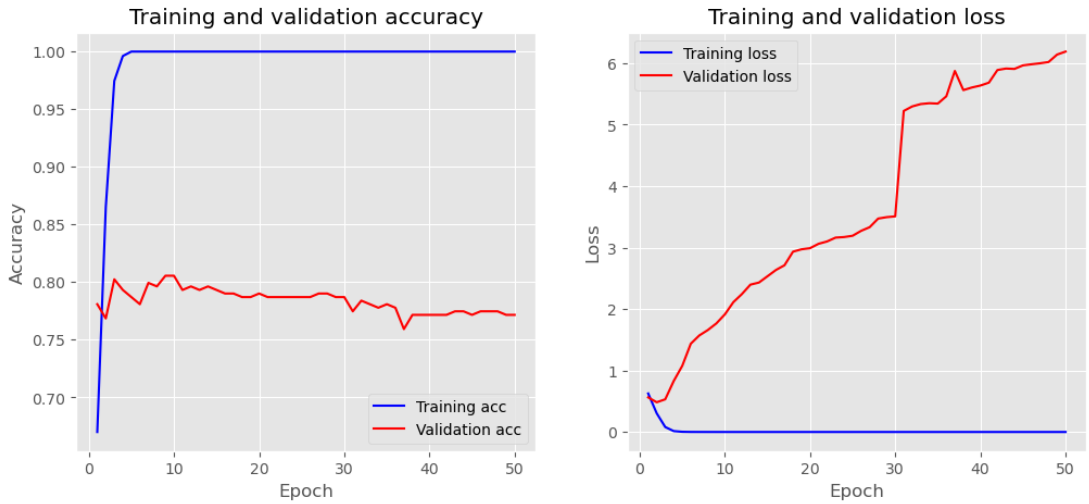
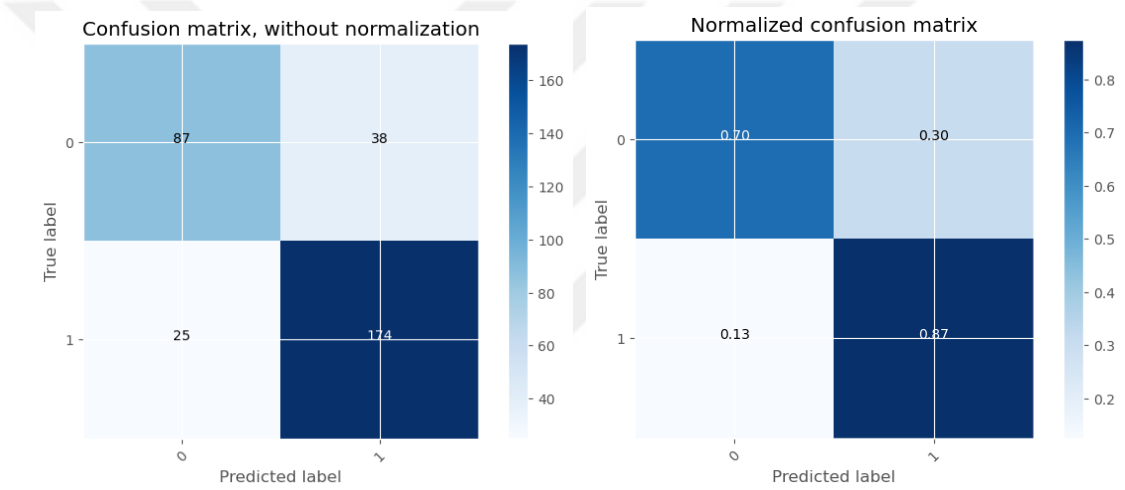


Figure 4.49 The loss and accuracy graphs for binary gru-cnn model with word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.50 The confusion matrices of test results for binary gru-cnn model with word embedding

When we used fastText's pre-trained word embedding, our results are as shown in Table 4.30, Figure 4.51 and Figure 4.52.

Table 4.30 Training and testing accuracies for binary gru-cnn model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Binary	Pre-trained word embedding	Train	100
Binary	Pre-trained word embedding	Test	80.25

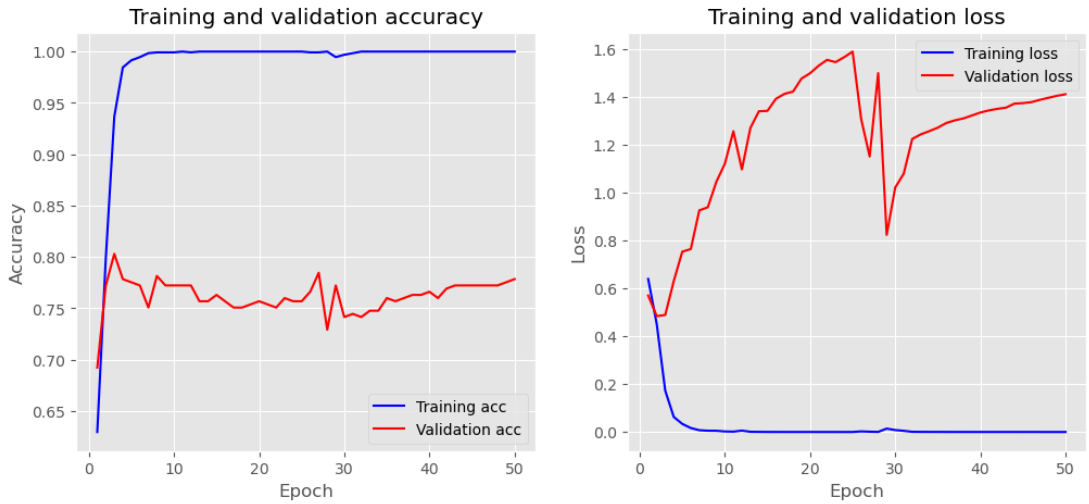
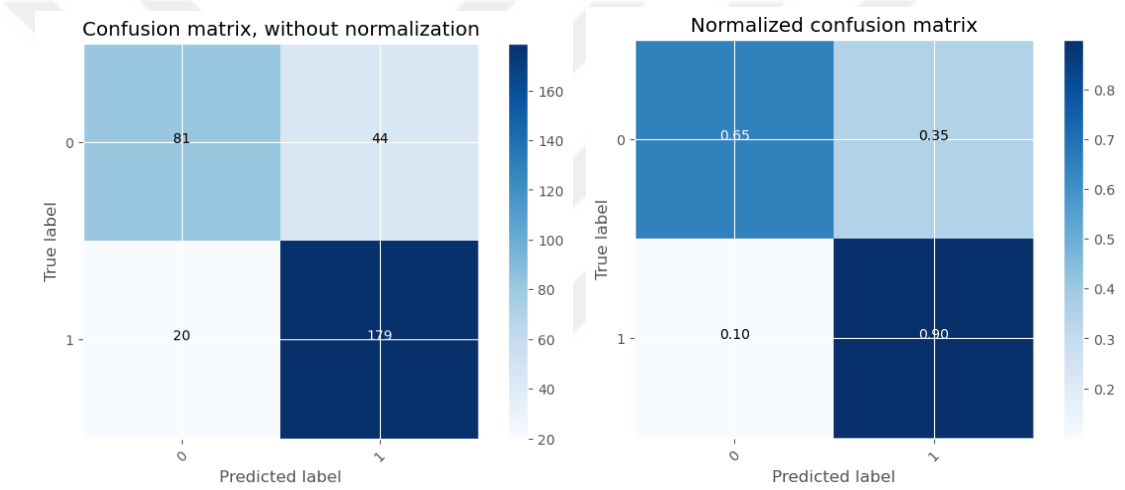


Figure 4.51 The loss and accuracy graphs for binary gru-cnn model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.52 The confusion matrices of test results for binary gru-cnn model with pre-trained word embedding

When we used multi-classes gated recurrent units model, our training and testing accuracy results with word embedding for multi-classes classification on multi-classes data set are shown in Table 4.31, Figure 4.53 and Figure 4.54;

Table 4.31 Training and testing accuracies for multi-classes gru-cnn model with word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Word embedding	Train	99.68
Multi-classes	Word embedding	Test	61.47

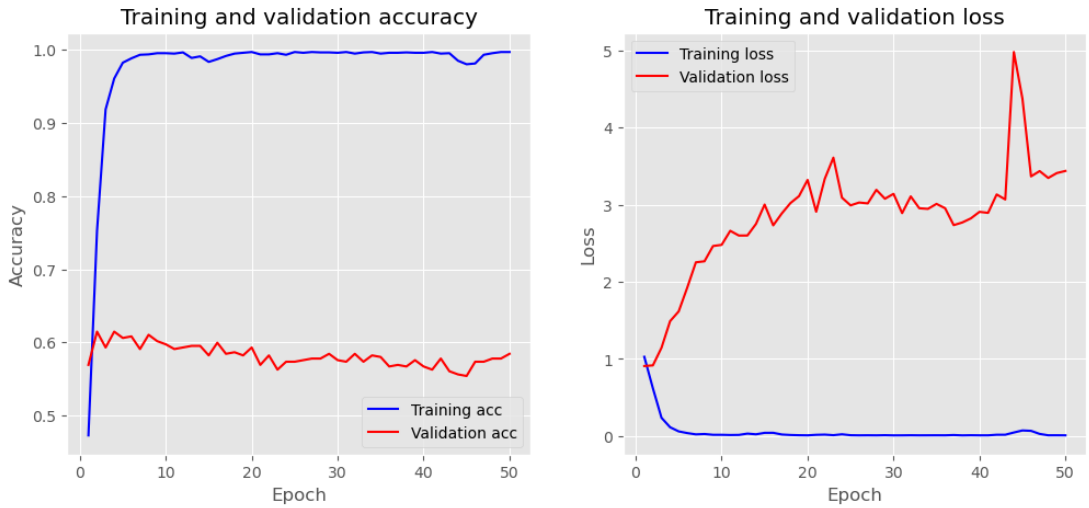
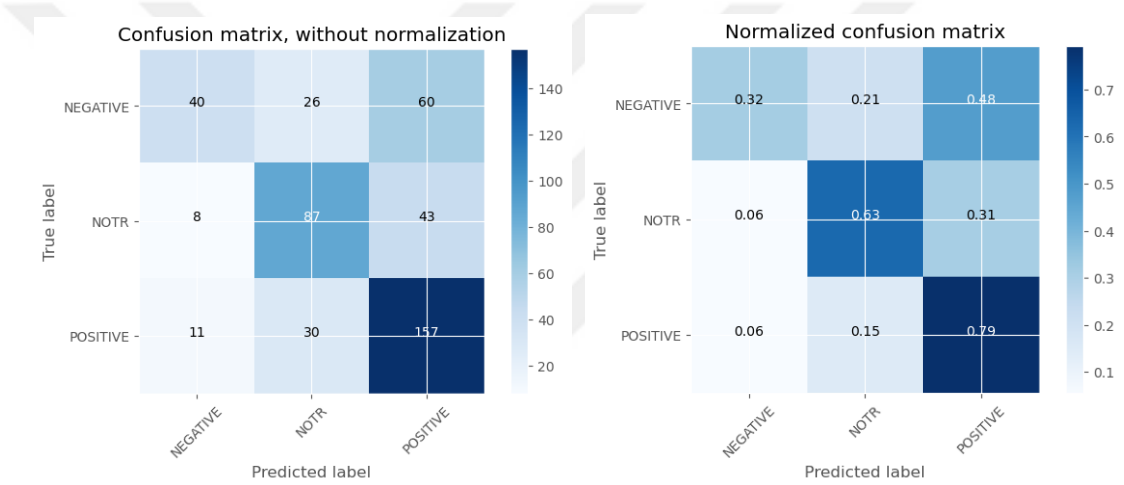


Figure 4.53 The loss and accuracy graphs for multi-classes gru-cnn model with word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.54 The confusion matrices of test results for multi-classes gru-cnn model with word embedding

When we used fastText's pre-trained word embedding, our results are as shown in Table 4.32, Figure 4.55 and Figure 4.56.

Table 4.32 Training and testing accuracies for multi-classes gru-cnn model with pre-trained word embedding

Classification	Embedding	Train/Test	Accuracy(%)
Multi-classes	Pre-trained word embedding	Train	99.68
Multi-classes	Pre-trained word embedding	Test	64.29

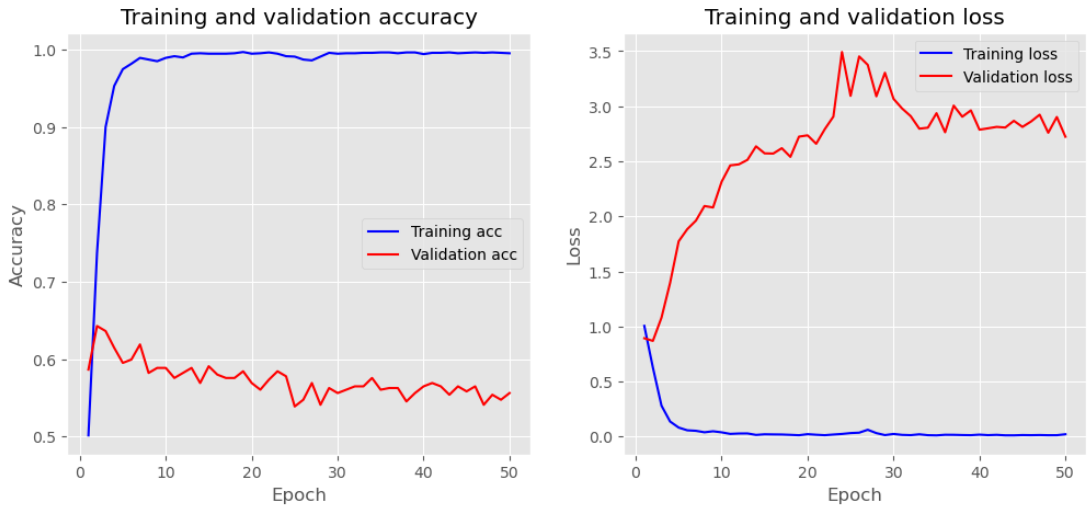
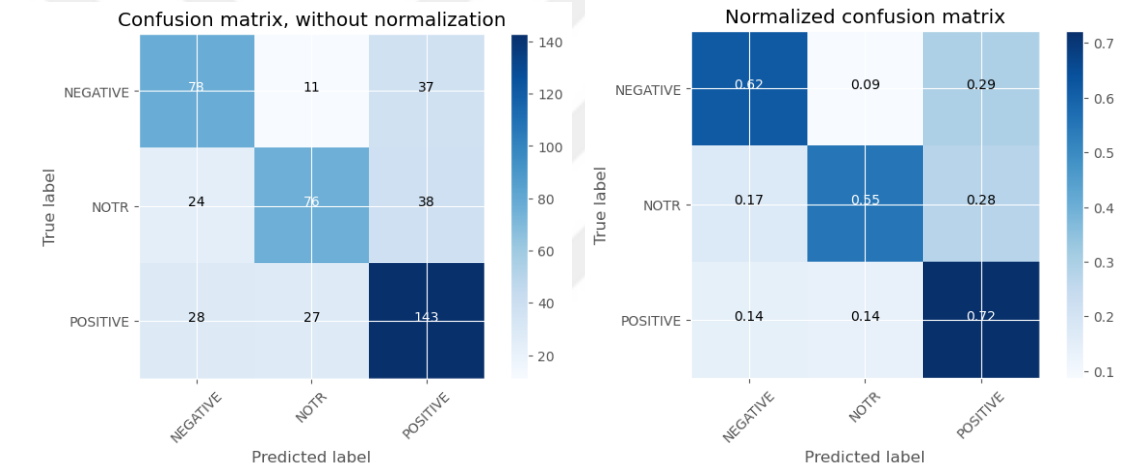


Figure 4.55 The loss and accuracy graphs for multi-classes gru-cnn model with pre-trained word embedding



(a) Confusion matrix

(b) Normalized confusion matrix

Figure 4.56 The confusion matrices of test results for multi-classes gru-cnn model with pre-trained word embedding

CHAPTER 5

EXPERIMENTAL RESULTS

We created our datasets using Turkish financial tweets, and we tried five different machine learning algorithms (Neural Network, CNN, LSTM, GRU, and GRU-CNN) to find sentiments on those datasets together with word embedding and pre-trained word embedding. Our classification results are as shown in Table 5.1.

Table 5.1 Comparisons of model accuracies

Classification	Model	Embedding	Training Accuracy(%)	Testing Accuracy(%)	Average of Testing Accuracies of all folds(%)
Binary	NN	word embedding	100.00	80.25	76.68
		pre-trained word embedding	100.00	79.32	76.19
	CNN	word embedding	100.00	77.23	75.82
		pre-trained word embedding	**100.00**	**83.02**	**78.35**
	LSTM	word embedding	98.30	74.69	72.36
		pre-trained word embedding	100.00	79.32	75.14
	GRU	word embedding	100.00	80.25	76.93
		pre-trained word embedding	100.00	80.31	77.60
	GRU-CNN	**word embedding**	**100.00**	**80.56**	**76.44**
		pre-trained word embedding	100.00	80.24	78.47
Multiclass	**NN**	**word embedding**	**99.57**	**63.85**	**61.59**
		pre-trained word embedding	99.51	65.23	61.59
	CNN	word embedding	99.73	63.71	61.98
		pre-trained word embedding	**99.73**	**72.73**	**65.05**
	LSTM	word embedding	99.24	59.52	58.34
		pre-trained word embedding	99.24	61.69	58.69
	GRU	word embedding	99.73	62.99	60.94
		pre-trained word embedding	99.68	64.07	62.67
	GRU-CNN	word embedding	99.68	61.47	60.08
		pre-trained word embedding	99.68	64.29	62.33

Our results show that generally, all models show better performance when they are run with pre-trained fastText word vectors. Also, binary classification results are better than multi-classes classification results as expected. Surprisingly, the results are close to each other. With pre-trained word embedding, CNN models had the best results of all. When we used word embedding, the GRU-CNN model gave better results on binary classification and the Neural Network model gave better results on multi-classes classification.

We wrote an article with title "comparative study for sentiment analysis of financial tweets with deep learning methods" taking advantage of our study. It was published in Applied Sciences [45].

CHAPTER 6

CONCLUSIONS

In this study, tweets from BIST-100 stock exchange index were analyzed as financial indicator. We found relations between BIST 100 stock exchange index companies. We collected tweets with keywords which are extracted from these relations.The collected tweets were tagged as “Positive”, “Negative” and “Neutral”. Then binary and multi-class datasets were created. As a classifier, Neural Network, Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM), Gated Recurrent Units (GRU) and GRU-CNN models were used.

Our results show that generally, all models show better performance when they are run with pre-trained fastText word vectors. Also, binary classification results are better than multi-classes classification results as expected. Surprisingly, the results are close to each other. With pre-trained word embedding, CNN models had the best results of all. When we used word embedding, the GRU-CNN model gave better results on binary classification and the Neural Network model gave better results on multi-classes classification.

We propose a CNN model with pre-trained word embedding for binary and multi-class classification. Its testing accuracy is 83.02% and the average of its testing accuracies of all folds is 78.35% for binary classification. For multi-classes classification, its testing accuracy is 72.73% and the average of its testing accuracies of all folds is 65.05%.

In future works, using additional layers in these models may improve their performances. The use of more specific preprocessing techniques can also improve model performances, as the collected Turkish tweets about the Turkish financial market contain many ambiguous words and phrases that make the pre-processing step difficult. In addition, enlarging the datasets can lead to better results. And the results of these models can be used as the input of BIST100 forecasting algorithms.

REFERENCES

[1] Cambria E., Schuller B. B., Xia Y., and Havasi C., "New Avenues in Opinion Mining and Sentiment Analysis," *IEEE Intelligent Systems*, vol. 28, no. 2, pp. 15–21, 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6468032

[2] Liu B., "Sentiment analysis and opinion mining," *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–184, 2012.

[3] Medhat W., Hassan A., and Korashy H., "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2090447914000550

[4] Memis E. and Kaya H., "Association Rule Mining on the BIST100 Stock Exchange," in *3rd International Symposium on Multidisciplinary Studies and Innovative Technologies, ISMSIT 2019 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., oct 2019.

[5] Eryiğit G., "ITU Turkish NLP Web Service," in *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Gothenburg, Sweden: Association for Computational Linguistics, apr 2014. [Online]. Available: https://web.itu.edu.tr/gulsenc/papers/itunlp.pdf

[6] Mikolov T., Karafiat M., Burget L., Cernocky J., and Khudanpur S., "Recurrent neural network based language model," in *INTERSPEECH 2010*, 2010, pp. 1045–1048.

[7] Mikolov T., Sutskever I., Chen K., Corrado G. S., and Dean J., "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[8] Socher R., Perelygin A., Wu J. Y., Chuang J., Manning C. D., Ng A. Y., and Potts C., "Recursive deep models for semantic compositionality over a sentiment

treebank," in *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2013, pp. 1631–1642. [Online]. Available: http://nlp.stanford.edu/

[9] Young T., Hazarika D., Poria S., and Cambria E., "Recent trends in deep learning based natural language processing [review article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[10] Grave E., Bojanowski P., Gupta P., Joulin A., and Mikolov T., "Learning Word Vectors for 157 Languages," *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pp. 3483–3487, feb 2018. [Online]. Available: http://arxiv.org/abs/1802.06893

[11] Han J., Kamber M., and Pei J., *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[12] Aggarwal C. C., *Data Mining: The Textbook*, 2015. [Online]. Available: www.springer.com

[13] Agrawal R., Imielinski T., and Swami A., "Database mining: a performance perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 6, pp. 914–925, 1993. [Online]. Available: http://ieeexplore.ieee.org/document/250074/

[14] Agrawal R. and Srikant R., *Fast algorithms for mining association rules*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, p. 580–592.

[15] Hu M. and Liu B., "Mining and summarizing customer reviews," *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining KDD 04*, vol. 04, p. 168, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1014052.1014073

[16] Özgür A., Özgür L., and Güngör T., "Text categorization with class-based and corpus-based keyword selection," in *Computer and Information Sciences - ISCIS 2005*, Yolum p., Güngör T., Gürgen F., and Özturan C., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 606–615.

[17] “Fundamentals of Bag Of Words and TF-IDF - Analytics Vidhya - Medium.” [Online]. Available: https://medium.com/analytics-vidhya/fundamentals-of-bag-of-words-and-tf-idf-9846d301ff22

[18] Nabi J., “Machine Learning — Text Processing - Towards Data Science,” 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958

[19] Harris Z. S., “Distributional Structure,” *Distributional Structure, WORD*, vol. 10, no. 3, pp. 146–162, 1954. [Online]. Available: https://www.tandfonline.com/action/journalInformation?journalCode=rwrd20

[20] “Practical Text Classification With Python and Keras – Real Python.” [Online]. Available: https://realpython.com/python-keras-text-classification/

[21] Mikolov T., Chen K., Corrado G. S., and Dean J., “Efficient estimation of word representations in vector space,” in *International Conference on Learning Representations*, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:5959482

[22] Pennington J., Socher R., and Manning C. D., “GloVe: Global vectors for word representation,” in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014, pp. 1532–1543. [Online]. Available: http://nlp.

[23] Bojanowski P., Grave E., Joulin A., and Mikolov T., “Enriching word vectors with subword information,” *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[24] “A Short Tutorial on Word Embeddings | Kaggle.” [Online]. Available: https://www.kaggle.com/c/spooky-author-identification/discussion/44190

[25] “A simple Word2vec tutorial - Zafar Ali - Medium.” [Online]. Available: https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1

[26] "14.5. Subword Embedding (fastText) — Dive into Deep Learning 0.7.1 documentation." [Online]. Available: http://www.d2l.ai/chapter_natural-language-processing/fasttext.html

[27] Cambria E. and White B., "Jumping NLP Curves: A Review of Natural Language Processing Research," *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 48–57, 2014.

[28] Collobert R., Weston J., Bottou L., Karlen M., Kavukcuoglu K., and Kuksa P., "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, no. null, p. 2493–2537, nov 2011.

[29] Collobert R. and Weston J., "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 160–167. [Online]. Available: http://wordnet.princeton.edu

[30] Elvis, "Deep Learning for NLP: An Overview of Recent Trends," 2018. [Online]. Available: https://medium.com/dair-ai/deep-learning-for-nlp-an-overview-of-recent-trends-d0d8f40a776d

[31] Elman J. L., "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[32] Hochreiter S. and Schmidhuber J., "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] Gers F., Schmidhuber J., and Cummins F., "Learning to forget: continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 850–855 vol.2.

[34] Wu C. and Su Y.-C., "Dynamic relations among international stock markets," *International Review of Economics & Finance*, vol. 7, no. 1, pp. 63–84, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1059056099800173

[35] Almohaimeed A. S., "Using Tweets Sentiment Analysis to Predict Stock Market Movement," *MSc Thesis*, 2017. [Online]. Available: https://etd.auburn.edu/bitstream/handle/10415/5759/UsingTweetsSentimentAnalysistoPredictStockMarketMovement.pdf?sequence=2

[36] Ariya A., "Stock Forecasting by Association Rule Mining," in *Proceedings of the 21st Asia-Pacific Conference on Global Business, Economics, Finance & Social Sciences (AP18Taiwan Conference)*, 2018. [Online]. Available: www.globalbizresearch.org

[37] Kumar M. and Kalia A., "Association Rules Mining in the Stock Data," Tech. Rep. [Online]. Available: http://www.lingayasuniversity.edu.in/ijdatpm/Pdf_paper/Paper_7.pdf

[38] Karpio K., Lukasiewicz P., Orowski A., and Zbkowski T., "Mining associations on the warsaw stock exchange," in *Acta Physica Polonica A*, vol. 123, no. 3, 2013, pp. 553–559. [Online]. Available: http://przyrbwn.icm.edu.pl/APP/PDF/123/a123z3p12.pdf

[39] Simsek M. and Ozdemir S., "Analysis of the relation between turkish twitter messages and stock market index." *2012 6th International Conference on Application of Information and Communication Technologies (AICT), Application of Information and Communication Technologies (AICT), 2012 6th International Conference on*, pp. 1 – 4, 2012.

[40] Yildirim M. and Yüksel C. A., "Sosyal medya ile hisse senedi fiyatının günlük hareket yönü arasındaki İlişkinin İncelenmesi: Duygu analizi uygulaması," *Uluslararası İktisadi ve İdari İncelemeler Dergisi*, 2017.

[41] Ozturk S. S. and Ciftci K., "A Sentiment Analysis of Twitter Content as a Predictor of Exchange Rate Movements," *Review of Economic Analysis*, vol. 6, no. 2, pp. 132–140, 2014.

[42] Savaş İ. and Can İ., “Euro-Dolar Paritesi ve Reel Döviz Kuru’nun İMKB 100 Endeksi’ne Etkisi,” pp. 323–339, 2011. [Online]. Available: http://www.acarindex.com/dosyalar/makale/acarindex-1423880307.pdf

[43] KENDİRLİ S. and ÇANKAYA M., “DOLAR KURU’NUN BORSA İSTANBUL-30 ENDEKSİ ÜZERİNDEKİ ETKİSİ VE ARALARINDAKİ NEDENSELLİK İLİŞKİSİNİN İNCELENMESİ,” *Celal Bayar Üniversitesi Sosyal Bilimler Dergisi*, vol. 14, no. 2, 2016. [Online]. Available: http://dergipark.gov.tr/download/article-file/224582

[44] Bauerle A., van Onzenoodt C., and Ropinski T., “Net2vis – a visual grammar for automatically generating publication-tailored cnn architecture visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 6, p. 2980–2991, Jun. 2021. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2021.3057483

[45] Memiş E., Akarkamçı H., Yeniad M., Rahebi J., and Lopez-Guede J. M., “Comparative study for sentiment analysis of financial tweets with deep learning methods,” *Applied Sciences*, vol. 14, no. 2, p. 588, 2024.

APPENDICES

Appendix A: Python Code: Transforms Stock Items Data File to the Tag File.

Appendix B: Python Code: Tweet Collection Program

Appendix C: Deep Learning Models' Codes

Appendix A – Python Code: Transforms Stock Items Data File to the Tag File

```
# Process BIST100 data and create a file with nominal values.
# Threshold value is selected as 1% to define "arti",
# "eksi" and "ayni".
import csv
import glob
import os
import sys
rootpath = "C:\\RelatedStocksDataFilesFolder\\"
outfilename="CreatedCSVfileName.csv"

def decideTrend(prev, curr, thr):
delta = 0.0
#print("prev = %f, curr = %f, thr = %f " % (prev,curr,thr))
delta = ((curr - prev) * 100) / prev
if(delta >= 0.0):
if(delta > thr):
return "arti"
else:
return "ayni"
else:
if(delta < -1 * thr):
return "eksi"
else:
return "ayni"

filenames = sorted(glob.glob(rootpath + '*.is.csv'))
numoffiles = len(filenames)
numoffiles += 1 ## Added for date column.
threshold = 0.1 # Threshold is selected as 1.0%
print('number of files: %s' % numoffiles)
filenumber=0
mylist=[];
for f in filenames:
filename=os.path.splitext(os.path.basename(f))[0]
print("%s dosyasi isleniyor." % filename)
```

```
handle=open(f)
count=0
previous=""
filenumber +=1
if filenumber == 1:
r = csv.reader(handle)
for row in r:
data=numoffiles*[0]
if count == 0:
data[0] = 'Date'
data[filenumber]= filename
print("%s, %s, %s" %(row[0], row[4], filename))
elif count == 1:
if((row[4]=='null') or (row[4] == "")):
print(red("ilk deger null"))
sys.exit()
previous=row[4]
data[0]=row[0]
data[filenumber] = 'ayni'
else:
if((row[4]=='null') or (row[4] == "")):
print("%s tarihli deger null onceki deger: %s atandi."\
%(row[0], previous))
row[4]=previous
value = decideTrend(float(previous),float(row[4]),threshold)
previous=row[4]
data[0]=row[0]
data[filenumber] = value
count += 1
mylist.append(data)
else:
#add file data into related column
r = csv.reader(handle)
for row in r:
if count == 0:
mylist[count][filenumber]= filename
print("%s, %s, %s" %(row[0], row[4], filename))
elif count == 1:
```

```
74  if((row[4]=='null') or (row[4] == "")):
75  print("ilk deger null")
76  sys.exit()
77  previous=row[4]
78  if mylist[count][0] != row[0]:
79  print("tarihler ayni degil! %s != %s in file %s"\
80  %(mylist[count][0],row[0],filename))
81  sys.exit()
82  mylist[count][filenumber] = 'ayni'; #ilk deger 0
83  else:
84  #value=0
85  if((row[4]=='null') or (row[4] == "")):
86  print("%s tarihli deger null, onceki deger: %s atandi."\
87  %(row[0], previous))
88  row[4]=previous
89  value = decideTrend(float(previous),float(row[4]),threshold)
90  previous=row[4]
91  if mylist[count][0] != row[0]:
92  print('tarihler ayni degil! %s != %s in file %s'\
93  %(mylist[count][0],row[0],filename))
94  sys.exit()
95  mylist[count][filenumber] = value
96  count += 1
97  handle.close()
98  #filenumber += 1
99  print('<-------------------------------------------------------->')
100 print('write created dataset into createdds.csv file')
101
102 with open(rootpath + outfilename,"w",newline='') as csv_file:
103 writer = csv.writer(csv_file,delimiter=',')
104 for row in mylist:
105 print(row)
106 writer.writerow(row)
```

Listing A.1: Pyton code to transform stock items data

Appendix B – Python Code: Tweet Collection Program

```
"""
utf 8 encoding.
It collects and stores tweets into mysql and mongodb databases
"""
from __future__ import print_function
from mysql.connector import MySQLConnection, Error, IntegrityError
from python_mysql_dbconfig import read_db_config
import tweepy
import json
from pymongo import MongoClient
from dateutil import parser
import re
"""text bilgisi extended ya da normal text alanindan aliniyor."""
MONGO_HOST= 'mongodb://localhost/twitterdb'
# assume that you have mongoDB installed locally
# and a database called 'twitterdb'
WORDS = ['#AFYON', '#AKBNK', '#AKSA', '#AKSEN', '#ALARK', '#ANACM',
         '#ARCLK', '#BRSAN', '#DEVA', '#ECILC', '#ECZYT', '#GARAN',
         '#GOLTS', '#GOODY', '#GUBRF', '#HALKB', '#IPEKE', '#ISCTR',
         '#ISGYO', '#KARTN', '#KCHOL', '#KOZAA', '#MGROS', '#PETKM',
         '#PGSUS', '#SAHOL', '#SISE', '#TATGD', '#THYAO', '#TMSN',
         '#TRKCM', '#TTKOM', '#VAKBN', '#VESTL', '#YKBNK', '#ZOREN']


db_config = read_db_config() #read config file
auth_config = read_db_config('config.ini','authentication')

"""
The store_data function takes the 'created_at', 'text',
'screen_name', 'tweet_id' and 'topic' and stores it into
a MySQL database
"""
def store_data(created_at, text, screen_name, tweet_id, topic):
    query = "INSERT INTO " \
        "twitter (tweet_id, screen_name, created_at, tekst) " \
        "VALUES (%s, %s, %s, %s)"
    args = (tweet_id, screen_name, created_at, text)
```

```
38
39      query1 = "INSERT INTO tweetsentiments (topic, tweet_id) " \
40                      "VALUES (%s, %s)"
41      args1 =(topic, tweet_id)
42
43      try:
44          db=MySQLConnection(**db_config)
45          print("MYSQL connection is opened")
46          cursor = db.cursor()
47          cursor.execute(query1, args1)
48          db.commit()
49          print("record inserted successfully into " \
50                "tweetsentiments table")
51          cursor.execute(query, args)
52          db.commit()
53          print("record inserted successfully into twitter table")
54      except IntegrityError as err:
55          print("Error: {}".format(err))
56      except Error as error:
57          print(error)
58      finally:
59          if(db.is_connected()):
60              cursor.close()
61              db.close()
62              print("MYSQL connection is closed")
63
64  def word_in_text(word, text):
65      word = word.lower()
66      text = text.lower()
67      match = re.search(word, text)
68      if match:
69          return True
70      return False
71
72  class StreamListener(tweepy.StreamListener):
73  # This is a class provided by tweepy to access
74  #the Twitter Streaming API.
75      def on_connect(self):
```

```
76          # Called initially to connect to the Streaming API
77          print("You are now connected to the streaming API.")
78      def on_error(self, status_code):
79          # On error - if an error occurs,
80          # display the error / status code
81          print('An Error has occured: ' + repr(status_code))
82          return False
83      def on_data(self, data):
84          # This is the core of the script...
85          # it connects to databases and stores the tweet
86          try:
87              client = MongoClient(MONGO_HOST)
88              # Use twitterdb database.
89              # If it doesn't exist, it will be created.
90              db = client.twitterdb
91              # Decode the JSON from Twitter
92              datajson = json.loads(data)
93              # mongodb part
94              # grab the 'created_at' data from
95              # the Tweet to use for display
96              created_at = parser.parse(datajson['created_at'])
97              # print out a message to the screen that
98              # we have collected a tweet
99              print("Tweet collected at " + str(created_at))
100             # insert the data into the mongoDB collection
101             # called twitter_search if twitter_search
102             # doesn't exist, it will be created.
103             db.twitter_search.insert(datajson)
104             print("Tweet stored into mongodb")
105             # end of mongodb part
106             # grab the wanted data from the Tweet
107             truncated = datajson['truncated']
108             #print('truncated: %s' % truncated)
109             if truncated :
110                 text = datajson['extended_tweet']['full_text']
111                 print('extended text: %s' % text)
112             else:
113                 text = datajson['text']
```

```
114                 print('normal text: %s' % text)
115             screen_name = datajson['user']['screen_name']
116             tweet_id = datajson['id']
117             # insert the data into the MySQL database
118             for word in WORDS:
119                 if word_in_text(word,text):
120                     print("The key: %s found in text" % word)
121                     store_data(created_at, text,
122                                screen_name, tweet_id, word)
123                 #else:
124                     #print("The key: %s does not " \
125                     #"exist in text" % word)
126             print("Tweet stored into mysql")
127         except Exception as e:
128             print(e)
129 
130 auth = tweepy.OAuthHandler(
131     auth_config['consumer_key'], auth_config['consumer_secret'])
132 auth.set_access_token(
133     auth_config['access_token'], auth_config['access_token_secret'])
134 # Set up the listener.
135 # The 'wait_on_rate_limit=True' is needed
136 # to help with Twitter API rate limiting.
137 listener = StreamListener(api=tweepy.API(wait_on_rate_limit=True))
138 streamer = tweepy.Stream(auth=auth, listener=listener)
139 print("Tracking: " + str(WORDS))
140 streamer.filter(track=WORDS)
```

Listing B.1: Collects and stores tweets

Appendix C – Deep Learning Models' Codes

```
#... codes before
#maxlen parameter to specify how long the sequences should be.
maxlen = 35
embedding_dim = 300
vocab_size=5000
...
#Create a new model
model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
output_dim=embedding_dim,
input_length=maxlen))
model.add(layers.GlobalMaxPool1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

#compile new model
model.compile(optimizer='adam',
loss='binary_crossentropy',
metrics=['accuracy'])
#... codes after
```

Listing C.1: Python code for neural network model

```

# ... codes before
vocab_size=5000
embedding_dim = 300
maxlen = 35
def get_cnn_model():
filter_sizes = [2, 3, 5]
num_filters = 128
drop = 0.3
inputs = Input(shape=(maxlen,), dtype='int32')
embedding = Embedding(input_dim=vocab_size,
output_dim=embedding_dim,
input_length=maxlen)(inputs)
reshape = Reshape((maxlen, embedding_dim, 1))(embedding)
conv_0 = Conv2D(num_filters,
kernel_size=(filter_sizes[0], embedding_dim),
```

```
padding='valid', kernel_initializer='normal',
activation='relu')(reshape)
conv_1 = Conv2D(num_filters,
kernel_size=(filter_sizes[1], embedding_dim),
padding='valid', kernel_initializer='normal',
activation='relu')(reshape)
conv_2 = Conv2D(num_filters,
kernel_size=(filter_sizes[2], embedding_dim),
padding='valid', kernel_initializer='normal',
activation='relu')(reshape)
maxpool_0 = MaxPool2D(pool_size=(maxlen - filter_sizes[0] + 1, 1),
strides=(1,1), padding='valid')(conv_0)
maxpool_1 = MaxPool2D(pool_size=(maxlen - filter_sizes[1] + 1, 1),
strides=(1,1), padding='valid')(conv_1)
maxpool_2 = MaxPool2D(pool_size=(maxlen - filter_sizes[2] + 1, 1),
strides=(1,1), padding='valid')(conv_2)
concatenated_tensor = Concatenate(axis=1)(
[maxpool_0, maxpool_1, maxpool_2])
flatten = Flatten()(concatenated_tensor)
dropout = Dropout(drop)(flatten)
output = Dense(units=1, activation='sigmoid')(dropout)
model = Model(inputs=inputs, outputs=output)
adam = Adam(lr=1e-4, beta_1=0.9, beta_2=0.999,
  epsilon=1e-08, decay=0.0)
model.compile(optimizer=adam, loss='binary_crossentropy',
metrics=['accuracy'])
return model
# ... codes after
```

Listing C.2: Python code for convolutional neural network model

```
# ... codes before
vocab_size=5000
embedding_dim = 300
maxlen = 35
def get_binary_lstm_model():
inp = Input(shape=(maxlen, ), dtype="int32")
x = Embedding(input_dim=vocab_size,
output_dim=embedding_dim,
input_length=maxlen)(inp)
```

```
x = LSTM(embedding_dim, dropout=0.2, recurrent_dropout=0.2)(x)
outp = Dense(1, activation="sigmoid")(x)
model = Model(inputs=inp, outputs=outp)
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
return model
# ... codes after
```

Listing C.3: Python Code for long short-term memory model

```
# ... codes before
vocab_size=5000
embedding_dim = 300
maxlen = 35
def get_binary_gru_model():
inp = Input(shape=(maxlen, ), dtype="int32")
x = Embedding(input_dim=vocab_size,
output_dim=embedding_dim,
input_length=maxlen)(inp)
x = SpatialDropout1D(0.3)(x)
x = Bidirectional(GRU(100, return_sequences=True))(x)
avg_pool = GlobalAveragePooling1D()(x)
max_pool = GlobalMaxPooling1D()(x)
conc = concatenate([avg_pool, max_pool])
outp = Dense(1, activation="sigmoid")(conc)

model = Model(inputs=inp, outputs=outp)
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
return model
# ... codes after
```

Listing C.4: Python Code for gated recurrent units model

```
# ... codes before
vocab_size=5000
embedding_dim = 300
maxlen = 35
def get_gru_cnn_model():
inp = Input(shape=(maxlen, ))
x = Embedding(input_dim=vocab_size,
output_dim=embedding_dim,
```

```
input_length=maxlen)(inp)
x = SpatialDropout1D(0.3)(x)
x = Bidirectional(GRU(100, return_sequences=True))(x)
x = Conv1D(128, kernel_size = 7, padding = "valid",
kernel_initializer = "he_uniform")(x)
avg_pool = GlobalAveragePooling1D()(x)
max_pool = GlobalMaxPooling1D()(x)
conc = concatenate([avg_pool, max_pool])
outp = Dense(1, activation="sigmoid")(conc)
model = Model(inputs=inp, outputs=outp)
model.compile(loss='binary_crossentropy',optimizer='adam',
metrics=['accuracy'])
return model
# ... codes after
```

Listing C.5: Python Code for gru-cnn model