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

**PhD THESIS**

**Yasin BEKTAŞ**

# A SENTENCE BOUNDARY DETECTION SYSTEM FOR TURKISH TEXTUAL DOCUMENTS

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**

**ADANA, 2022**

**ABSTRACT**

**PhD THESIS**

**A SENTENCE BOUNDARY DETECTION SYSTEM FOR TURKISH TEXTUAL DOCUMENTS**

**Yasin BEKTAŞ**

**ÇUKUROVA UNIVERSITY**
**INSTITUTE OF NATURAL AND APPLIED SCIENCES**
**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**

Supervisor : Prof. Dr. Selma Ayşe ÖZEL
Co-supervisor : Prof. Dr. Sera Yeşim AKSAN
Year: 2022, Pages: 83
Jury : Prof. Dr. Selma Ayşe ÖZEL
: Prof. Dr. Mustafa GÖK
: Prof. Dr. Mutlu AVCI
: Assoc. Prof. Dr. Erdinç AVAROĞLU
: Assoc. Prof. Dr. İrem ERSÖZ KAYA

This study aims to design a high-performance method for sentence boundary detection in accordance with its own language structure in order to increase the prevalence of Turkish in the digital world. When general-purpose sentence boundary detection applications were tested on corpora containing different usage patterns of Turkish, it was observed that these applications remained at low success rates. In the study within the scope of the thesis, a hybrid method is proposed by combining the FP-Growth method, which is used to generate association rules, and Recurrent Neural Network (RNN) methods. A set of features has been obtained, in which suffix and POS tag information are used together, as well as some formal rules for the tokens before and after the possible end-of-sentence character. Then, this data set was tested with RNN methods such as LSTM and BiLSTM in the Turkish National Corpus (TNC) sub-corpus. In general-purpose applications, it was observed that the f-score value, which was 77% in the tests performed with the same data set, reached 96% with the proposed method. Within the scope of the study, it was observed that the success reached similar values in different Turkish corpora. In addition, in the study, it was concluded that the POS tag information and suffix structures of the tokens for Turkish play an important role in the detection of the boundary of a sentence.

**Keywords:** Sentence Boundary Detection, Recurrent Neural Network, FP-Growth, Turkish Natural Language Processing, Corpus Annotation, Grammatical Cues

# ÖZ

## DOKTORA TEZİ

## TÜRKÇE METİN BELGELERİ İÇİN CÜMLE SINIRI TESPİT SİSTEMİ

**Yasin BEKTAŞ**

**ÇUKUROVA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

Danışman : Prof. Dr. Selma Ayşe ÖZEL
İkinci Danışman : Prof. Dr. Sera Yeşim AKSAN
Yıl: 2022, Pages: 83
Jüri : Prof. Dr. Selma Ayşe ÖZEL
: Prof. Dr. Mustafa GÖK
: Prof. Dr. Mutlu AVCI
: Doç. Dr. Erdinç AVAROĞLU
: Doç. Dr. İrem ERSÖZ KAYA

Bu çalışma, Türkçe'nin dijital dünyada yaygınlığının arttırılması amacıyla kendi dil yapısına uygun başarımı yüksek bir cümle sonu belirleme yönteminin tasarlanmasını amaçlamaktadır. Genel amaçlı cümle sonu belirleme uygulamaları Türkçe'nin farklı kullanım şekillerini barındıran derlemler üzerinde test edildiğinde, bu uygulamaların düşük başarı değerlerinde kaldığı görülmüştür. Tez kapsamındaki çalışmada, birliktelik kuralları üretmeye yarayan FP-Growth yöntemi ile Tekrarlamalı Sinir Ağları (TSA) yöntemleri birleştirilerek hibrit bir metod önerilmiştir. Muhtemel cümle sonu karakterinin öncesi ve sonrasındaki dizgeciklere ait bir takım biçimsel kuralların yanında ek ve POS etiket bilgilerinin birlikte kullanıldığı bir nitelik kümesi elde edilmiştir. Ardından bu veri seti LSTM ve BiLSTM gibi TSA yöntemleri ile Türkçe Ulusal Derlemi (TUD) alt derleminde test edilmiştir. Genel amaçlı uygulamalarda aynı veri seti ile yapılan testlerde %77 olan f-ölçeği başarı değerinin, önerilen yöntem ile %96 değerine ulaştığı gözlenmiştir. Çalışma kapsamında farklı Türkçe derlemlerde de başarının benzer değerlere ulaştığı görülmüştür. Ayrıca çalışma içerisinde Türkçe için dizgeciklere ait POS etiket bilgileri ve ek yapılarının cümle sonu tespitinde önemli rol oynadığı sonucuna ulaşılmıştır.

**Anahtar Kelimeler:** Cümle Sonu Tespiti, Tekrarlamalı Sinir Ağları, FP-Growth, Türkçe Doğal Dil İşleme, Derlem Açıklaması, Dilbilgisel İpuçları

## EXTENDED ABSTRACT

Natural Language Processing (NLP), which is basically a sub-domain of Artificial Intelligence and Linguistics, enables the interaction between the languages that people use in daily life and computers. This interaction between computers and humans started in the 1950s (Grissman 1986). Later on, the use of NLP studies has increased rapidly in many research areas such as document translation, question-answering, or text summarization. The biggest problem encountered in all these processes is the complexity of natural languages. With the rapid developments in technology, this complexity has been successfully solved by computers (Eric, 1997). The primary problem to be solved is to determine the sentence boundaries correctly.

Sentence boundary detection (SBD) is the most critical preprocessing step in NLP studies. The process of finding the correct meaning of a word with multiple meanings can only be determined with a good sentence analysis. At the same time, a good translation system or question-answering system first considers the basic unit as a sentence (Kiss and Struck, 2006). In all NLP studies, the SBD procedure affects the success in the first degree. The characters used at the end of sentences in written texts are punctuation marks such as period (.), ellipsis (…), question mark (?), exclamation point (!), and colon (:) (TDK, 2020). However, these signs do not only indicate the end of the sentence. In addition, they can be found in the text for other purposes. On the basis of the SBD process, the purpose of the use of these characters should be clarified.

It is possible to examine SBD processes under two groups: The first of these is the rule-based approach. In this approach, hand-created rule lists are used especially for detection of abbreviations. In addition, it is difficult to understand the rules used, and these rules are limited to the texts used in the studies (Dinçer & Karaoğlan, 2004). The other group of approaches includes machine learning methods. In this approach, a comprehensive corpus is needed for the system to learn. In machine learning methods, we usually come across corpora prepared in British

English and American English, which are widely used in the world. Moreover, there is also a corpus created for Turkish to be used in Turkish Linguistics and NLP studies.

In general-purpose studies trained in various English corpora, f-score values have been reported between 97% and 98.80%. In studies conducted in Turkish corpora, it has been observed that this rate is between 91% and 99.80%. When the reasons for the high f-score rates were examined, it was determined that the contents of the corpus that is used were mostly composed of newspapers, news, and articles which have similar sentence structures(Aksan et al., 2014). Some commonly used SBD systems have been tested in the study conducted with the sub-corpus of the Turkish National Corpus (TUD), in which Turkish has different usage patterns. In these tests, the highest accuracy rate was seen as 88%. The fact that the test results are lower than those reported in a large corpus of different uses of Turkish reveals the necessity of a SBD system for Turkish.

In this thesis, a new SBD method is proposed in order to use the Turkish language more effectively in digital environments. The proposed method includes a machine learning model that uses the specific linguistic features of Turkish. At the same time, the main purpose is to train the learning model to be created with a corpus that can represent Turkish in the best way.

Our proposed method consists of a two-stage hybrid system. In the first step, matrices consisting of sequential vectors describing the tokens before and after the end-of-sentence characters are obtained. These vectors contain prefix and suffix information as well as format properties of tokens, and Part of Speech (POS) tag information. In the study, for the first time, it was presented that there is an association between the end-of-sentence characters and the suffix information of the tokens. This association was determined by using the FP-Growth method. As a result, it was decided to place suffix information on the vectors describing the tokens. Thus, sequential data is created using vectors describing tokens.

In the second stage of the system, this sequential data was used in Recurrent Neural Networks (RNN) for classification. Two different RNN models were designed and tested both in the TNC sub-corpus and in different Turkish corpora. As a result of the tests, the best result in the TNC sub-corpus was obtained with the f-score value of 95.92%, with Bidirectional Long Short Term Memory (BiLSTM).

In comparison with other end-of-sentence detection systems, the proposed system obtained 19% improvement with the same data set. In the framework of the study, it has been shown that in addition to the formal features of the texts in Turkish SBD operations, the POS tag information and the suffixes used should be taken into consideration in machine learning based SBD systems.

## GENİŞLETİLMİŞ ÖZET

Doğal Dil İşleme (DDİ) temelde insanların günlük hayatta kullandığı diller ile bilgisayarlar arasında iletişimi sağlayan Yapay Zeka ve Dilbilim'in bir alt dalıdır. Bilgisayar ile insan arasındaki bu etkileşim 1950' li yıllarda başlamıştır (Grissman 1986). Daha sonraları DDİ çalışmalarının kullanımı döküman çevirisi, soru-cevaplama veya metin özetleme gibi bir çok alanda hızla artmıştır. Tüm bu süreçlerde karşılaşılan en büyük problem doğal dillerin karmaşıklığıdır. Teknolojide yaşanan hızlı gelişmeler ile bu karmaşıklığın bilgisayarlar tarafından başarıyla çözümlenmesi sağlanmıştır (Eric, 1997). Bu karmaşıklığın en başında cümle sınırlarının doğru bir şekilde tespiti gelmektedir.

Cümle sonu tespit (CST) işlemi DDİ uygulamalarındaki en kritik ön işlem adımıdır. Bir çok anlama sahip bir kelimenin doğru anlamının bulunma süreci ancak iyi bir cümle analiziyle ortaya konulabilir. Aynı zamanda iyi bir çeviri sistemi veya soru-cevaplama sistemleri öncelikle temel birimi cümle olarak ele almaktadır (Kiss and Struck, 2006). Tüm DDİ uygulamalarında da CST işlemi başarıyı birinci derecede etkilemektedir. Yazılı metinlerde cümle sonunda kullanılan karakterler nokta (.), üç nokta (…), soru işareti (?), ünlem işareti (!) ve iki nokta üst üste (:) gibi noktalama işaretleridir (TDK, 2020). Fakat bu işaretler sadece cümle sonunu belirtmemektedir. Bunun yanında başka amaçlar ile de metin içinde bulunabilmektedirler. CST işlemi esasında bu karakterlerin kullanım amaçlarının belirginleştirilmesi gerekmektedir.

CST işlemlerini iki grup altında incelemek mümkündür. Bunlardan ilki kural tabanlı yaklaşımlardır. Bu yaklaşımda özellikle kısaltmalar için elle oluşturulmuş kural listeleri kullanılmaktadır. Ayrıca çıkarılan kurallar anlaşılması zor ve kural çıkarımında kullanılan metinler ile sınırlıdır (Dinçer ve Karaoğlan, 2004). Bir diğer yaklaşım ise makine öğrenmesi yöntemlerini kapsamaktadır. Bu yaklaşımda ise sistemin öğrenme yapabilmesi için kapsamlı bir derleme ihtiyaç vardır. Makine öğrenmesi yöntemlerinde genellikle karşımıza dünyada yaygın olarak kullanılan

İngiliz İngilizcesi ve Amerikan İngilizcesi ile hazırlanmış derlemler çıkmaktadır. Bunların yanında Türkçe Dilbilim ve DDİ çalışmalarında kullanılmak üzere Türkçe için oluşturulmuş derlemler de mevcuttur.

Genel amaçlı ve çeşitli İngilizce derlemlerde eğitilmiş çalışmalarda f-ölçeği değerleri %97 ile %98.80 arasında bildirilmiştir. Türkçe derlemlerde yapılan çalışmalarda ise bu oranın %91 ile %99.80 arasında kaldığı gözlemlenmiştir. F-ölçeği oranların yüksek çıkma sebepleri incelendiğinde kullanılan derlemlerin içeriklerinin büyük çoğunlukta benzer cümle yapılarına sahip gazete, haber ve makalelerden oluşmasından kaynaklandığı tespit edilmiştir (Aksan et al., 2014). Türkçenin farklı kullanım şekillerinin bulunduğu Türkçe Ulusal Derlemi (TUD)'ne ait alt derlem ile yapılan çalışmada, yaygın kullanılan bazı CST sistemleri test edilmiştir. Bu testlerde en yüksek doğruluk oranı %88 olarak görülmüştür. Türkçe'nin farklı kullanımlarının da olduğu geniş bir derlemde test sonuçlarının bildirilenlerden daha düşük çıkması Türkçe için bir CST sistemi gerekliliğini ortaya koymaktadır.

Bu tezde Türkçe dilinin dijital ortamlarda daha etkin kullanılması amacıyla yeni bir CST yöntemi önerilmektedir. Önerilen yöntem Türkçenin kendine has biçimsel özelliklerini kullanan bir makine öğrenmesi modelini kapsamaktadır. Aynı zamanda oluşturulacak olan öğrenme modelinin Türkçeyi en iyi temsil edebilecek bir derlem ile eğitilmesi amaçlanmıştır.

Önerdiğimiz yöntem iki aşamalı hibrit bir sistemden oluşmaktadır. İlk aşamasında, cümle sonu karakterlerinin öncesindeki ve sonrasındaki dizgecikleri tanımlayan sıralı vektörlerden oluşan matrisler elde edilmektedir. Bu vektörler, dizgeciklerin biçim özellikleri, sözcük tür (Part of Speech-POS) etiket bilgilerinin yanı sıra ek bilgilerini de barındırmaktadır. Çalışmada ilk kez cümle sonu karakterleri ile dizgeciklerin ek bilgileri arasında bir birliktelik olduğu ortaya konmuştur. Bu birliktelik FP-Growth yöntemi ile tespit edilmiştir. Bunun neticesinde dizgecikleri tanımlayan vektörlere ek bilgilerinin de yerleştirilmesine

karar verilmiştir. Böylece dizgecikleri tanımlayan vektörler kullanılarak bir sıralı veri oluşturulmuştur.

Sistemin ikinci aşamasında ise, oluşturulan bu sıralı veri Tekrarlayan Sinir Ağları (TSA)'nda sınıfladırma amacıyla kullanılmıştır. Tasarlanan iki farklı TSA modeli hem TUD alt derleminde hem de farklı Türkçe derlemlerde test edilmiştir. Testlerin neticesinde TUD alt derleminde alınan en iyi sonuç %95.92 f-ölçeği değeri ile Çift Yönlü Uzun Kısa Terim Hafızası (BiLSTM) yöntemi ile elde edilmiştir.

Diğer cümle sonu belirleme sistemleri ile yapılan karşılaştırmada, bu tezde önerilem yöntem ile aynı veri seti üzerinde CST işleminde %19 iyileşme sağlandığı görülmüştür. Çalışma çerçevesinde Türkçe CST işlemlerinde dizgeciklerin biçimsel nitelikleri yanında POS etiket bilgilerinin ve kullanılan eklerin makine öğrenmesi tabanlı CST sistemlerinde göz önünde bulundurulması gerektiği gösterilmiştir.

**ACKNOWLEDGEMENTS**

First of all, I would like to express my sincere thanks to my supervisor, Prof Dr. Selma Ayşe ÖZEL for her guidance, continuous support, patience, precious time, and help during my Ph.D.journey. I am always will be proud of being her student.

I would also like to thank my thesis committee members, Prof. Dr. Mustafa GÖK, Prof. Dr. Mutlu AVCI, Assoc. Prof. Dr. İrem ERSÖZ KAYA, and Assoc. Prof. Dr. Erdinç AVAROĞLU for their valuable pieces of advice during this study.

Additionally, I would like to thank the whole Turkish National Corpus (TUD) team, especially Prof. Dr.Sera Yeşim AKSAN and Assist. Prof. Dr.Umut U. DEMİRHAN, who made the data available for my experiments.

Finally, I especially want to thank my parents, my wife, and my daughter for their continuous support, understanding, and encouragement.

**CONTENTS** **PAGE**

**LIST OF TABLES** **PAGE**

**LIST OF FIGURES** **PAGE**

**ABBREVIATONS**

| | |
|---|---|
| 9ENLP | : 9 Eylül University Natural Language Processing |
| AI | : Artificial Intelligence |
| ANC | : American National Corpus |
| ANN | : Artificial Neural Networks |
| BERT | : Bidirectional Encoder Representations from Transformers |
| BiLSTM | : Bidirectional Long Short-Term Memory |
| BNC | : British National Corpus |
| BPNN | : Back Propagation Neural Network |
| CBOW | : Continuous Bag of Words |
| CNN | : Convolutional Neural Network |
| DNN | : Deep Neural Network |
| FN | : False Negative |
| FP | : False Positive |
| FP-GROWTH | : Frequent Pattern Growth |
| HMM | : Hidden Markow Model |
| ICE | : The International Corpus of English |
| JSBD | : Julie Sentence Boundary Detector |
| LSNLIS | : Lunar Sciences Natural Language Information System |
| LSTM | : Long Short- Term Memory |
| ME | : Maximum Entropi |
| METU | : Middle East Technical University |
| NB | : Naive Bayes |
| NLP | : Natural Language Processing |
| NSP | : Next Sentence Prediction |
| POS | : Part of Speech |
| RBFN | : Radial Basis Function Network |
| RNN | : Recurrent Neural Network |
| SBD | : Sentence Boundary Detection |

| | |
|---|---|
| SVM | : Support Vector Machine |
| TN | : True Negative |
| TNC | : Turkish National Corpus |
| TP | : True Positive |
| WSJ | : Wall Street Journal |
| WWW | : World Wide Web |
| XML | : eXtensible Markup Language |

# 1. INTRODUCTION

Natural Language Processing (NLP) is the provision of interaction with the computer using the structure of natural languages. It is known that this interaction was first started using translation methods in the 1950s (Grissman, 1986). NLP studies gained momentum and began to be used in many fields such as document translation, automatic speech, question-answering systems, emotion analysis, and text summarization at the beginning of 1970s. One of the biggest challenges in NLP is how to provide a computer with the linguistic complexity necessary for it to successfully perform language-based tasks (Eric, 1997).

In the process of understanding the language, the necessity of semantic and general world knowledge, other than lexical and linguistic abilities, was quickly noticed and new approach requirements emerged. The best solutions to these requirements have been produced with artificial intelligence (AI). Since the beginning of artificial intelligence (AI), which produces solutions to complex problems in many different fields, one of its most important goals has been to develop various computational methods for NLP. The first examples of using AI in NLP are question and answer systems. Examples of this are the LSNLIS (Lunar Sciences Natural Language Information System) (Woods, 1972), which provides data access to lunar geologists, and the Waltz system, which can perform question-answering from the database with aircraft maintenance (Waltz, 1978). The main disadvantage of these systems was that they were insufficient in matters other than a specific field.

Over time, two types of approaches have emerged in NLP studies. The first of these is the traditional natural language processing approach consisting of hand-coded rules. During the 1980s, there was a continuous improvement in these systems, which were coded manually. The reason for the difficulty of developing these systems was that they required domain knowledge. Accordingly, they were also insufficient outside the relevant domain where specific coding was performed.

Another approach is called empirical or corpus-based. In this approach, which is based on a large amount of data called corpus, automated systems have been emerged with machine learning methods.

### 1.1. What is a Corpus?

A Corpus as a language resource is a set of texts/speeches structured on the basis of specific purposes. A general-purpose corpus represents a particular language, at a specific time interval, and it contains a set of documents that consist of written and/or transcribed spoken use of language, texts/calls, writes/speaks properties. This set of documents consist of media and broadcast media type (books, periodicals, etc.). Furthermore, these types of documents have a balanced and layered structure obtained through sampling that cover basic knowledge and tools for interpreting the corresponding language. Furthermore, specified criteria and detailed linguistic analysis of data (metadata) are provided along with offering resources called corpora in electronic environments (Sinclair, 2005 - McEnery, 2012).

Due to advances in computer technologies, large compilations have been created that involve the use of real languages. “Text corpuses” or “Text Corpora" in their plural form, are used for statistical analysis and verification of language rules in a particular language or language workspace. Thanks to the studies in the field of linguistics and informatics conducted through these collections, many important features of the language that cannot be seen by other methods and tools have been revealed. Nowadays, a special or general-purpose corpus of a large number of languages has been established and made available to users (Lee, 2012).

The most well-known corpora created for the English language is the British National Corpus (BNC) (Burnard, 1998). BNC, created in English, is a corpus of 100 million words covering a wide range of written and transcribed spoken data resources. The American National Corpus (ANC) is a corpus created in American English containing written and transcribed spoken data of 22 million words produced

since 1990 (Fillmore, 1998). The Brown University Standard Corpus, which is called "Brown Corpus" for short, has been prepared with modern American English, and it contains 1 million words consisting of 500 text samples (Francis, 1979). Apart from these examples, there is "The International Corpus of English" (ICE) corpus, which represents the diversity of English in various regions of the world. There are examples of more than 20 countries where English is the first or second language (Greenbaum, 1996).

One of the earliest corpus for Turkish language is the METU Turkish Corpus, which is the closest one to the corpus definition made by Say (2002). This first written corpus of the Turkish language, consisting of 2 million words and taking samples of written texts from 291 different data sources after 1990, covers ten different text types. Researchers and application developers can run this corpus offline through platform-dependent software. The METU Turkish Corpus interface allows you to perform simple and regular expression queries, while it does not have compilation tools such as concordance lines, sorting, distribution, and numeric/ordinal collocation lists that are language assembly features. In the last 15 years, the inclusion of data on many languages on the Internet has made www fast, easy, and used to automatically compile resources of several natural languages without requiring manual optimizations (Hundt, 2007).

TurkishWaC (Ambati, 2012), developed for Turkish, consists of 42 million words acquired from Wikipedia pages by searching of the seed words. This corpus is accessible to users with the paid Sketch Engine corpus query system, suitable for performing word profile studies in the fields of linguistics and social sciences (http://sketchengine.co.uk ). Furthermore, the written corpora prepared by computer engineers for Turkish have received their data from the www again, and the corpora are large in terms of the number of words but do not comply with the principles of corpus design. Most of these corpora are a collection of texts compiled to obtain quantitative information about the linguistic units of the Turkish language, such as prefixes, suffixes, words, sentences, etc.

Turco (Dalkılıç, 2004) is a corpus containing more than 50 million words collected from 12 different internet addresses combined as a source, and some statistical characteristics of Turkish words and word groups were determined using this corpus. The BOUN Corpus (Sak, 2008) includes four different sub-corpora containing the internet pages of three different newspapers mainly read in Turkey and in which gathers around 423 million words from the webpages. A statistical model of the morphemes in Turkish words has been developed through this corpus. The BOUN Corpus is a language resource available to researchers in XML format. In addition, there is a 3.37 billion word-sized corpus (Baisa, 2012) to test formal deciphering software among Turkic languages, but this corpus is not available to researchers.

Contrary to the aforementioned corpora for Turkish language, the Turkish National Corpus (TNC) is developed according to the principles of linguistics, has a unique web-based interface, has the power to represent the language, is balanced, and has a high level of representation of Turkish (Aksan, 2012). In addition, it is the first reference corpus of the Turkish language containing written texts and transcribed spoken data samples.

### 1.2. Sentence Boundary Detection Process

The sentence boundary detection (SBD) process is the most important and critical preprocessing step in many natural language processing (NLP) studies. In most NLP studies, sentences form the basic unit above a word unit (Kiss and Struck, 2006). Therefore, SBD affects the success of NLP studies in the first degree.

The SBD process is difficult and complicated due to the ambiguity of the characters used to determine the boundary of the sentence. The characters used at the end of the sentence consist of punctuation marks such as period (.), ellipsis (…), question mark (?), exclamation point (!), and colon (:). These punctuation marks are used at the end of sentences, as well as in different positions to make the text easier to read and understand and to indicate features such as emphasis and tone. For example, the period (.) character is frequently encountered in title abbreviations,

ordinal numbers, and first name abbreviations, as well as in marking the end of the sentence. Examples of different ways of using the boundary of sentence characters for Turkish are seen in Table 1 (TDK, 2020).

Table 1.1. Different uses and examples of punctuation marks indicating the boundary of sentences (TDK, 2020)

| Character | Purpose of usage | Example |
|---|---|---|
| **Period** | In abbreviations | Alb. (albay), Dr. (doktor), Yrd.Doç. (yardımcı doçent), Prof. (profesör), Cad. (cadde), Sok. (sokak), s. (sayfa), sf. (sıfat), vb. (ve başkası, ve benzeri, ve benzerleri, ve bunun gibi), Alm. (Almanca), Ar. (Arapça), İng. (İngilizce) |
| **Period** | Declaring number of rows | 3. (üçüncü), 15. (on beşinci); II. Mehmet, XIV. Louis, XV. yüzyıl; 2. Cadde, 20. Sokak, 4. Levent vb. |
| **Period** | To indicate bullet points in a text | I. 1. A. a.<br>II. 2. B. b. |
| **Period** | To separate the numbers showing the month, day, and year in historiography | 29.5.1453, 29.X.1923 vb. |
| **Period** | To separate the numbers indicating hours and minutes | Tren 09.15'te kalktı.<br>Toplantı 13.00'te başladı.<br>Tören 17.30'da, hükûmet daireleri kapandıktan yarım saat sonra başlayacaktır. |
| **Period** | At the end of the tags of books, magazines, etc. | Agâh Sırrı Levend, *Türk Dilinde Gelişme ve Sadeleşme Evreleri,* TDK Yayınları, Ankara, 1960. |
| **Period** | As thousandth separator | 1.000, 326.197, 49.750.812 vb. |
| **Period** | On internet addresses | http://tdk.gov.tr<br>http://cu.edu.tr |
| **Period** | As a multiplication sign in math | 4.5=20, 12.6=72 vb. |
| **Colon** | To show long vowel in phonetics | a:ile, ka:til, usu:le, i:cat. |
| **Colon** | On internet addresses | http://tdk.gov.tr<br>http://cu.edu.tr |
| **Colon** | As a division sign in math | 56:8=7, 100:2=50 vb. |

| | | |
|---|---|---|
| **Ellipsis** | Instead of words and parts that are not meant to be written clearly | Kılavuzu karga olanın burnu b…tan çıkmaz.<br>Arabacı B…'a yaklaştığını söylüyor, ikide bir fırsat bularak arabanın içine doğru başını çeviriyordu. |
| **Ellipsis** | To show omission within the quotations | … derken şehrin öte başından boğuk boğuk sesler gelmeye başladı… |
| **Question mark** | Unknown, uncertain or suspicious place, date, etc. | Yunus Emre (1240 ?-1320), (Doğum yeri: ?) vb.<br>1496 (?) yılında doğan Fuzuli…<br>Ankara'dan Antalya'ya arabayla üç saatte (?) gitmiş. |
| **Exclamation point** | To give the meaning of ridicule, allusion, or contempt | İsteseymiş bir günde bitirirmiş (!) ama ne yazık ki vakti yokmuş (!).<br>Adam, akıllı (!) olduğunu söylüyor. |

The aim of this thesis is to detect the boundary of a sentence for Turkish by using the stylistic and idiosyncratic usage features of the Turkish language. Hereby, further contribution will be made for enhancing the prevalence of Turkish in the digital environment and power in NLP applications. The most important premise of this study is that the boundary of sentence detection systems for general use is not satisfactory in Turkish texts written in different genres and fields. For this reason, the use of a corpus that includes all forms of language use makes the study meaningful. By this means, a hybrid system that combines association rule mining and deep learning was proposed. In the proposed system, association rule mining were used to determine the grammatical units that are effective in sentence boundary detection, then these units were used as features for deep learning models. The contribution of the study to the literature can be expressed as follows;

- This study aims to design an effective system for sentence boundary detection by considering formal language features of Turkish, as general systems have not good performance for Turkish,

- In this study, a hybrid model has been trained using a corpus that contains examples of Turkish in different literary fields,
- To our best knowledge, this study is the first one that uses suffixes and POS tags of tokens around the end of sentence characters to determine the boundary of sentence, and their effects on success of the SBD process are discovered,
- Using type sequences of a certain number of words, including the boundary of sentence character, as features is investigated for both classical classifiers and deep learning methods for SBD from Turkish documents,
- The effect of using suffix sequences on the determination of the boundary of sentence with deep learning methods for Turkish is evaluated,
- The evaluation of results obtained by classical classifiers, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in the data sets created with the factors mentioned in the above items are made and discussed.

# 2. PREVIOUS WORKS

Different approaches have been used in the existing NLP systems in the sentence boundary detection process. There are many studies available for English and a few other common languages. These studies generally focused on two different methods. The first of these is the rule based SBD approach. In this approach, there are hand-made heuristics methods and hand-created rule lists (Ex: Abbreviations) to be checked (Read et al., 2012). The most important disadvantages of the rule-based SBD approach are; it can be said that it is difficult to understand and these systems are limited only to the texts used for developing the system (Dinçer and Karaoğlan, 2004).

The other used methods for the SBD process include machine learning techniques that apply supervised and unsupervised methods. A system trained on a corpus with sentence boundaries is called as supervised method. In the unsupervised method, which is less used in the SBD process in the literature, the system is expected to learn from unlabeled raw data.

When the previous studies are examined, it is possible to group them into two main sections that are general-purpose studies based on English, and studies conducted in Turkish, as summarized in the below subsections.

## 2.1. General-Purpose SBD Studies

Riley prepared one of the first applications of supervised machine learning for solving SBD problem by using decision tree classifiers (Riley, 1989). In the study based on American English, it was examined whether the character period marks the end of the sentence. Riley's approach used features such as word probabilities at the beginning or end of sentences and word lengths. Riley tested this work on Brown Corpus and achieved 99.8% success.

Palmer and Hearst proposed the SATZ system, which considers the context of the punctuation mark and uses a neural network or a decision tree to detect a true

sentence boundary (Palmer and Hearst, 1997). This system is also adaptable and has given good results for a variety of languages. The system does not require handmade grammar or syntax rules. They tested their system on the WSJ (Wall Street Journal) corpus. They reported an error rate of 1.1% in their experiment using a neural network and 1.0% error rate by using a decision tree classifier. Later, when they integrated the decision tree model with a heuristic-based approach, they managed to reduce the error rate to 0.5%.

One of the first general-purpose SBD studies is the study of Reynar and Ratnaparkhi (Reynar and Ratnaparkhi, 1997). They proposed a maximum entropy approach, which is a trainable probability model for detecting the boundary of a sentence. The system requires a detailed collection as it uses part of speech (POS) tags. This system needs data produced and labeled with the Latin alphabet. The study was tested on both the WSJ dataset and the Brown Corpus and achieved 98.8% and 97.9% accuracy, respectively.

Gllick, on the other hand, used the Support Vector Machine (SVM) and Naive Bayes (NB), which are among the supervised learning models, in his more recent study (Gillick, 2006). One of the best practices of the study, which focuses on period uncertainty, is the Splitta system. With this system, an error rate of 0.25% was reported in the WSJ dataset and 0.36% in the Brown Corpus.

Another study using the combination of the Hidden Markow Model (HMM) and Maximum Entropy (ME) method was revealed by Mikheev (Mikheev, 2000). In his work, which presents a prediction method for unknown abbreviations, he suggests that the order of POS tags may be effective for end-of-sentence highlighting. The error rate in his study was reported as 0.45% for the WSJ dataset and 0.28% for the Brown Corpus.

The study using a completely unsupervised learning system was presented by Punkt, Kiss, and Strunk (Kiss and Strunk 2006). The work is based on the definition of abbreviations by finding the purpose of placement of the point. The reported error rate in known test sets is 1.02% for Brown Corpus and 1.65% for

WSJ dataset. Furthermore, it has been observed that good results are also obtained in corpus collected from newspaper texts in different languages.

Another boundary of sentence detection method, which was created with the incremental machine learning approach, is the system called iSentenizer (Wong and Chao, 2010). The system uses an incremental decision tree learning algorithm. It is trained with a rule dataset consisting of the characteristics of the predecessor and successor tokens of the potential boundary of sentence character. In addition to the Brown Corpus in English, it was also tested with the Portuguese corpus of Tycho Brahe (Galves and Britto, 1999) and Hoje Macau. The f-measure values obtained were reported as 99.80%, 98.15%, and 97.61%, respectively. In another study conducted in 11 different languages using the same iSentenizer system, various f-measure results were measured between 95.10% and 98.07% (Wong et all, 2014).

In another recent study, three neural network architectures consisting of long short-term memory (LSTM), bidirectional long short-term memory (BiLSTM), and CNN were used for sentence end detection (Schweter and Ahmed, 2019). The neural network architectures specified with the dataset prepared from the states of the characters of the tokens surrounding the possible end-of-sentence marker were tested. As a result of the tests conducted with 11 different languages other than English (Europarl corpus was used for English) (Koehn, 2005), it has been reported that f-measure value of 97.59% was achieved in LSTM and BiLSTM networks, and 97.50% in CNN.

Another study with deep learning used the Indonesian language (Santosa et all, 2021). A bidirectional long short-term memory (BiLSTM) approach, which was designed according to Indonesian language features, was used. It has been reported that 98.49% of f-measure was achieved in the tests performed on the corpus of newspaper news.

Collective information regarding the general-purpose studies carried out is shown in Table 2.1.

Table 2.1. Summary of General-Purpose SBD Studies

| Study | Method | Dataset (Corpus) | Reported f-measure |
|---|---|---|---|
| **Riley(1989)** | Decision Trees | Brown Corpus | 99.80% |
| **Palmer and Hearst(1997)** | Neural Networks | WSJ | 98.90% |
| | Decision Trees | | 99.00% |
| **Reynar and Ratnaparkh(1996)** | Maximum Entropy | Brown Corpus | 98.80% |
| | | WSJ | 97.90% |
| **Gillick(2009)** | Support Vector Machines | Brown Corpus | 99.64% |
| | | WSJ | 99.75% |
| **Mikheev(2000)** | Hidden Markow Model | Brown Corpus | 99.72% |
| | | WSJ | 99.55% |
| **Kiss and Strunk(2006)** | Unsupervised Learning | Brown Corpus | 98,98% |
| | | WSJ | 98.35% |
| **Wong and Chao (2010)** | Incremental Decision Tree | Brown Corpus | 99.80% |
| | | Tycho Brahe | 98.15% |
| | | Hoje Macau | 97.61% |
| **Schweter and Ahmed(2019)** | LSTM | Europarl Corpus | 97.59% |
| | BiLSTM | | 97.59% |
| | CNN | | 97.50% |
| **Santosa Et. All (2021)** | BiLSTM | News Documents | 98.49% |

## 2.2. Studies with Turkish Corpora

It has been discovered that in most of the studies for determining the boundary of sentences for Turkish, statistical or machine learning-based methods are used. One of the earliest studies for Turkish belongs to Tür who has adapted the sentence determination system to his work on information extraction from Turkish texts using a statistical language processing model (Tür, 2000). The Segmentation System aimed to divide a given set of words into sentences in a syntactic context. Using web resources, a dataset consisting of the articles of Milliyet newspaper from 1 January 1997 to 12 September 1998 was used. The f-measure in this study, which was carried out with the help of supporting elements, was stated as 91.35%.

Another study focusing on the uncertainty of the point character was made by Dinçer and Karaoğlan (Dinçer and Karaoğlan, 2004). In the study, which does not need any training data, simple intuitive information about the syllable and the

phonetic rules of the language were used. Since the boundary of sentence character is generally used as period and question mark in Turkish, a rule list for these characters has been prepared and it is aimed to eliminate the ambiguity with this rule list in the tests. It has been reported that a 96.02% of f-measure was achieved in the tests performed using a dataset collected from Milliyet newspaper articles.

Aktaş and Cebi, on the other hand, suggested a rule-based sentence boundary determination method with the thought that rules and abbreviations in Turkish cause sentence ambiguity (Aktaş and Çebi, 2013). The system works by using predetermined rules and an abbreviation list. The f-measure obtained in the tests made from newspaper texts was published as 99.80%.

Another study is the work of Özbey and Dinçsoy, which focused on frequently encountered quotes in newspaper texts (Özbey and Dinçsoy, 2019). According to the experimental analysis done over the newspaper documents collected from the web, 98.34% of the ambiguity at the boundary of the sentence, including quotations, were determined with a single regular expression.

In the study of Schweter and Ahmed mentioned in the previous section, a Turkish corpus was tested (Schweter and Ahmed, 2019). Three neural network architectures, LSTM, BiLSTM, and CNN, were used in the study. Experiments were conducted with SETimes Corpus, which covers Balkan languages, including English, German and Arabic, as well as Turkish. Their method uses the character contexts of the tokens before and after the potential end-of-sentence marker (Tyers and Alperen, 2010). As a result of these experiments, 98.56%, 98.58%, and 98.54% of f-measure results were obtained for LSTM, BiLSTM, and CNN networks for Turkish, respectively.

A summary regarding on Turkish corpus and SBD methods developed for Turkish is shown in Table 2.2.

Table 2.2. Studies and suggested SBD methods for Turkish

| Study | Method | Dataset (Corpus) | Reported Success Rate (F-Measure) |
|---|---|---|---|
| **Tür(2000)** | Statistical | Web(Milliyet) | 91.35% |
| **Dinçer and Karaoğlan(2004)** | Rule-based | Web(Milliyet) | 96.02% |
| **Aktaş and Çebi(2013)** | Rule-based | Web(Newspaper) | 99.80% |
| **Özbey and Dinçsoy(2019)** | Regular expression | Web(Newspaper) | 98.34% |
| **Schweter and Ahmed(2019)** | LSTM | SETimes Corpus | 98.56% |
| | BiLSTM | | 98.58% |
| | CNN | | 98.54% |

## 2.3. Motivation of the Thesis

As mentioned in the previous sections, the studies on the boundary of sentence detection process have been mostly tested with English corpus, even if they are general purpose techniques that can be applied for other natural languages. In particular, supervised systems have been trained or tested with text documents collected from a specific domain and specific format due to the lack of sufficient corpus in natural languages. In the studies conducted for Turkish, researchers generally use either newspaper articles collected over the internet or METU Corpus, which consists of 53.24% of newspaper news texts (Say et al, 2002). Read et al. also stated that there are deficiencies in species and site-specific diversity in previous studies (Read et al, 2012). In addition, they measured poor performance in variations outside the standard structure of the official language and in the transition to a less formal language.

In our previous study, we compared existing SBD systems on ten million-word sub-corpus of the Turkish National Corpus (TNC), which was developed for Turkish language studies (Aksan et al, 2014). In this study, well-known general systems that are Julie Sentence Boundary Detector (JSBD) (Tomanek et al., 2007), GeniaSS (Kim et al., 2003), and SplittA (Gillick, 2009), and Turkish specific SBD system designed by Dokuz Eylül University Natural Language Processing Research

Group (9EDDİ) were tested and compared for SBD of Turkish texts using the TNC sub-corpus. The methods were tested with sub-text collections belonging to eight different domains. Results from the entire TNC subcollection are given in Table 2.3. According to the experimental analysis, we have observed that although the GeniaSS system seems to be the most successful with a result of 88% accuracy, this result is far from the mentioned values in sections 2.1 and 2.2. We have also seen that although the GeniaSS was the software that produced the highest number of correct sentences, 9EDDI software gave better results in text documents from Social Sciences, World Problems, Thought-Belief, and Freelance fields.

Table 2.3. Performance of Sentence Boundary Detection Software on TNC Sub-corpus (Aksan et al, 2014).

| **Software** | **Total Number of Sentences** | **Number of Correct Sentences** | **Accuracy Rate** |
|---|---|---|---|
| **JSBD** | 690,998 | 539,628 | 70% |
| **Splitta** | 664,769 | 171,467 | 22% |
| **GeniaSS** | 893,401 | 681,850 | 88% |
| **9EDDİ** | 683,609 | 576,920 | 75% |

As a result of these obtained results and problems mentioned, it has been understood that more effective sentence boundary determination systems are needed for Turkish texts. While developing a machine learning-based or rule-based method for determining the boundary of a sentence, it was thought that working with a corpus capable of representing the language, such as the TNC sub-corpus, would help us to develop more effective systems.

With the proposed method, the effect of Turkish affixes and species as well as their syntax in determining the boundary of sentence has been examined. In the current deep learning studies, sentences are handled as sequence data, and word or character information in the sentences are used. In this study on the other hand, unlike the entities themselves or the character information, Turkish-specific suffix

structure and sequence were tested in the deep learning method. In addition, contrary to previous studies, the TNC sub-collection, which is balanced and has many special language uses, has been used for testing the deep learning techniques in which this sequencing is applied. As mentioned above, the success rate of this corpus, which has a low success rate with the existing systems, was found to be higher in the model proposed in our study.

# 3. MATERIALS

## 3.1. Sub-Corpus of Turkish National Corpus (TNC)

The dataset used in the study is a sub-corpus of the Turkish National Corpus (TNC) used in the same distribution criteria (Aksan, 2012). TNC belongs to a project that aims to create a large-scale general corpus of contemporary Turkish. It consists of 50 million words and covers a period of 20 years (1990-2009). A large part (98%) of TNC consists of textual examples. The rest is a written version of the transcribed spoken data. The TNC sub-corpus is a corpus that has approximately 10 million words sampled with the same distribution criteria as the TNC, and the sentence endings are labeled (Demirhan, 2013). The distribution of the TNC sub-corpus by fields is shown in Figure 3.1.

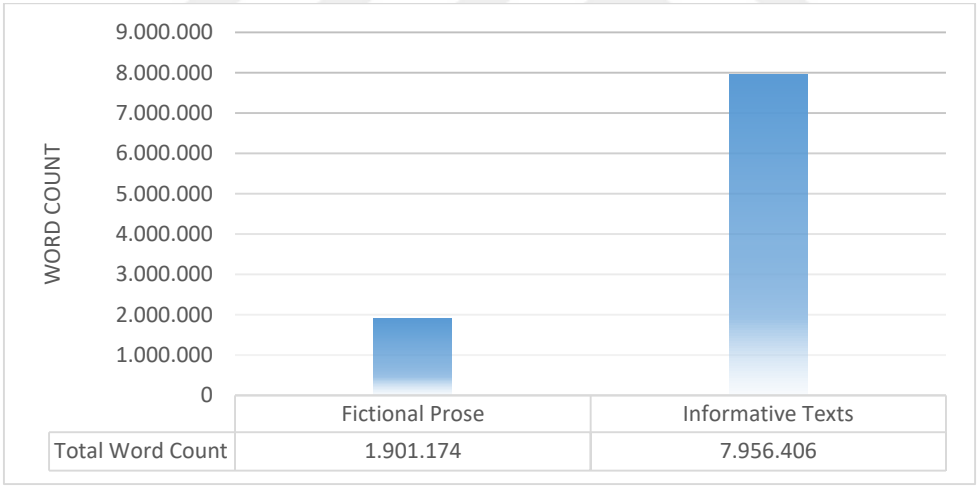


Figure 3.1. TNC Sub-Corpus Document Distribution by Fields

The distribution of the documents in the fictional prose field in the TNC sub-corpus according to their text formats is given in Figure 3.2. The distribution of informative texts by media is shown in Figure 3.3.

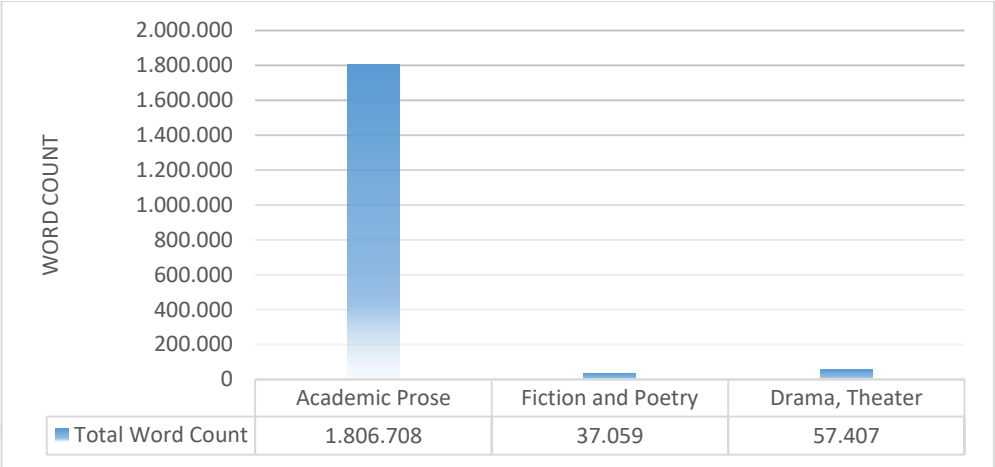


Figure 3.2. Distribution of Documents in Fictional Prose Field by Text Format

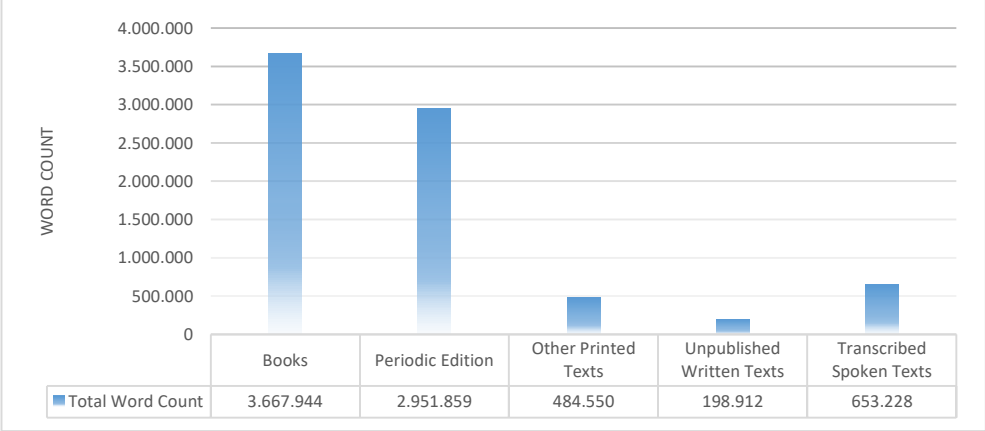


Figure 3.3. Distribution of Documents in Informative Texts Field by Media

The distribution of the documents in informative text group in the TNC sub-corpus by fields is shown in Figure 3.4. As shown in Figure 3.2, 3.3 and 3.4, the TNC sub-corpus contains different type of documents collected from books, periodicals, published and unpublished written texts, transcribed spoken texts, in domains natural and pure sciences, applied sciences, social sciences, world affairs, arts, belief and thoughts, leisure, commerce and finance. Therefore, the TNC sub-corpus has various different types of sentences that can be observed in Turkish language.

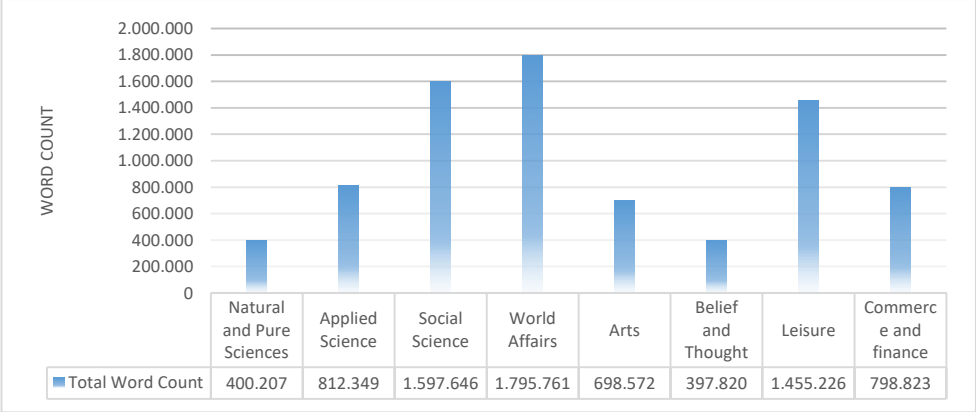


Figure 3.4. Distribution of Informative Texts by Media

All experiments, which will be discussed later, were performed with a randomly specified number of samples of the TNC sub-corpus. In the relevant section, the prepared data format according to the method used is explained in detail.

### 3.2. Genia Treebank Corpus

Another corpus used in this study is Genia Treebank Corpus version 1. Genia Corpus is in English and consists of 1999 Medline summaries. The linguistically marked text-based natural language processing tool was created for the purpose of biomedical text mining (Tateisi et al, 2005). In the corpus in XML format, all unit features are marked beside the end of the sentence. There are 18541 sentences in the corpus, which contains a total of 21079 potential boundary of sentence markers, and there are 2538 data samples that look like sentences but do not indicate the boundary of sentences. Agarwall et al. have obtained an F1 score of 98% in their study using the Genia corpus (Agarwal et al., 2005).

### 3.3. Lob Corpus

The other corpus used in a part of this study is the Lob Corpus. The original Lob Corpus, developed under the leadership of the University of Oslo, was published in 1976, and the POS-tagged version was published in 1986 (Lob, 1986). Lob

Corpus, which was written as a counterpart to the widely used Brown Corpus, which was produced in American English, is entirely in British English (Kucera and Francis, 1979). Lob Corpus, which has more than 1 million words in total, has been prepared according to the same criteria as Brown corpus, taking into account the same word species and genus characteristics. In our study, a Lob sub-corpus consisting of 17 texts was used. The Lob sub-corpus we have used contains a total of 64263 potential sentence boundaries. While 54043 of these potential sentence endings are actual sentence endings, 10220 of them consist of markers that do not actually point to sentence boundary.

### 3.4. SETimes Corpus

In particular, SETimes Corpus is a parallel corpus composed of news texts based on the content published on the www.setimes.com news site, which also includes a Turkish corpus (Tyers and Alperen, 2010). It is a corpus with texts in 9 southeastern European languages along with Turkish and English texts.

# 4. METHODS

In this thesis, SBD of Turkish texts were performed in three ways: First, we applied classical supervised machine learning methods to datasets that have features obtained by applying heuristic rules. Second, we used word embeddings to obtain word vectors for each word in the sentence, then we considered each sentence as a sequence and applied deep learning based methods. Third, we applied association rule mining to find associations among POS tags, suffixes, prefixes and end of sentence characters to enlarge our feature set, then we applied deep learning methods for classification.

For the first method, Back Propagation Neural Network (BPNN), Radial Basis Function Network, Naive Bayes Classification, Decision Trees, and SVM methods from classical classifiers were used to evaluate the success of the rule-based dataset created in this thesis. In addition to these classifiers, the LSTM method, which is one of the RNN methods, was also tested with a rule-based dataset.

In the second method, CNN was used to classify the dataset created by using the word vectors. The same dataset was also tested with the BERT method for comparison with CNN.

In the third method, the FP-Growth algorithm was used to determine the associations of the additional information of the tokens and the boundary of sentence characters. With the results obtained from here, a sequential dataset was obtained, in which token type information was added. This dataset has been tested with RNN methods named LSTM and Bidirectional LSTM (BiLSTM).

## 4.1. Classical Classifiers

This section includes classical classifiers that were used for SBD from the rule based dataset.

### 4.1.1. Back Propagation Neural Network

The Back Propagation Neural Network (BPNN) method is a multilayer feedforward network trained according to the error backpropagation algorithm and is one of the most widely applied neural network models (Rumelhart et al., 1985). When we use a feedforward neural network to accept an input $x$and produce an output $y'$, information flows forward through the network. The $x$ input provides information that is then propagated to the hidden units in each layer and finally produces $y'$. This is called forward propagation. During training, forward propagation may continue until the scalar produces a cost $J(\theta)$. The back propagation algorithm then allows information from the cost to flow back through the network to calculate the gradient (Rumelhart, 1986). The general operation of BPNN is shown in Figure 4.1.

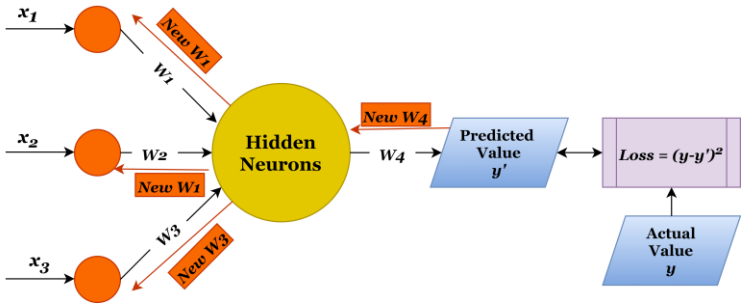


Figure 4.1. The Architecture of Back Propagation

### 4.1.2. Radial Basis Function Network

Radial basis function network (RBFN) is an artificial neural network that uses radial basis function (RBF) as activation function. RBFN consists of input layer, hidden layer and output layer, similar to feed forward Artificial Neural Networks (ANN) structures (Broomhead et al, 1988). The transformation from the input layer to the hidden layer is a nonlinear constant transformation with the RBF activation function. A linear transformation takes place from the hidden layer to the output

layer. Free parameters that can be adapted in RBFN are center vectors, width of centripetal functions, and output layer weights. The mathematical expression of the radial basis function used in RBFN is given in formula 4.1.

$$\emptyset_j = exp\left[\frac{-\|x-c_j\|}{\sigma_j^2}\right] \quad (4.1.)$$

where $x$ denotes input vector,$c_j$ is the $j$ th gaussian function center, $\sigma_j$ is the value of standard deviation. $\|x - c_j\|$ expression shows the Euclidean distance between $x$ and $c_j$ vectors. The activation level of the node $j$ is equal to $\emptyset_j$. The general architecture of RBFN is shown in Figure 4.2.

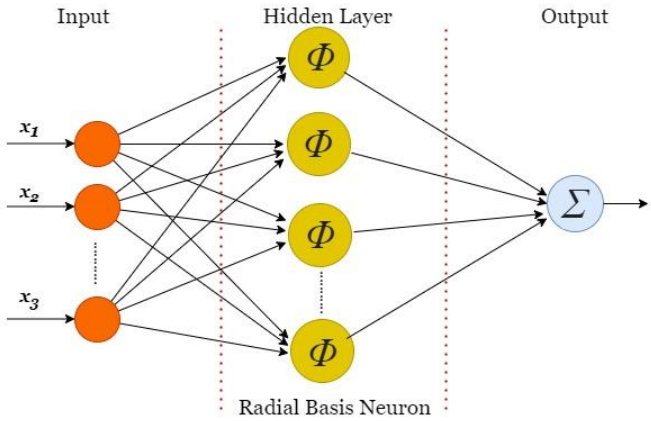


Figure 4.2. The architecture of RBFN

### 4.1.3. Naive Bayes Classification

Bayesian classifiers offer the supervised learning method together with the statistical method for classification. It assigns a particular instance defined by the feature vector to the most probable class (Hand and Keming, 2001). Understanding such classifiers can be greatly simplified by assuming that all features are independent of their class labels. Despite this unrealistic assumption, the resulting

classifier known as Naive Bayes is remarkably successful in practice and often competes with much more complex techniques (Hilden, 1984). Naive Bayes is known to be effective in many practical applications, including text classification, medical diagnostics, and system performance management.

Naive Bayes classifier is a simplified version of Bayes' theorem with an independence proposition. Bayes' theorem is expressed by the following equation;

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (4.2.)$$

Here a priori probability adds subjectivity to Bayes' theorem. In other words, for example, $P(A)$ is the information about event *A* before data is collected. On the other hand, $P(B|A)$ is a successive probability because after data collection, it gives information about the probability of event *B* in cases where event *A* has occurred (Pawlak, 2003).

The formula in Eq.4.2 is used to decide whether or not any sample belongs to which class. If the statistical independence proposition is used in the mentioned Bayes decision theorem, this type of classification is called Naive Bayes classification.

Although the usage area of the Naive Bayes classifier seems limited, successful results can be obtained by stretching the condition that the components of $x$ (feature set) are statistically independent in high-dimensional space and with sufficient data.

### 4.1.4. Decision Trees

One of the most used methods in classification with Data Mining is decision trees. In the structure of the decision tree, each node represents a feature. Branches and leaves are elements of tree structure (Quinlan, 1986). The top element is called the root, the bottom element is called the leaf, and the elements between them are called intermediate nodes. In the decision tree, the rules are written down from root

to leaf in the form of IF-THEN rules. In the decision tree method, action is taken according to the answer of the question in concluding an event.

There are applications of decision tree algorithms that have been successfully applied in many different areas. The decision tree induction application we used in our tests is the C4.5 algorithm. The C4.5 algorithm is one of the most well-known decision tree induction algorithms (Quinlan, 1993). In the C4.5 algorithm, the information gain ratio is used as the test feature selection criterion and for each set, the feature with the highest information gain ratio is selected. The C4.5 algorithm is a method based on the ID3 algorithm and removes some of the limitations of this algorithm (Quinlan, 1993). C4.5 algorithm can work with both continuous and discrete features. In addition, it can work with training datasets containing missing feature values. Also, it eliminates the overfitting problem by pruning some nodes or subtrees during or after the decision tree creation and ensures the removal of exceptional and noisy values in the training set (Niuniu et al., 2010).

### 4.1.5. Support Vector Machine

One of the most successful machine learning algorithms developed for the solution of two-group classification problems, which is frequently used, is Support Vector Machines (SVM) (Cortes and Vapnik, 1995). SVM has been successfully applied in solving many classification problems and has taken its place in the literature as one of the efficient and effective machine learning algorithms with high generalization performance.

The most important advantage of Support Vector Machines is that it transforms the classification problem into a quadratic optimization problem and solves it. Thus, in the learning phase of the solution of the problem, the number of operations decreases, and a faster solution is achieved compared to other techniques/algorithms (Nitze et al., 2012). Due to this feature of the technique, it provides a great advantage, especially in large volume datasets. In addition, since it

is based on optimization, it is more successful than other techniques in terms of classification performance, computational complexity and usefulness.

Kernel function selection and parameter optimization play an important role in the implementation of Support Vector Machines for solving the classification problem for various datasets. It has been seen that the RBF Kernel function is used in the applications in the literature, with the thought that it generally gives better results. In this study, tests were carried out by selecting the standard parameters and RBF Kernel in the Weka software.

## 4.2. Deep Neural Networks

The neural network is a model for machine learning inspired by the human brain. As seen in Figure 4.3., it is a large network consisting of many layers and neurons. It is a field of study that covers artificial neural networks and similar machine learning algorithms with one or more hidden layers. Deep learning can be performed supervised, semi-supervised or unsupervised. Bengio et al. used a neural network for language modeling in 2003 and achieved better results than n-gram models (Bengio et al., 2003). Neural networks have a flexible structure. It can have layers with various numbers of nodes and various numbers of hidden layers linked by weight values. The more complex layers a neural network has, the more learning capacity it has. Neural networks with more than one hidden layer are called Deep Neural Networks (DNN).

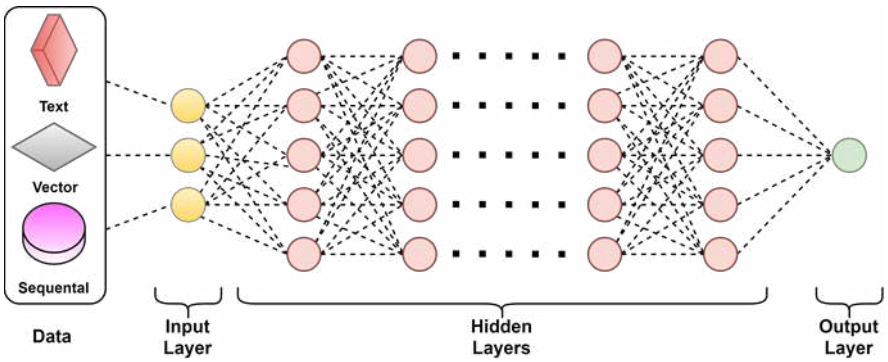


Figure 4.3. Deep Neural Network Architecture

### 4.2.1. Recurrent Neural Network

Recurrent Neural Network (RNN), one of the deep learning architectures, is applied by Mikolov et al. on the speech recognition problem in 2010 (Mikolov et al., 2010). The results show that RNN is better than n-gram techniques. The advantage of RNN in language modeling is that the previous state is used to calculate its current state; this is similar to the context of most natural languages. Because the idea behind RNNs is to use sequentially ordered information. In traditional neural networks, all inputs and outputs are assumed to be independent of each other. But for many transactions, this is not a good idea. In order to predict what the next word in a sentence is, it is important what the previous words are and what sort of sequence they have. RNNs are called recurrent because the calculation of the value of each element of the array depends on previous calculations. A typical RNN architecture is shown in Figure 4.4 (Wiki, 2021). If the figure is to be examined in order, the layers are;

Input : $x_t$, is taken as input to the network at time *t*. For example, $x_1x(1)$ can be a single active vector corresponding to a word of a sentence.

Hidden State : $h_t$ represents a hidden state at time *t* and serves as the "memory" of the network. $h_t$ is calculated based on latest input and the latent state of the previous time step: $h_t = f(Ux_t + Wh_{t-1})$. The *f* function is taken as a non-linear transformation such as tanh, ReLU.

Output : $o_t$ , represents the output of the network.

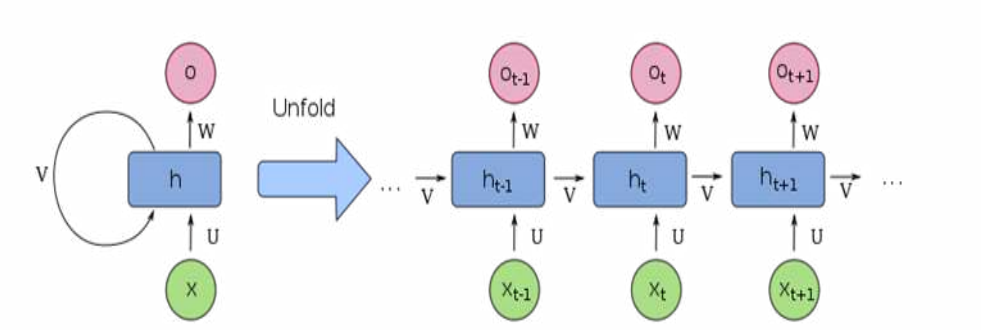


Figure 4.4. Recurrent Neural Network Architecture (Wiki, 2021)

#### 4.2.1.1. Long Short Term Memory

The Long Short Term Memory (LSTM) architecture (Hochreiter and Schmidhuber, 1997), (Gers et al., 2002), was revealed as a result of an analysis of the error stream in existing RNNs. Long time delays were found to be unattainable for current architectures because the back propagated error in RNNs is exponentially reduced or completely corrupted.

An LSTM layer consists of a series of repeatedly linked blocks known as memory blocks. These blocks can be thought of as a differentiable version of the memory chips in a digital computer. Each contains three multiplicative units with input, output, and forget gates that provide continuous write, read, and reset operations for one or more repetitively connected memory cells. More precisely, the input to cells is multiplied by the activation of the input gate, the net output is multiplied by that of the exit gate, and the previous cell values are multiplied by the forget gate. The network can interact with cells only through gates.

Figure 4.5 shows the inputs and outputs of an LSTM for a single timestep. The LSTM has an input $x_t$ which can be the output of a CNN or a direct input sequence. $h_{t-1}$and $c_{t-1}$ are inputs from the previous timestep of the LSTM.$o_t$ is the output of the LSTM for this timestep. LSTM also generates $c_t$ and $h_t$ for consumption of the next step of LSTM.

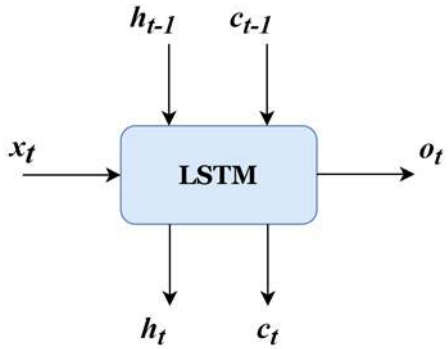


Figure 4.5. LSTM for a single timestep

The below equations are used to compute input, output, forget, hidden, and cell states for a single LSTM unit at time *t*.

$$f_t = \sigma_g(W_f \times x_t + U_f \times h_{t-1} + b_f) \quad (4.3.)$$

$$i_t = \sigma_g(W_i \times x_t + U_i \times h_{t-1} + b_i) \quad (4.4.)$$

$$o_t = \sigma_g(W_o \times x_t + U_o \times h_{t-1} + b_o) \quad (4.5.)$$

$$c'_t = \sigma_g(W_c \times x_t + U_c \times h_{t-1} + b_c) \quad (4.6.)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t \quad (4.7.)$$

$$h_t = o_t \cdot \sigma_c c_t \quad (4.8.)$$

where $f_t$ is forget gate, $i_t$ is input gate, $o_t$ is output gate, $c_t$ is cell state, $h_t$ is hidden state, $\sigma_g$ is Sigmoid and $\sigma_c$ is Tanh functions. It should be noted that the LSTM equations also generate the values for$f_t, i_t, c'_t$. These are for internal consumption of LSTM and are used to produce$c_t$ and $h_t$.

LSTM has been applied to real world string processing problems in many studies. In particular, there are many studies on isolated word recognition and continuous speech recognition based on LSTM.

#### 4.2.1.2. Bidirectional Long Short Term Memory

Bidirectional LSTM (BiLSTM) is a recurrent neural network primarily used in natural language processing. Unlike standard LSTM, input flows in both directions and can use information from both sides. It is also a powerful tool for

modeling cascading dependencies between words and phrases in both directions of the sequence (Schuster and Paliwal, 1997).

In summary, BiLSTM adds another layer of LSTM that reverses the direction of the information flow. Therefore, the input sequence flows backwards in the additional LSTM layer.

Then, the outputs from both LSTM layers are combined in various ways such as mean, sum, multiply, or combine. BiLSTM sampling is presented in Figure 4.6

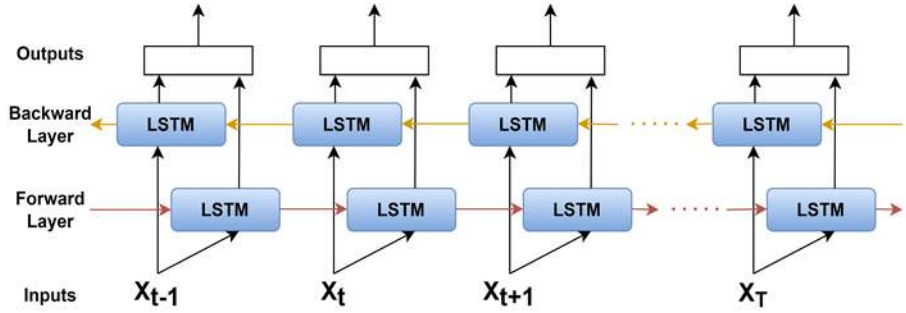


Figure 4.6. Illustration of BiLSTM

This type of architecture has many advantages in real world problems, especially in NLP. The main reason for this is that each component of an input sequence has meaningful information both forward and backward. Therefore, BiLSTM can produce a more meaningful output by combining the LSTM layers from both directions.

### 4.2.2. Convolutional Neural Network

The terms Deep Learning or Deep Neural Network refer to multi-layer Artificial Neural Networks (ANN). Networks with deeper hidden layers have recently begun to surpass the performance of classical methods in different fields, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Network (CNN). It gets its name from the mathematical linear operation between matrices called convolutions. CNN has multiple layers; these are

the convolutional layer, the nonlinear layer, the pooling layer, and the fully connected layer (Figure 4.7). Convolutional and fully connected layers have parameters, but pooling and nonlinear layers have no parameters. CNN has excellent performance in machine learning problems.

Especially for the largest image classification dataset (Image Net) and for applications dealing with image data such as computer vision, the results are very successful (Rastegari et al., 2016) (Abdel-Hamid et al., 2013).

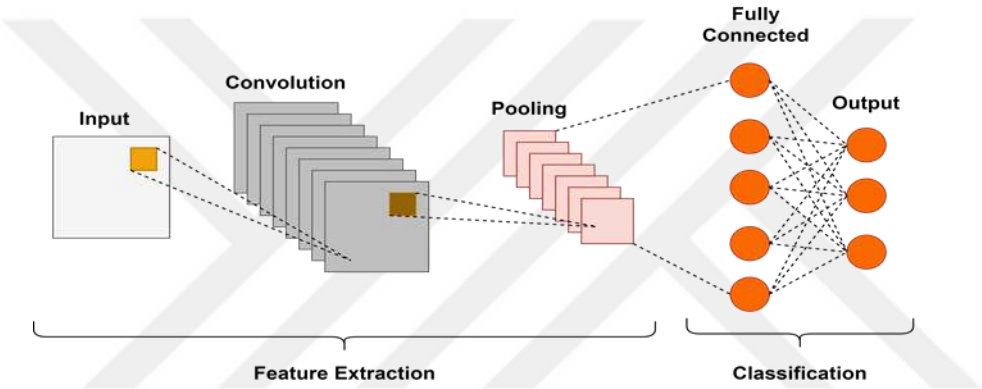


Figure 4.7. Schematic diagram of convolutional neural network (CNN)

CNNs are customized for image processing and give excellent results in this regard (Krizhevsky et al., 2012)( Maggiori et al., 2017). However, apart from image processing, CNNs are used in sentiment analysis, question classification (Kim, 2014), part-of-speech and named entity tagging, and semantic similarity (Collobert and Weston, 2008). It is also widely used in NLP applications such as sentence boundary detection (Che et al., 2016) and word recognition.

### 4.2.3. Word Level Information Vectors

Word vectors are a method that has been developed recently, works in an unsupervised way, and its use in natural language processing applications has increased considerably in recent years. The basis of these vectors is based on Word

Embedding methods. Word representation methods are a language modeling technique that digitizes words or phrases and turns them into vectors.

Word level information vectors (Word2Vec) method is a prediction-based word representation method and was developed with the aim of training words using two different NN models (Mikolov et al., 2013). The two models Word2Vec uses are CBOW (Continuous Bag of Words) and Skip-Gram Model. In the CBOW model, the words that are not in the window size center are taken as input and the words in the center are tried to be estimated as output. In the Skip Gram model, the words in the center of the window size are taken as input, and the words that are not in the center are tried to be estimated as output (Figure 4.8). With the Word2Vec method, all words in a given text are expressed with vectors in a defined space. With the help of these vectors created, the similarity or contrast between the desired words etc. can be expressed mathematically. Studies using Word2Vec include sentiment analysis, text classification, text representation, etc.

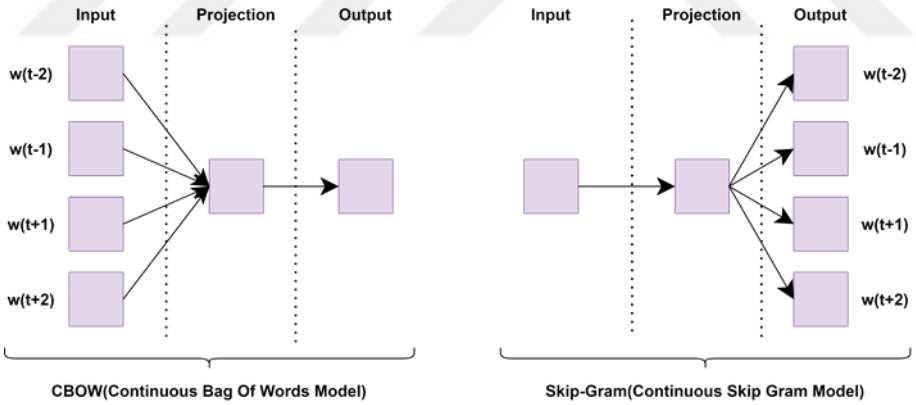


Figure 4.8. Models of Word2Vec

As an example, Tang et al. performed sentiment analysis on tweets using the Word2Vec method. In the sentiment analysis study performed with various n-grams and the Word2Vec method, 77.33% accuracy value is achieved (Tang et al., 2014). Polpinj et al. developed a Word2Vec-based system for the classification of

comments made on a hotel's web page. SVM algorithm was used for classification of comments where each word is represented by word vectors computed by Word2Vec (Polpinij et al., 2017). Şahin has designed a system that builds models on Turkish texts with Word2Vec and classifies them with SVM. In the study, classification was carried out on 22,729 documents (Şahin, 2017). Ma et al. proposed a TF-IDF, Word2Vec and Doc2Vec based system for semantic similarity analysis in course selection and measured their performance in course selection (Ma et al., 2017).

### 4.2.4. Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a converter-based machine learning technique for NLP pre-training developed by Google. BERT was created and published by Jacob Devlin and colleagues from Google in 2018 (Devlin et. al, 2018). BERT is a method of pre-training language representations used to build models that NLP practitioners can use for free. You can use these models to extract high-quality language features from text data or fine-tune these models for a specific task (e.g., classification, named entity recognition, question answering, etc.) to reveal new advanced prediction systems.

BERT uses transformer, an attention mechanism that learns the contextual relationships between words (or subwords) in a text (Vaswani et. al, 2017). The transformer includes two separate mechanisms: an encoder that reads the text input and a decoder that generates a prediction for the task. Since the purpose of BERT is to build a language model, it uses only the encoder mechanism. Unlike directional models, which read text input sequentially (left to right or right to left), the transformer encoder reads the entire string of words at once. Therefore, although the model is unidirectional, it can be accepted as bidirectional. This feature allows the model to learn the context of a word based on its entire circumference (left and right of the word).

A generalized description of the transformer encoder is given in Figure 4.9. The input is a set of tokens that are first embedded in vectors and then processed in the neural network. The output is an array of vectors where each vector corresponds to an input token with the same index.

Many models predict the next word in a sequence. This method, by its nature, is a directional approach that limits context learning. To meet this challenge, BERT uses two training strategies. The subsections that follow examine these strategies.

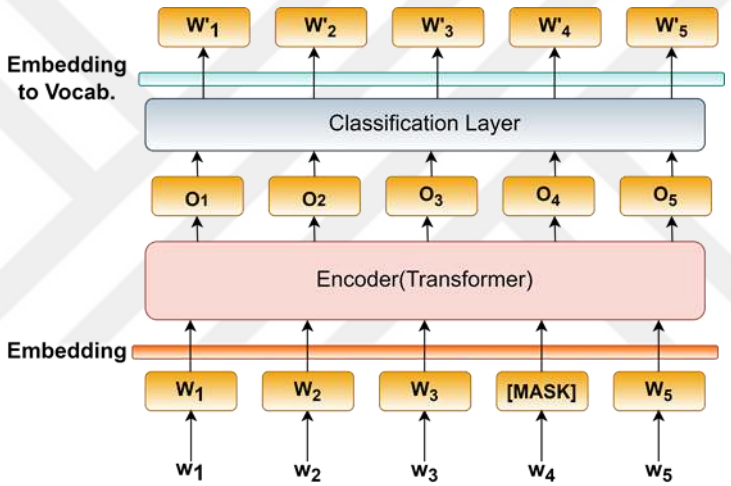


Figure 4.9. Diagram of Transformation

#### 4.2.4.1. Masked LM (MLM)

Before presenting the word strings as input to BERT, 15% of the words in each string are replaced with a [MASK] token. The model then tries to predict the original value of the masked words based on the context provided by the other unmasked words in the array. A generalized description is given in Figure 4.10. From a technical point of view, the estimation of the output words requires the following steps;

- Adding a classification layer on top of the encoder output.
- Multiplying the output vectors with the placement matrix, converting them to word size.
- Calculating the probability of each word in the vocabulary with Softmax.

The loss function of BERT only considers the prediction of masked values and ignores the prediction of unmasked words. As a result, the model approaches the result more slowly than the directional models. This is a feature that is offset by increased context awareness.

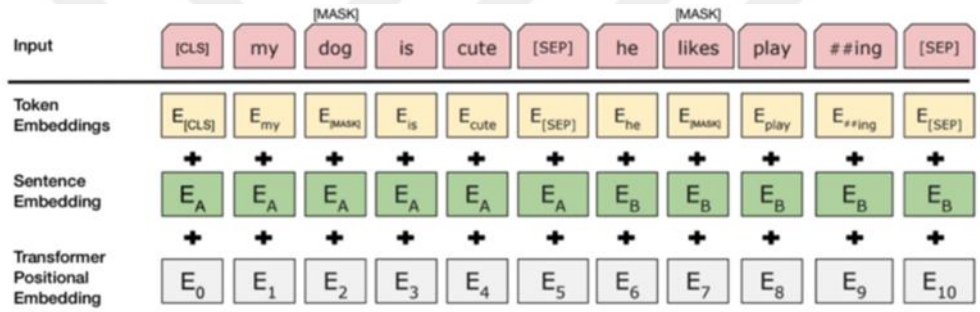


Figure 4.10. BERT (Devlin et al., 2018), with modifications

#### 4.2.4.2. Next Sentence Prediction (NSP)

During the BERT training process, the model takes pairs of sentences as input and learns to predict whether the second sentence is the next sentence in the original document. During training, 50% of the entries are a pair in the original document, where the second sentence is the next sentence. In the other 50%, a random sentence from the corpus is chosen as the second sentence. The assumption here is that the random sentence will be disconnected from the first sentence.

To help the model distinguish between the two sentences in the training, the input is processed as follows before entering the model:

- A [CLS] token is added at the beginning of the first sentence and a [SEP] token at the end of each sentence.
- A sentence indicating Sentence A or Sentence B is added to each token. Sentence insertions are similar in concept to token insertions with a vocabulary of 2.
- A positional overlay is added to each icon to indicate its next position.

To understand whether the second sentence really depends on the first, the following steps are performed:

- The entire input sequence goes through the Transformer model.
- The output of the [CLS] token is converted to a 2x1 shaped vector using a simple classification layer (learned weights and biases matrices).
- IsNextSequence probability is calculated with Softmax.
- While training the BERT model, Masked LM and Next Sentences Prediction are trained together to minimize the combined loss function of the two strategies.

### 4.3. Association Rules and Frequent Pattern Growth Algorithm

Mining association rules are a very important problem in the field of data mining. It involves defining common sets of items and then creating applicable rules that apply between them. This knowledge is useful in improving the quality of many business decision-making processes, such as customer buying behavior analysis, cross-marketing, and catalog design.

Association rules are described as follows: Let $I = \{i_1, i_2, i_3 \dots \dots i_n\}$ be the set of our items. Let each transaction consisting of a subset of items in $I$ be named $T$, and let our dataset consisting of many transactions be $D$. Each subset of $I$ is called $X$, and subsets containing $k$ items are expressed as $X$ with *k-itemsets* (Agrawal et. al, 1993). The support value of any $X$ itemset in a transaction is shown as $\sup(X)$. If

$\sup(X)$ is not less than a user defined *minimum support*, then $X$ is called a frequent itemset. An association rule is a conditional inference between sets of items $X$ and $Y$ in the form of $X \Rightarrow Y$ where $X \subseteq I, Y \subseteq I, X \neq \emptyset, Y \neq \emptyset$ and $X \cap Y \neq \emptyset$.

The confidence of the association rule given as $\sup(X \cup Y) / \sup(X)$ is the conditional probability between $X$ and $Y$. The problem of association rule mining is the discovery of association rules with greater support and confidence than *minimum support* and user-defined *minimum confidence*.

Discovering frequent itemsets is the most computationally intensive step of the association rule mining phases. The biggest challenge is that mining often needs to generate a large number of candidate itemsets. For example, if there are $n$ items in the database, in the worst case, all $2^n$ candidate itemsets need to be created and examined. The rule creation step is simple and less expensive. Therefore, most researchers focus on the first step to find common itemsets (Agrawal et al., 1993, Zaki et al., 1997).

Apriori algorithm was proposed by Agrawal and Srikant to solve the frequent item sets mining problem (Agrawal and Srikant, 1994). Apriori uses a candidate generation method such that frequent $k$-itemsets in one iteration can be used to generate candidate ($k$+1)-itemsets for the next iteration. Apriori terminates its process when new candidate itemsets can not be generated.

Unlike Apriori, the Frequent Pattern Growth (FP-growth) method (Han et al., 2004 , Han et al., 2000) uses an FP-tree to store the frequency information of the transaction database. Without candidate itemset generation, FP-growth uses a recursive divide-and-conquer and database projection approach to find frequent itemsets.

After determining the associations, values such as Confidence and Lift are checked for the rules created. The purpose here is to determine the validity of the created rules.

### 4.3.1. Confidence

Let's assume that we have $X$ and $Y$ itemsets related to the transaction dataset we have. If $X \Rightarrow Y$ represents an association rule; *Confidence* is expressed as the percentage that $Y$ is met in all trades that satisfy $X$ (Wong et. al, 1999). In other words, the confidence value of an association rule denoted as $X \Rightarrow Y$ is the ratio of trades containing both $X$ and $Y$ to the total amount of $X$ values available; where $X$ is the premise and $Y$ is the conclusion.

It is usually expressed as:

$$conf(X \Rightarrow Y) = \frac{\text{number of transaction containing X and Y}}{\text{number of transaction containing X}} \quad (4.9.)$$

Shortly:

$$conf(X \Rightarrow Y) = \frac{\sup(X \cap Y)}{\sup(X)} \quad (4.10.)$$

### 4.3.2. Lift

It is the ratio of observed support of $X$ and $Y$ to their expected support. Lift formula is as follows:

$$lift(X \Rightarrow Y) = \frac{\sup(X,Y)}{\sup(X) \times \sup(Y)} \quad (4.11.)$$

If the rule has a lift of *1*, it would mean that the probability of occurrence of the antecedent and contecedent are independent of each other. When two events are independent of each other, no rule involving these two events can be mentioned.

If *Lift > 1*, it lets us know to what extent these two occurrences are dependent on each other and tells us that these rules will potentially be useful for predicting the outcome in future datasets.

If *Lift* < *1*, this allows us to understand that the items are interchangeable. This means that the presence of one substance negatively affects the presence of the other and vice versa. The importance of the lift value is that it takes into account both the support of the rule and the overall dataset.

### 4.4. Evaluation Metrics

In this thesis, accuracy, precision, recall, and F-measure values were used to evaluate the performance of classifiers for detecting the end of sentence (Han and Kamber, 2006; Liu, 2011; Mundluru, 2008). These evaluation metrics, which are frequently encountered in information retrieval domain, are related to the number of samples assigned to the correct class and the large number of samples assigned to the wrong class. Performance evaluation obtained as a result of the tests expressed by the confusion matrix shown in Figure 4.11. In the confusion matrix, the rows show the actual numbers of the samples in each class in the test set, and the columns show the model's prediction.

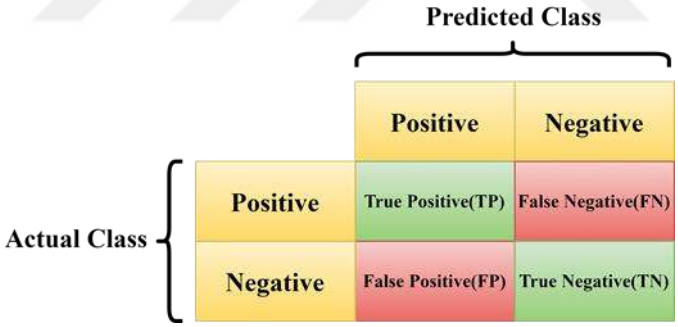


Figure 4.11 Confusion Matrix

#### 4.4.1. Accuracy

The most popular and simple method used to measure the performance of the classification model is the accuracy rate of the model. It is the ratio of the number of correctly classified samples $(TP + TN)$ to the total number of samples $(TP + TN + FP + FN)$ (Han and Kamber, 2006). The worst problem in measuring

performance with accuracy value is that the data is unbalanced. Misleading results can be obtained in case of large imbalances in class distributions (Japkowicz, 2000).

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (4.12.)$$

### 4.4.2. Precision

Precision is the ratio of the number of True Positive samples with the Positive class predicted as true to the number of all samples with the class predicted as Positive (Han and Kamber, 2006).

$$Precision = \frac{TP}{TP+FP} \quad (4.13.)$$

### 4.4.3. Recall

It is the ratio of the number of correctly classified positive samples to the total number of positive samples (Han and Kamber, 2006).

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \quad (4.14.)$$

### 4.4.4. F-Measure

Precision and recall measures alone are not sufficient to draw a meaningful comparison result. Evaluating both criteria together gives more accurate results. For this, the F-measure value is defined. The F-measure is the harmonic mean of precision and recall. It is used to get more valid results when the class distribution is not balanced (Han and Kamber, 2006).

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad (4.15.)$$

### 4.5. Proposed Method

In this thesis, sentence boundary detection problem is solved by converting the problem into binary classification problem. Given a text to be processed, first we find characters that can show end of a sentence. For each such a character, we create features (such as POS tags, suffixes, prefixes) from the *n* tokens before and after the character. Then we apply classification to determine whether the character shows the end of a sentence or not. The POS tag information of the tokens to be considered was determined by the tests presented in the thesis that they formed together with the boundary of sentence character. In addition to this, it has been revealed that the additional information that the tokens have to support this also provides a unity with the boundary of sentence character. The difference of the proposed method is that at this point, a sequence data consisting of POS tag information and additional information of the tokens before and after the sentence boundary characters is used with deep learning methods.

The operating principle of the system is presented in Figure 4.12. To explain this, it can be said that the system is operated in three basic stages. For this, the corpus is brought into a data format that can be used in deep learning after some preprocessing. In this regard, after the corpus reading, the boundary of sentence characters and whether these characters are the end of the sentence or not are marked. Then, the formal features of the token information are determined. And a token vector is obtained by marking each token's POS tags and additional information detected by the FP-Growth method. These token vectors are arranged according to the desired number of sequences and the desired sequential data is obtained.

The creation and training of the RNN deep learning network with the BiLSTM architecture shown in Figure 4.6, which will work for sentence end detection in accordance with the data created in the second stage of the system. Finally, the created network was tested with data that had not seen before and the results were obtained.

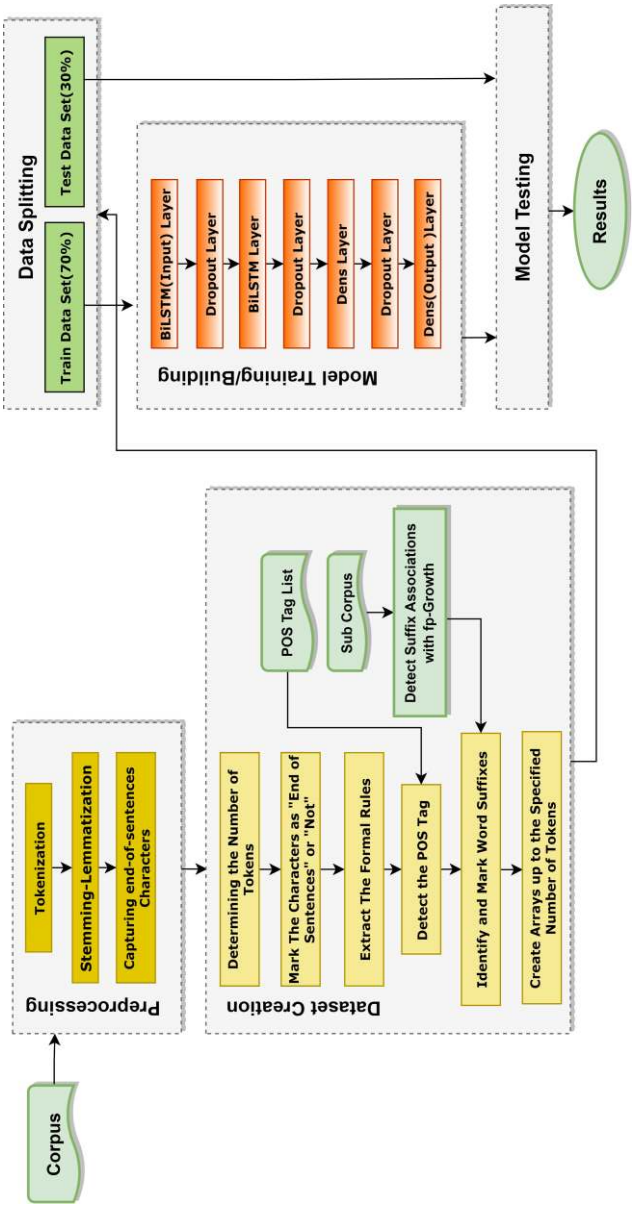


Figure 4.12. General System Description

# 5. RESULTS AND DISCUSSIONS

In this thesis we performed experiments to determine which classifers and features that should be used for our SBD system.

## 5.1. Comparison of Classical Classifiers on Rule-based Dataset

In the following subsections, first we showed the importance of the corpus used for developing a SBD system. For this purpose, a comparison of the TNC subcorpus and some well-known English corpus was made. In this comparison, the performances of rule-based datasets (used features) and common classifiers were evaluated.

### 5.1.1. Corpora Used in the Study

Two English corpora with the TNC sub-corpus were compared in this section of the study. In addition to the corpus evaluation, the methods were also compared in terms of Turkish and English languages.

In addition to the Genia and Lob corpora, the TNC subcorpus containing 10 million words was also used. In Table 5.1. Numerical information about the boundary of sentence characters of the three corpora is given.

Table 5.1. Sentence numbers of corpus

| | Total Number of Sentence Boundary Characters | Number of Characters pointing to actual sentences | Number of Characters not pointing to sentence boundaries |
|---|---|---|---|
| **Genia** | 21,079 | 18,541 | 2,538 |
| **Lob** | 64,263 | 54,043 | 10,220 |
| **TNC** | 917,974 | 754,370 | 163,604 |

The same number of samples was determined for a robust evaluation of the tests we performed on the datasets. For this purpose, after various experiments, three

separate datasets of 30,000 examples were created, of which 15,000 were examples of real sentence endings and 15,000 examples that did not show the ending of sentences. In the study, it was determined that working with a higher number of samples did not affect the success rate of the models tested, but created too much processing load. These samples were randomly selected from the datasets. In cases where there were not enough samples, the data was replicated with the random oversampling method. In case of oversampling, random oversampling method was used.

Before applying oversampling, 70% of the data instances from both classes in each of the original datasets were selected as the training sets and the remaining 30% of the instances were used as the testing sets. After data was reserved for 70% Training and 30% Testing, oversampling was applied separately for train and test datasets. In this way, the possibility of the duplicated data existing on both train and test sides is eliminated and the system's memorization is prevented. Sample numbers of the new balanced dataset are also shown in Table 5.2.

Table 5.2. Number of instances in the balanced corpus

| | **Total Number of Sentence Boundary Characters** | **Number of Characters pointing to actual sentences** | **Number of Characters not pointing to sentence boundaries** |
|---|---|---|---|
| **Genia** | 30,000 | 15,000 | 15,000 |
| **Lob** | 30,000 | 15,000 | 15,000 |
| **TNC** | 30,000 | 15,000 | 15,000 |

### 5.1.2. Determining the Features

In order to run a classifier over a dataset, there is a need to extracting some features from the raw data. First of all, it is necessary to determine the characters that determine the boundary of a sentence in the raw data. After this study was done on

three raw datasets, five characters showing the boundary of the sentence were obtained. These characters are ".", "!", "?", ":" and "…".

Then, for each end of sentence character, the below 9 features and a class label were produced according to the positions of the characters indicating the boundary of the sentence. These forms the feature set of our raw datasets. Thefeore the features used are listed as follows :

**[1]** Is the first letter of the previous word capitalized?

**[2]** Is the first letter of the next word capitalized?

**[3]** Is the previous word all uppercase?

**[4]** Is the next word all uppercase?

**[5]** Does the next word consist of numbers?

**[6]** Is the previous word shorter than 5 characters?

**[7]** Length of previous word?

**[8]** Length of next word?

**[9]** What is the mod value of the number of double quotes before it?

While creating these features, the feature selection criteria in Wong and Chao's study were used (Whong and Chao, 2010). In addition to this, feature 9 aims to capture the expressions in quotation marks. These features were obtained for each of our three basic datasets and tested in classification algorithms by taking 15,000 samples from each of the examples showing and not showing the boundary of the sentence. We called these datasets as rule-based datasets.

### 5.1.3. Comparison of Classifiers for the Rule-based Datasets

With the above-mentioned features, the specified number of samples from all three corpus were used. The three balanced datasets created were compared with the decision tree (C4.5), Back Propagation neural network, Support Vector Machine (using RBFKernel), RBF Network and Naive Bayes classification methods. The

specified algorithms have been tested with a data mining application, Weka (x64). Algorithms were run with the default parameter values suggested by Weka.

The F1-score(F-measure) value was used as the evaluation criterion. The results obtained are listed in Table 5.3. In the results seen in Table 5.3, although some algorithms that were used before gave good results in well-known datasets such as Genia Corpus and LOB Corpus, it was seen that they did not give good results in TNC Sub Corpus. Although the best result was obtained by the decision tree algorithm, the success rate in TNC Corpus showed us that it needed improvement.

Table 5.3. Performance results (F-score rate) of classical classifiers in rule-based dataset

| | **Genia Corpus AVG-F1 score** | **LOB Corpus AVG-F1 score** | **TNC Corpus AVG-F1 score** |
|---|---|---|---|
| **Back Propagation** | 0.96 | 0.96 | 0.84 |
| **SVM – RBFKernel** | 0.95 | 0.94 | 0.784 |
| **RBF Network** | 0.92 | 0.95 | 0.754 |
| **Naive Bayes** | 0.94 | 0.94 | 0.770 |
| **C4.5** | **0.98** | **0.98** | **0.85** |

### 5.1.4. LSTM with Rule-based Dataset

The rule-based datasets with 9 features obtained from various English corpuses and TNC were classiferd with various conventional classifiers in the above subsection. According to the results, although classical classifiers gave good results in English corpus containing mostly uniform texts, their f-measure values remained low in TNC with texts in various types and fields. Therefore, we need improvement for TNC corpus. The same TNC dataset was also tested with a deep learning method called LSTM. Test results of LSTM are shown in Table 5.4. In this study, the

obtained result with LSTM in the rule-based dataset was below the C4.5 and back propagation methods.

Table 5.4. Comparison of classical classifiers with LSTM

| TNC Corpus | |
|---|---|
| | **AVG-F1 score** |
| **Back Propagation** | 0.84 |
| **SVM – RBFKernel** | 0.78 |
| **RBF Network** | 0.75 |
| **Naive Bayes** | 0.770 |
| **C4.5** | 0.85 |
| **LSTM** | 0.83 |

### 5.1.5. Effect of POS Tags on Rule-based Dataset

A POS tag (Part of Speech Tag) is a tag assigned to each token (word) information of the text. This tag specifies grammatical information such as time, status, number (plural/singular) of the relevant token. POS tags are used in corpus searches, text analysis tools and algorithms. Many natural language tasks require the correct assignment of POS tags to texts encountered for the first time. By using manually tagged corpus, with the help of different methods and techniques, texts that have not been encountered before are tagged. At the beginning of these, there is the method in which the maximum entropy method is used and an f-measure of 96.6% is obtained (Ratnaparkhi, 1996). Various POS taggers have also been revealed by neural network (Schmid, 1994) (Gimpel et al., 2010), HMM (Hidden Markow Model) (Cutting et al., 1992), trigram (Kempe, 1993) methods. Zemberek NLP framework, on the other hand, has a POS tagger for Turkish texts.

In this part of the study, the grammatical type of the last word of the sentence and the first token (word) of the new sentence that will come after the end-of-sentence character has been examined (Figure 5.1). In the regular sentence sequence in Turkish, the type information of the first and last tokens of all kinds of sentences

is usually the same (Böler, 2019). For example, in sentences declaring action, the last token consists of words with verb POS tag information. Likewise, in a sentence that states a question, the sentence ends with the question word (Question Particle) POS tag information. Headings of sentences begin with “name” (noun) or “proper noun” POS tag information. Based on this, we proposed that in addition to the 9 features used in the previous experiment in this study, the POS tag information of these two tokens should have been added to the feature list. Since the POS tags of each token on the used corpus TNC are tagged manually, this POS tag information was used in our test.

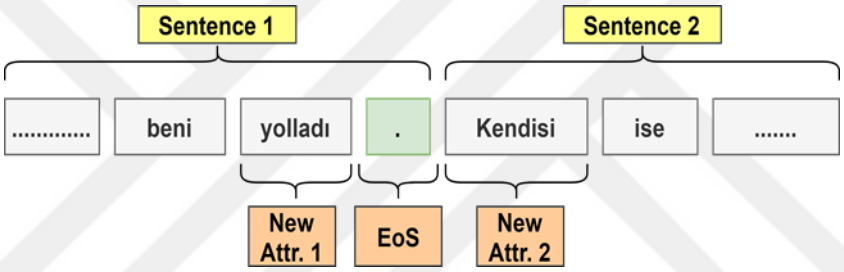


Figure 5.1. New Features in Sentence Example

When two new features were added, the total number of features became 11. The improvement obtained in classical classifiers with the addition of these two POS tag features were shown in Table 5.5. (Bektaş and Özel, 2018).

Table 5.5. The effect of new features on sentence boundary determination

| | F1 score | |
|---|---|---|
| **Classifiers** | **9 Features** | **11 Features** |
| **Back Propagation** | 0.84 | 0.86 |
| **SVM – RBFKernel** | 0.78 | 0.81 |
| **RBF Network** | 0.75 | 0.75 |
| **Naive Bayes** | 0.77 | 0.78 |
| **C4.5** | **0.85** | **0.86** |

### 5.1.6. Adding Appropriate Suffix Information to the Rule-based Dataset and Its Impact on Performance

In the previous section, Section 5.1.5., it was determined that the results improved between 0.5% and 2% with two new features added to the rule-based dataset. Based on these results, it was decided to examine the additional information of the token at the end of the sentence and to examine which suffixes are used with the boundary of sentence character. It is arranged according to the order of subject, object and verb (predicate) according to the order of the constructors of Turkish basic sentences (Tosun, 2005). The word to be emphasized comes before the predicate and may vary depending on the situation. On the other hand, in all regular sentences, the predicate is at the end and is effective in concluding the sentence. It was shown that the only element that does not change in regular sentences is the place of the predicate, which is the boundary of sentence element (Mete and Ceyhan, 2015). In this direction, the suffixes of the last token of the potential sentences, that is, before our potential sentence end character, were examined.

Various sources have stated that there are a total of 32 suffixes that can be added to a word in Turkish and shown in Table 5.6 (Aksan et al., 2016). Although the boundary of sentence token is a verb in regular sentences, it can also be from different word groups in interrogative sentences, inverted sentences and in some special cases. For this reason, additional information of all word types has been examined. First of all, a dataset of 50,000 random samples with 32 additional features and class label with binary values was prepared in order to determine the additional associations formed by the boundary of sentence characters. Here, the class label contains the information whether the relevant example is the end of the sentence or not.

Table 5.6. List of suffixes and examples used in Turkish

| Suffix definition | Suffix definition in Turkish | Example |
|---|---|---|
| **reduplication** | ikileme | gide |
| **voice** | çatı | uyuttu, yaptırdı |
| **Verb** | eylem | gittiyse |
| **referential/perfective** | bitmişlik | gitmiş |
| **past / perfective** | geçmiş zaman | gitti |
| **necessity** | gereklilik | gitmeli |
| **imperfective** | bitmemişlik | gidiyor |
| **continuous** | şimdiki zaman | gitmektedir |
| **future** | gelecek zaman | gidecek, gidecek lerden |
| **aorist** | geniş zaman | acımayız, uyursun, uyumaz |
| **pronominal** | adıllaştırıcı | benimki |
| **person** | kişi uyum | geliyor |
| **nominalizer** | adlaştırıcı | uyumak |
| **negative** | olumsuz | gitmedik |
| **imperative** | buyruk | gelsinler, gideler |
| **copula** | koşaç | gitmektedir |
| **clitics** | parçacık | verebilir, alabilir |
| **buffer phoneme** | arases | uyuyacak |
| **auxiliary verb** | yardımcı eylem | gelemez, gelebilir |
| **adverbial** | belirteçleştirici | gideli |
| **adjectival** | sıfatlaştırıcı | diyesilermiş, diyesiler, diyesi |
| **number/person** | uyum_sayı | okullar |
| **buffer phoneme** | arases | gecesi gündüzü |
| **case-nominative** | durum_yalın | masa |
| **case-accusative** | durum_belirtme | defteri |
| **case-genitive** | durum_tamlayan | defterin rengi |
| **case-dative** | durum_yönelme | deftere |
| **case-locative** | durum_kalma/bulunma | defterde |
| **case-ablative** | durum_çıkma | defterden |
| **case-instrumental** | durum_araç | defterle |
| **person_copula** | koşaç | nöbetçiler (onlar) |
| **possessive** | iyelik | onların saçları |

FP-Growth algorithm was run on the created dataset and frequency lists were obtained. From this frequency list, the association rules that emerged with the class label were created. The rule list with the highest confidence value is given in Table 5.7.

Table 5.7. List of rules that appear with the class feature

| Association | Results | Confidence | Lift |
|---|---|---|---|
| **voice, copula, person** | Class | 0.860 | 1.720 |
| **voice, copula person** | continuous, Class | 0.860 | 1.734 |
| **continuous, voice, copula, person** | Class | 0.860 | 1.720 |
| **continuous, copula, person** | Class | 0.856 | 1.711 |
| **copula, person** | Class | 0.856 | 1.711 |
| **copula, person** | continuous, Class | 0.856 | 1.725 |
| **continuous, copula, person** | voice, Class | 0.851 | 1.758 |
| **copula, person** | voice, Class | 0.851 | 1.758 |
| **copula, person** | continuous, voice, Class | 0.851 | 1.758 |

From the frequency lists, it has been determined that the class feature is frequently used together with suffix information such as "voice, copula, person and continuous ". Accordingly, new binary features containing 4 suffix information have been added to the expanded rule-based dataset that has 15 features used in Section 5.1.5. Afterwards, the same balanced dataset with 30,000 samples was reconstructed with new features. The same tests were repeated with the classifiers used in the previous sections. The results of this test are presented in Table 5.8.

Table 5.8. The effect of new features containing suffix information on sentence boundary determination

| | F1 score | | |
|---|---|---|---|
| **Classifiers** | **9 Features** | **11 Features** | **15 Fetaures** |
| **Back Propagation** | 0.84 | 0.86 | 0.92 |
| **SVM – RBFKernel** | 0.78 | 0.81 | 0.86 |
| **RBF Network** | 0.75 | 0.75 | 0.82 |
| **Naive Bayes** | 0.77 | 0.78 | 0.87 |
| **C4.5** | **0.85** | **0.86** | **0.93** |

When the results in the table are examined, it is seen that the highest F1-score is obtained in the decision tree algorithm, as in the previous tests. However, it

has been determined that f-measure rate has increased significantly in all the methods tested and that suffix information in Turkish has a positive effect on determining the boundary of sentences.

After the tests on balanced datasets, the method was compared with other sentence detection systems. Since other systems determine sentences on raw text data, it is not possible to test these systems on a balanced data. Therefore, a dataset consisting of 95 documents containing 50,493 potential sentences randomly drawn from the TNC corpus was prepared. Systems were trained with 70% of the dataset and tests were performed with the remaining 30%. In order for the comparison between the systems to be accurate, comparisons were made through Precision, Recall and F1 score measurement methods. The purpose of using these metrics mentioned above is due to the fact that the situation in which the characters that are likely to be the end of the sentence point to the end of the sentence is very dominant compared to the other case. In other words, end of sentence case is much more frequently observed than the case of no end of sentence, and the dataset falls into an unbalanced state.

The test results of the 4 existing methods mentioned in the previous parts of the thesis and the method we recommend are shown in Table 5.9.

Table 5.9. Comparison of the proposed method with existing systems

| Method | Number of Documents | Number of Total Sentence | Precision | Recall | F1 |
|---|---|---|---|---|---|
| **JSBD** | 95 | 50493 | 0.65 | 0.61 | 0.63 |
| **Splitta** | 95 | 50493 | 0.64 | 0.59 | 0.62 |
| **Geniass** | 95 | 50493 | 0.76 | 0.79 | 0.77 |
| **9 Eylül NLP** | 95 | 50493 | 0.75 | 0.72 | 0.73 |
| **Our Method** | 95 | 50493 | **0.87** | **0.93** | **0.89** |

As can be seen from the results in Table 5.9, our proposed method shows a significant improvement on Turkish documents compared to other existing systems.

## 5.2. Using CNN with Word Vectors

Word vectors, called Word2Vec, are a two-layer neural network used for processing texts. Its input is a text collection and its output is a vector set. Word2Vec is not a deep neural network. It also converts texts into digital form that deep neural networks can process. The vectors we use to represent words are called Neural Word Embeddings.

Within the scope of the study, 50, 100 and 200 dimensional word vectors were created for all words. After the word vectors are created, the process is to express each candidate sentence as a sequence with a 2-dimensional matrix. The matrix of a sentence is shown in Figure 5.2.

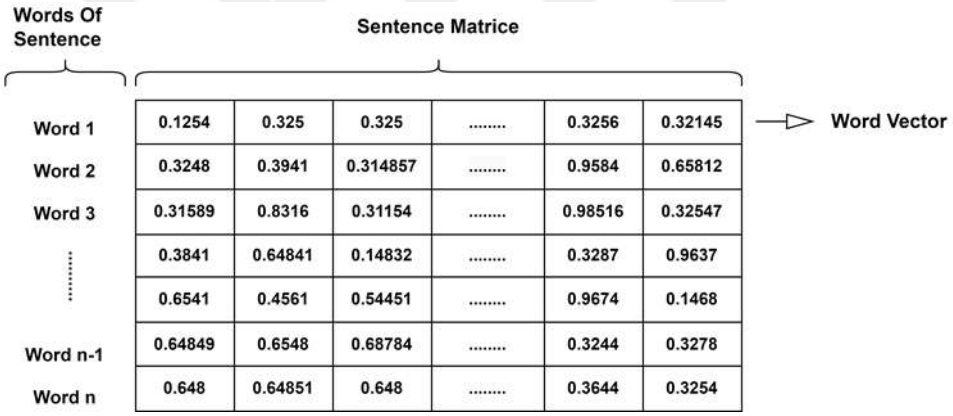

Figure 5.2. Example matrix created with Word2Vec for a sentence candidate

In this study, the root information of the words and the type tag (POS) information of the words were created with separate word vectors and tests were carried out on the same networks. In the tests, the sequences of all the tokens of the sentence candidates and the last 3 token sequences were used and an example of the two method is shown in Figure 5.3.

| Words Of Sentence | Sentence Matrice | | | | | |
|---|---|---|---|---|---|---|
| sermaye | 0,32547 | 0,32545 | 0,12547 | ..... | 0,3256 | 0,32145 |
| kayıp | 0,3248 | 0,3692 | 0,31545 | ..... | 0,9584 | 0,698152 |
| çeşitli | 0,98742 | 0,325894 | 0,31154 | ..... | 0,684541 | 0,6814 |
| tanım | 0,841654 | 0,64851 | 0,68784 | ..... | 0,3244 | 0,9788415 |
| yapmak | 0,64841 | 0,64851 | 0,648 | ..... | 0,681 | 0,9815 |

(a)

| POS Tag Of Sentence | Sentence Matrice | | | | | |
|---|---|---|---|---|---|---|
| noun | 0,39484 | 0,3215 | 0,12547 | ..... | 0,3256 | 0,32145 |
| noun | 0,39484 | 0,3215 | 0,12547 | ..... | 0,3256 | 0,32145 |
| adjective | 0,98415 | 0,2135 | 0,68741 | ..... | 0,841 | 0,987491 |
| noun | 0,39484 | 0,3215 | 0,12547 | ..... | 0,3256 | 0,32145 |
| verb | 0,6451 | 0,64851 | 0,648 | ..... | 0,65451 | 0,9815 |

(b)

Figure 5.3. Matrices formed for the example sentence "Sermaye kaybının çeşitli tanımları yapılmıştır." (a) Matrix formed by word vectors of the words in the sentence. (b) Matrix formed by word vectors of the POS tags of words in the sentence.

### 5.2.1. CNN Networks for Word Vectors

Two CNN networks were applied on the matrices obtained from the word sequences or POS tag sequences of the sentence, and their performances were compared. Word vectors were prepared separately by word2vec as 50, 100 and 200 units. However, since the sentence lengths are not fixed, the matrix sizes are equalized by resizing while sending all sentence sequences to the network. When only the last 3 tokens are looked at, there is no need to equalize the sizes because the matrices have 50x3, 100x3 and 200x3 dimensions. The first CNN network used is seen in Figure 5.4 and is named CNN_a. CNN_a contains fewer layers as network depth than other network used namely CNN_b which is shown in Figure 5.5. CNN_b

network has more depth and it is aimed to extract more features by using 3 convolution layers. In addition, it is aimed to increase training performance by adding a dropout layer.

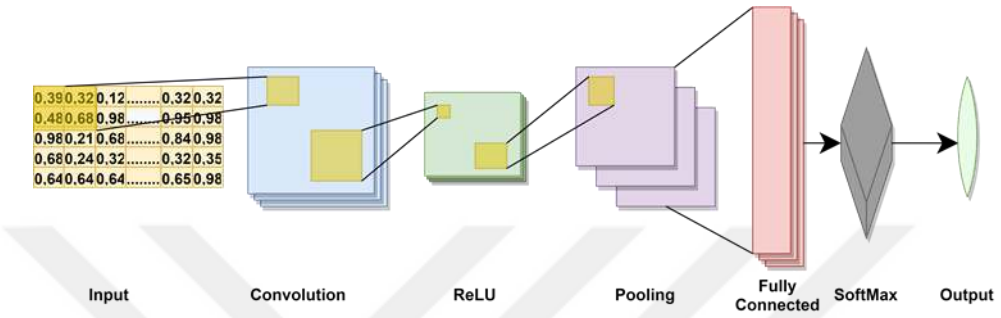


Figure 5.4. Network layers of CNN_a

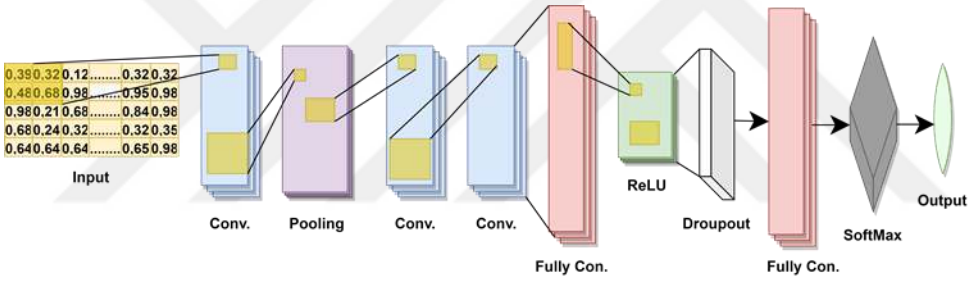


Figure 5.5. Network layers of CNN_b

### 5.2.2. Implementation

Tests were conducted using datasets consisting of root sequences of words and POS tag sequences as defined in Section 5.2. Two CNN designs, named CNN_a and CNN_b, were used in these tests. Word2vec was used to form input matrices which contain either all the sentence tokens or the last 3 token values. The tests were performed with 15,000 positive and 15,000 negative samples mentioned in section 5.1. The results were shown in Table 5.10. Best results for each row are written in bold.

Table 5.10. Performance (F1-score) of sentence matrices in CNN_a and CNN_b networks

| Word Vector Length | Network | Roots of Words | | PosTag Information | |
|---|---|---|---|---|---|
| | | Whole Sentence | Last 3 Tokens | Whole Sentence | Last 3 Tokens |
| **50 units** | CNN_a | 0.59 | 0.60 | 0.60 | **0.62** |
| | CNN_b | **0.65** | 0.50 | 0.63 | 0.64 |
| **100 units** | CNN_a | 0.59 | 0.58 | 0.60 | **0.61** |
| | CNN_b | **0.65** | 0.50 | 0.62 | 0.64 |
| **200 units** | CNN_a | 0.57 | 0.61 | 0.58 | **0.62** |
| | CNN_b | 0.63 | 0.54 | 0.62 | **0.64** |

According to the evaluation, the network named as CNN_b gave better results compared to the other design. Although the choice of word vector lengths with different values such as 50, 100 and 200 did not make a difference in terms of performance, there were huge differences in processing times. For this reason, it has been observed that word vectors with a length of 50 units are more suitable for use. However, although there is not much difference in the use of root or type tag information as input, it has been observed that root information has a higher accuracy rate. When we look at the values of the whole sentence and the last 3 units, the using the roots of the words in the whole sentence and POS tags of the last three words of the sentence as input produced more successful results. On the other hand, it has been observed that the classical classifiers used in Section 5.1. have higher success in rule-based datasets. The possible reason for this is that it creates a word proximity map and creates vectors by using the semantic relations of words such as Word2vec. It has been observed that these Word2vec-style methods are not a suitable approach in detecting the end of sentences, since the word sequences in structures that indicate or do not indicate the end of the sentence do not make a semantic difference.

### 5.2.3. Execution of BERT

The dataset consisting of root sequences of words defined in Section 5.2.1 and POS tag sequences has also been tested with the BERT algorithm. The BERT method, which has been frequently used recently for classifications made on the

semantic relations of words, was used for the same dataset and compared with the Word2Vec method. Since the outward optimization adjustments of the BERT algorithm are limited and there is no possibility to interfere with the method, the comparison was made only on the whole sentences.

Table 5.11. Results of CNN_a, CNN_b and BERT methods

| Network | F-Score for TNC Corpus |
|---|---|
| **CNN_a** | 0.59 |
| **CNN_b** | 0.65 |
| **BERT** | **0.77** |

When the results given in Table 5.11 are examined, it has been observed that the BERT method is more successful for boundary of sentence detection than CNN networks operated using Word2Vec. Despite this, the f-measure of 77% showed that the BERT method did not offer the desired level of improvement compared to the classical classifers with the rule based dataset.

### 5.3. Execution of RNN

In the tests, the results of which were shared in the previous sections, some features consisting of the formal properties of the tokens were obtained. Later, POS tag information of various numbers of tokens before and after the possible end-of-sentence character was added to the dataset, and the success achieved was brought to a better level. Finally, additional information about the possible last tokens of the sentences was added to the dataset. It was observed that enlarging the feature set with features obtained from the last tokens of the sentences significantly increased the success of all conventional classification algorithms. A sequential dataset was created by combining the results obtained from all these tests and the feature of a sentence having a sequential structure, and it was tested with LSTM, an application

of RNN. Since the sentence structures have a certain sequence in both directions, the obtained dataset was also checked with Bidirectional LSTM.

### 5.3.1. Creating the Dataset

In this thesis, we proposed to create a sequential dataset based on the importance of the sequential structure of each word in a sentence. For this, a vector of each processed token was created. What the vector elements are and the value ranges they take are given in Table 5.12. The first element of the token vector contains the POS tag information of the relevant token, the 2-6 range elements contain the formal information about the token, and the 7th element determines whether it is a header line or not. The features from 8 to 40 contain whether the additional information specified as binary is available in the relevant token. The last vector element (element 41) contains the class label.

Table 5.12. Vector elements for each token

| Vector Element | Detail of Element | Value Range |
|---|---|---|
| **1** | POS Tag | 1-23 |
| **2** | Punctuation Code | 1-17 |
| **3** | Is the First Letter Capitalized? | 0/1 |
| **4** | All Uppercase? | 0/1 |
| **5** | Does it consist of numbers? | 0/1 |
| **6** | Token Length | 0-200 |
| **7** | Next Enter? | 0/1 |
| **8** | Suffix(Bare) | 0/1 |
| **9** | Suffix(Possessive) | 0/1 |
| **10** | Suffix(person_copula) | 0/1 |
| **11** | Suffix(case-instrumental) | 0/1 |
| **12** | Suffix(redublication) | 0/1 |
| **13** | Suffix(referential/perfective) | 0/1 |
| **14** | Suffix(past / perfective) | 0/1 |
| **15** | Suffix(necessity) | 0/1 |
| **16** | Suffix(imperfective) | 0/1 |
| **17** | Suffix(future) | 0/1 |
| **18** | Suffix(aorist) | 0/1 |
| **19** | Suffix(pronominal) | 0/1 |
| **20** | Suffix(person) | 0/1 |

| | | |
|---|---|---|
| **21** | Suffix(nominalizer) | 0/1 |
| **22** | Suffix(negative) | 0/1 |
| **23** | Suffix(imperative) | 0/1 |
| **24** | Suffix(copula) | 0/1 |
| **25** | Suffix(clitics) | 0/1 |
| **26** | Suffix(buffer phoneme) | 0/1 |
| **27** | Suffix(auxiliary verb) | 0/1 |
| **28** | Suffix(adverbial) | 0/1 |
| **29** | Suffix(adjectival) | 0/1 |
| **30** | Suffix(number/person) | 0/1 |
| **31** | Suffix(case-nominative) | 0/1 |
| **32** | Suffix(case-accusative) | 0/1 |
| **33** | Suffix(case-genitive) | 0/1 |
| **34** | Suffix(case-dative) | 0/1 |
| **35** | Suffix(case-locative) | 0/1 |
| **36** | Suffix(case-ablative) | 0/1 |
| **37** | Suffix(case-instumental) | 0/1 |
| **38** | Suffix(Verb) | 0/1 |
| **39** | Suffix(Voice) | 0/1 |
| **40** | Suffix(reduplication) | 0/1 |
| **41** | Sentence End or Not | 0/1 |

Each generated token vector is given to LSTM and BiLSTM networks in certain window size (Figure 5.6). As explained in the Method section, sequential data is used in all RNN networks. For this reason, in our LSTM applications, all data were scanned and a sequential sample set was obtained according to the determined window size as shown in Figure 5.7. If a dataset was created in this way, a large number of examples without end-of-sentence characters were obtained. It has been observed that the extremely high number of data, which makes it difficult to operate on the dataset, also makes the negative sample weight very dominant. As can be seen in the sample data piece with the window size value of 5 in Figure 5.7, too many examples are formed in order to capture a single sentence end character.

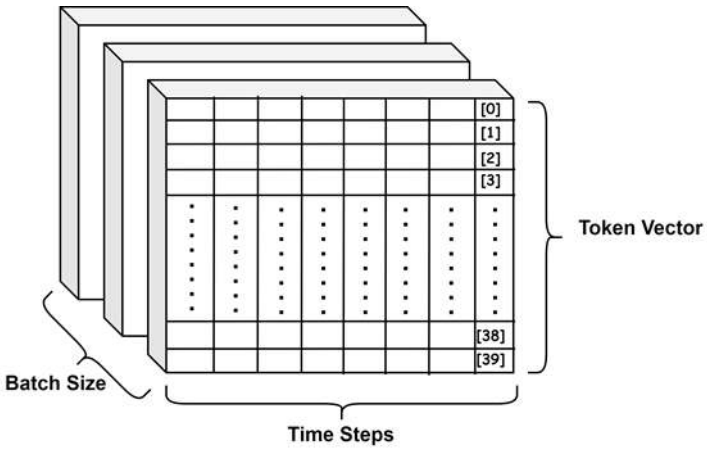


Figure 5.6. LSTM Input Format

| Sample No | Token 1 | Token 2 | Token 3 | Token 4 | Token 5 | Class |
|---|---|---|---|---|---|---|
| 1 | Yenilik | üretimde | verimli | kombinasyonların | ortaya | False |
| 2 | üretimde | verimli | kombinasyonların | ortaya | çıkmasıdır | False |
| 3 | verimli | kombinasyonların | ortaya | çıkmasıdır | . | True |
| 4 | kombinasyonların | ortaya | çıkmasıdır | . | Bu | False |
| 5 | ortaya | çıkmasıdır | . | Bu | kavram | False |
| 6 | çıkmasıdır | . | Bu | kavram | yalnızca | False |
| ..... | ..... | ..... | ..... | ..... | ..... | ..... |

Figure 5.7. An example part of dataset (window size=5)

There are many more examples of long sentence structures that do not contain any end-of-sentence characters. In addition, keeping the window size value low increases this sample number even higher. This approach is widely adopted in speech recognition studies. However, in this study, the ambiguities of the end-of-sentence characters in written texts are tried to be eliminated. For this reason, it is a more correct approach to focus on examples with potential end-of-sentence characters rather than all examples. Another problem is whether this example will be labeled as “True” if the end-of-sentence character is in the nature of it. Two approaches are possible here. In the first approach, if there is an end-of-sentence character anywhere in the sequential array, this sequential example is added to the

dataset. This sample is labeled as "True" if it actually marks the end of the sentence, otherwise as "False" class label. Another approach is the potential end-of-sentence character is chosen as the middle element in the sequential sample. In this case, the last tokens of the probable sentence and the first tokens of the following sentence are added to the sequential sample with equal weight. Then, it is ensured that the sample is labeled as “True” or “False” depending on whether it is really the end of the sentence or not. In our study, in order to focus on potential sentence endings, it was decided that the end-of-sentence characters should be the feature in the middle of the sequential sample. An example illustration of the approach outlined in Figure 5.8 is given.

| Sample No | Pre. Token 2 | Pre. Token 1 | Potential EOS | Next Token 1 | Next Token 2 | Class |
|---|---|---|---|---|---|---|
| 1 | özellikle | kullanılmıştır | . | Bunun | yerine | True |
| 2 | Joseph | A | . | Schum-peter | farklı | False |
| 3 | güvenen | varmış | ! | Kemik'e | , | True |
| 4 | ceharıdü | oynayacak | ... | Bekle | , | True |
| 5 | ortaya | çıkmasıdır | . | Bu | kavram | True |
| 6 | , | beşinci | ... | onuncu | bakış | False |
| ..... | ..... | ..... | ..... | ..... | ..... | ..... |

Figure 5.8. An example from the dataset (Window size=5)

In this usage, if the window size value is 5, the 3rd token always carries a possible end-of-sentence character. A total of 4 tokens before and after it form the remaining features. The same situation occurs when window size is 7 in which the 4th token is the end-of-sentence character.

It is aimed to obtain a better result in comparison with other systems in performing our RNN tests. For this reason, instead of a balanced dataset, a dataset consisting of 95 documents containing 50493 potential sentences randomly drawn from the TNC corpus was prepared. This dataset, which we also used in our previous experiments, made it possible to compare with our previous experiments and other systems. As in all our experiments, 70% of the data set was used for training and the

remaining 30% was used for testing. The processes from document pulling to receiving classification results over networks are shown in Figure 5.9.

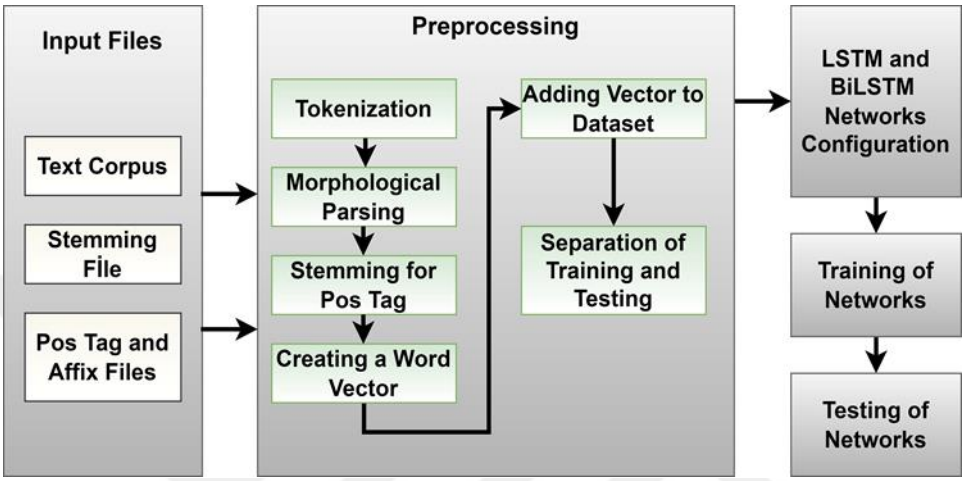


Figure 5.9. Flow Chart of the Proposed Method

### 5.3.2. Execution of LSTM

The general illustration of the LSTM network prepared for the experimentation of the dataset prepared in different window sizes (3, 5, 7, and 9) is shown in Figure 5.10. Here, sentence end determination requires a binary classification. For this reason, the network to be created is designed with a many-to-one type RNN architecture.

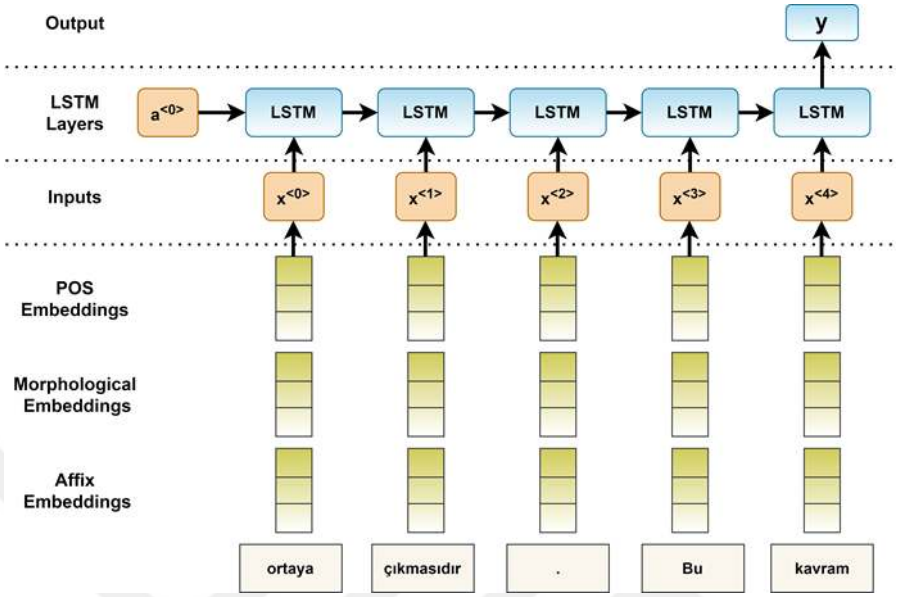


Figure 5.10. Many-to-One LSTM Architecture (window size=5)

The standard LSTM network is used in the created model. The input size in the input layer can be [3, 40], [5,40], [7,40] and [9,40] depending on the window sizes used in the tests. 128 units are used in LSTM, which is used as a hidden layer. The size in the input layer consists of an average of 240 units, depending on the incoming data. Since the last output layer has 2 units, the value of 128, which is between these units and corresponds to approximately two-thirds of the input, was used as the number of units in the hidden layer in LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000). Such a large amount of incoming data ReLU (Rectified Linear Unit) was used as the activation function to not extend the process time. Neurons that produce negative values in ReLU take the value zero. This allows ReLU to work more efficiently and faster than the Hyperbolic Tangent and Sigmoid function. Therefore, ReLU is more preferred activation function in multilayer neural networks. Then dropout is used to ensure that the nodes in the hidden layers are less affected by each other's weight changes, but still improve network performance. Using dropout, the bonds in fully-connected layers are broken. Thus, nodes have less information about each other, and as a natural consequence, nodes are less affected

by each other's weight changes. The output of these two layers enters again in an LSTM (hidden layer) and a Dropout layer with the same parametric values. Then, with a 32-neuron Dense layer, the transitions of neurons or nodes are provided for the next output layer. In other words, it allows neurons from one layer to be connected to the next output layer as input. Finally, a Dropout layer is followed by an output layer with 2 neurons. Sigmoid or hyperbolic tangent functions are used as activation functions for this layer. This LSTM network diagram created specifically for the study is shown in Figure 5.11.

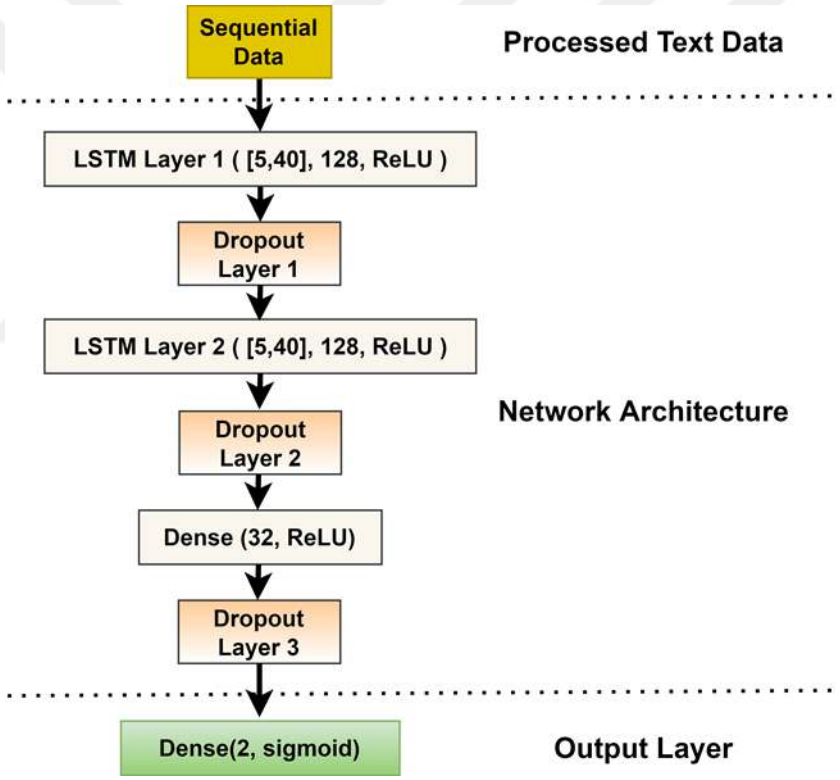


Figure 5.11. LSTM Network Architecture (window size=5)

The dimensions of the input layer of the prepared LSTM network vary according to the window size value. The token count before and after the end-of-sentence character was checked in the tests performed in section 5.1.5. According to

the results obtained in these experiments, it was determined that while the effects of tokens close to the end of the sentence character were positive on the dataset, the effect on f-measure decreased as they moved away. Therefore, in the experiments presented in this section, the window size value was tested as 3, 5, 7 and 9 by taking 1, 2, 3 and 4 tokens before and after the end of the sentence character. Tests were carried out with a sub-corpus of 95 documents having 50,493 sentences obtained from the TNC corpus, which was used in the tests in Section 5.1.6. This corpus was used to make a fair comparison with other systems. The results obtained are presented in Table 5.13. The same application was also tested with a dataset of 50000 samples obtained from the corpus of SETimes, which includes a Turkish corpus, and the results are given in Table 5.14. In these experiments, 70% of the data were used for training and the remaining 30% were used for testing purposes. According to the experimental results, it was seen that increasing the window size did not make a big difference on the result. Despite this, it was observed that the window size values of 5 and 7 were successful in both corpuses.

Table 5.13. LSTM Results with TNC sub-corpus

| Window Size | Accuracy | F1 Score |
|---|---|---|
| **3** | 0.968 | 0.931 |
| **5** | **0.972** | **0.941** |
| **7** | 0.972 | 0.940 |
| **9** | 0.971 | 0.939 |

Table 5. 14. LSTM Results with SETimes sub-corpus

| Window Size | Accuracy | F1 Score |
|---|---|---|
| **3** | 0.958 | 0.922 |
| **5** | 0.963 | 0.933 |
| **7** | **0.968** | **0.937** |
| **9** | 0.963 | 0.930 |

### 5.3.3. Execution of BiLSTM

It is predicted that the BiLSTM network can be applied to the related problem based on the two-way meaningful relationships of the sentence sequences. For this reason, the same datasets were tested on the network with BiLSTM layers. BiLSTM is based on the principle of operating LSTM layers in both directions. This structure is shown in Figure 5.12.

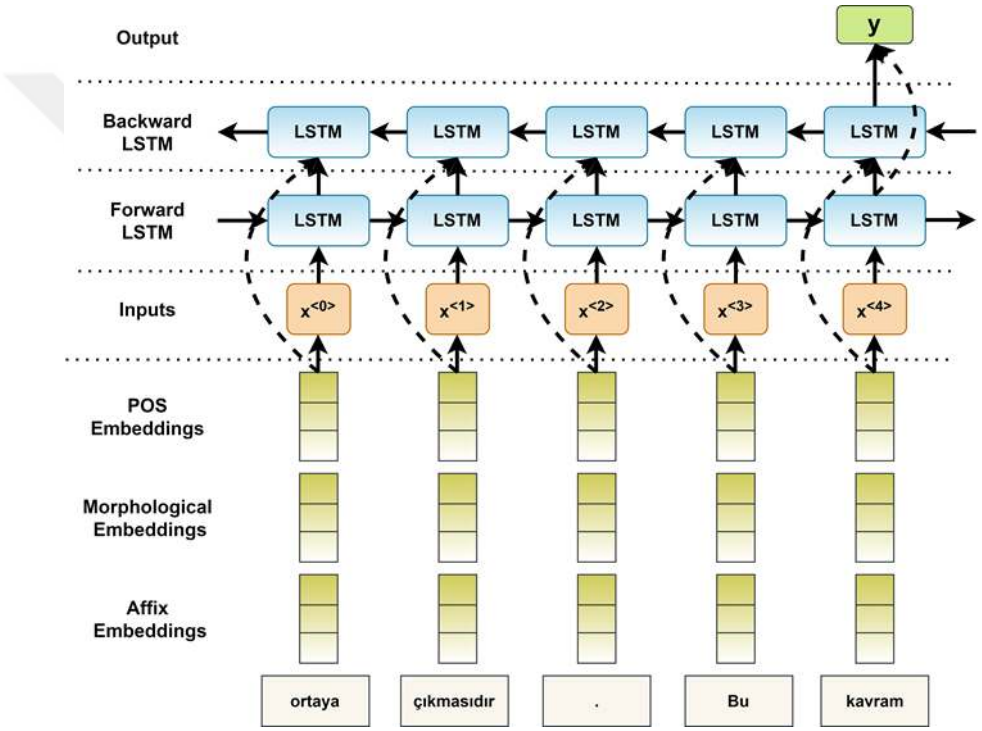


Figure 5.12. Many-to-One BiLSTM Architecture (window size=5)

The structure of the BiLSTM network is the same as the LSTM used in the previous section. LSTM layers are arranged to work bidirectionally and the remaining layers are left with all their parameters. The network prepared accordingly is shown in Figure 5.13.

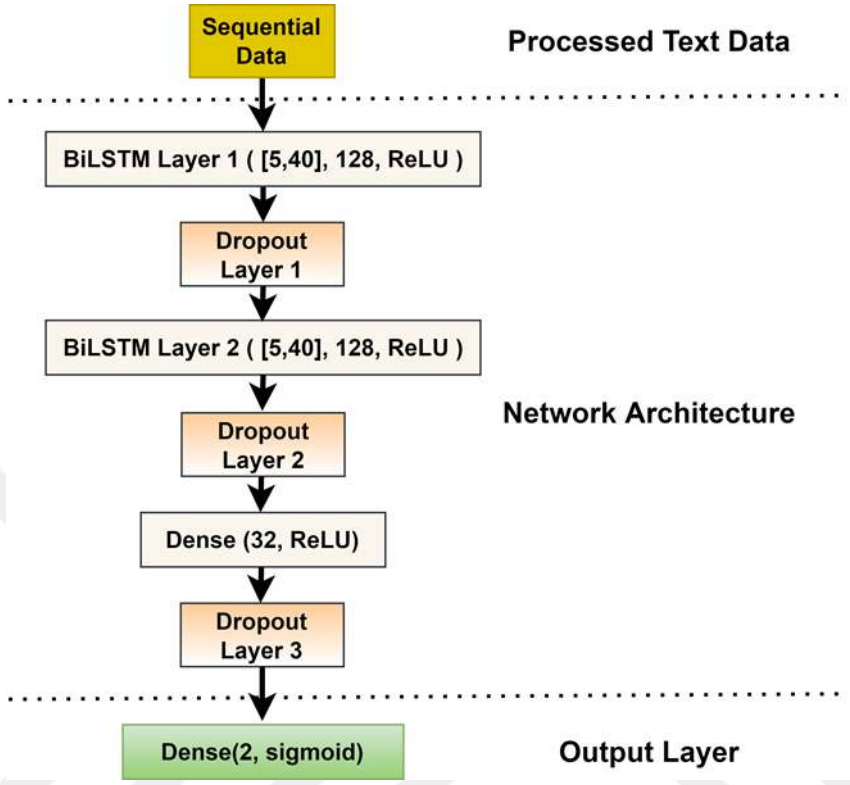


Figure 5.13. BiLSTM Network Architecture (window size=5)

The same dataset used in the LSTM tests was also used in the BiLSTM tests. There was no big difference in f-meaure values at different window sizes. However, little improvement was achieved compared to the success achieved with LSTM. Comparative results with the TNC sub-corpus are given in Table 5.15. Comparative results with the SETimes dataset are shown in Table 5.16.

Table 5.15. Evaluation Results of LSTM-BiLSTM with TNC sub-corpus

| | LSTM | | BiLSTM | |
|---|---|---|---|---|
| **Window Size** | **Accuracy** | **F1 Score** | **Accuracy** | **F1 Score** |
| **3** | 0.968 | 0.931 | 0.975 | 0.944 |
| **5** | **0.972** | **0.941** | 0.983 | 0.956 |
| **7** | 0.972 | 0.940 | **0.987** | **0.959** |
| **9** | 0.971 | 0.939 | 0.98 | 0.954 |

Table 5.16. Evaluation Results of LSTM-BiLSTM with SETimes Corpus

| | LSTM | | BiLSTM | |
|---|---|---|---|---|
| **Window Size** | **Accuracy** | **F1 Score** | **Accuracy** | **F1 Score** |
| **3** | 0.958 | 0.922 | 0.967 | 0.944 |
| **5** | 0.963 | 0.933 | **0.976** | 0.942 |
| **7** | **0.968** | **0.937** | 0.976 | **0.944** |
| **9** | 0.963 | 0.930 | 0.974 | 0.943 |

### 5.4. The Comparison of the Proposal with the Related Studies

In the literature, there are many methods and applications for boundary of sentence detection in written texts. On the other hand, there is no study that is directly compatible with the formal structure of the Turkish language. Based on this, in this section, the highlights of the existing studies and the results of the proposed method are compared. Many of the existing systems are based on the English language. For this reason, it has been shown in the previous sections that the desired success could not be achieved in corpuses where Turkish has various uses. Some studies are rule-based systems, while others cover various uses of machine learning methods. In our proposed method, for the first time, a hybrid system in which suffix associations are obtained with FP-Growth and classification is made with the recurrent deep learning method with the BiLSTM structure, has been proposed and tested. Comparison of results with other systems has also been presented in Table 5.17. The relevant comparison was made with 50493 sample sentence candidates obtained from 95 randomly selected documents from the TNC sub-corpus used in the previous

experiments. When the results obtained are examined, it is seen that the proposed method has achieved quite successful results compared to the general methods applied to Turkish.

Table 5.17. Comparison of the proposed method with existing methods on the TNC collection

| Method | Number of Documents | Total number of Sentence Candidate | F1-Score |
|---|---|---|---|
| **JSBD** | 95 | 50493 | 0.63 |
| **Splitta** | 95 | 50493 | 0.62 |
| **Geniass** | 95 | 50493 | 0.77 |
| **9 Eylül NLP** | 95 | 50493 | 0.73 |
| **Our Method** | 95 | 50493 | **0.96** |

Another corpus that was compared in the study was the SETimes corpus. Here, since the SETimes corpus is given on a sentence basis, an experiment was conducted with a random dataset of 50,000 samples. In this created dataset, 70% was used for training and the remaining 30% for testing. The comparison of the results with the TNC corpus is given in Table 5.18. According to Table 5.16, the results obtained from the studies of Schweter and Ahmed (2019) seem to be better than the method we suggested in the study. In the tests we performed with the parameters detailed in the mentioned study, it was seen that the mentioned results could not be obtained. In the experiments conducted by creating the LSTM network to which they send the character sequences, we obtained results that are lower than the results reported by Schweter and Ahmed (2019). When tested with the TNC corpus, much lower accuracy was obtained by their method. For this reason, we can say that the hybrid classification method based on BiLSTM and suffix associations we propose has more successful results.

Table 5.18. Comparison of the results of the proposed method in SETimes and TNC Corpora

| Corpus | F1 Score |
|---|---|
| **SETimes** | 0.944 |
| **TNC** | **0.959** |

# 6. CONCLUSION

It has been determined that the boundary of sentence detection systems and methods are predominantly made for English or other languages that are widely used in the digital world. For this reason, the need for a sentence-end determination method suitable for the language structure of Turkish has been the starting point of this study. Other than that, most publications have been tested with corpora of seamless text that is uniform and written in accordance with grammatical rules, such as newspaper and magazine articles. When these general purpose methods are used, the TNC corpus, which contains various usage forms of Turkish, has received very unsuccessful results. A hybrid system in which deep learning methods can be used with the formal features of Turkish has been proposed for the first time within the scope of this thesis.

First of all, rule-based datasets used in different studies were tested with various traditional classification methods and the desired success levels could not be achieved. After that, the POS tags of a certain number of tokens before and after the end-of-sentence characters were included in the datasets and the success value was improved by about 2% for all classifiers. The associations of the end-of-sentence characters with the suffixes in the tokens before and after them were determined by the FP-Growth method and added to the dataset. With the newly added features, the success value in traditional classifiers has been improved by about 4%-5% and reached to 92%. In the meantime, various new neural network based methods such as Word2Vec, BERT and deep learning methods such as CNN were tested, but it was determined that the semantic relationship among terms did not have an effect on the boundary of sentence detection.

In the later stages of the thesis, a vectorial and sequential dataset was created from the formal properties, POS tags and suffix information of the tokens. This dataset is classified with recurrent networks created in LSTM and BiLSTM architecture, respectively. Satisfactory classification results were obtained for both

networks in the SBD process. Especially when the sentence order has taken as a double-sided sequence, a success of 96% has been achieved in the BiLSTM network. In the literature, it was reported that other methods achieved higher success due to the datasets they used. On the other hand, it has been seen in the experiments that they are far from the mentioned achievements in the TNC corpus. The proposed method has achieved very successful results on the TNC corpus.

It is thought that the methods and experiments proposed in this thesis will make a great contribution to the sentence boundary determination studies for Turkish texts to be made in the future. In addition, it is possible to increase the success value by improving the design of the proposed network architecture. The most important output of this study can be stated as it has been shown that the association of suffix structures with the end-of-sentence characters eliminates the usage ambiguities of these characters. With this study, Turkish NLP studies were supported and contributed to increasing the prevalence of Turkish in the digital world.

## REFERENCES


Abdel-Hamid, O., Deng, L., and Yu, D., 2013. Exploring convolutional neural network structures and optimization techniques for speech recognition. In Interspeech, 1173-5.

Agarwal, N., Ford, K. H., and Shneider, M., 2005. Sentence boundary detection using a maxEnt classifier. In Proceedings of MISC, 1-6.

Agrawal, R., 1999. Fast algorithms for mining association rules in large databases. In Proc. 20th VLDB Conf., 487-499.

Agrawal, R., Imieliński, T., and Swami, A., 1993. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data, 207-216.

Akın, A. A., and Akın, M. D., 2007. Zemberek, an open source NLP framework for Turkic languages. Structure, 10(2007), 1-5.

Aksan, Y., Aksan, M., Koltuksuz, A., Sezer, T., Mersinli, Ü., Demirhan, U. U., ... and Kurtoğlu, Ö., 2012. Construction of the Turkish national corpus (TNC). In Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), 3223-3227.

Aksan, Y., Aksan, M., Mersinli, Ü., and Demirhan, U. U., 2016. A frequency dictionary of Turkish: Core vocabulary for learners. Routledge.

Aksan, Y., Özel, S. A., Bektaş, Y., Aksan, M., Demirhan, U. U., Mersinli, Ü., and Yılmazer, H., 2014. Türkçe Tümcelerin Sonunu Belirlemede Açık Kaynak/Ücretsiz Yazılımlar ve Performans Analizleri. Akademik Bilişim, Mersin 727-734.

Aktaş, Ö., and Cebi, Y., 2013. Rule-based sentence detection method (rbsdm) for turkish. International Journal of Language and Linguistics, 1(1), 1-6.

Ambati, B. R., Reddy, S., and Kilgarriff, A., 2012. Word Sketches for Turkish. In LREC, 2945-2950.

Baisa, V., and Suchomel, V., 2012. Large corpora for Turkic languages and

unsupervised morphological analysis. In Proceedings of the Eighth conference on International Language Resources and Evaluation (LREC'12), Istanbul, Turkey. European Language Resources Association (ELRA).

Bektaş, Y., and Özel, S. A., 2018. The Effect of POS Tag Information on Sentence Boundary Detection in Turkish Texts. In 2018 Innovations in Intelligent Systems and Applications Conference (ASYU), Adana, 1-5.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C., 2003. A neural probabilistic language model. journal of machine learning research, Vol. 3, No.

Biewer, C., Hundt, M., and Nesselhauf, N., 2007. Corpus linguistics and the Web. Rodopi, 551–563.

Böler, T., 2019. Türkiye Türkçesi söz dizimi. Kesit Yayınları, p400.

Brill, E., and Mooney, R. J.,1997. An overview of empirical natural language processing. AI magazine, 18(4), 13-13.

Broomhead, D. S., and Lowe, D., 1988. Radial basis functions, multi-variable functional interpolation and adaptive networks. Royal Signals and Radar Establishment Malvern (United Kingdom).

Burnard, L., 1998. The BNC Handbook: Exploring the British National Corpus with SARA. Edinburgh University Press, Edinburgh, 256p.

Che, X., Wang, C., Yang, H., and Meinel, C., 2016. Punctuation prediction for unsegmented transcript based on word vector. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), 654-658.

Collobert, R., and Weston, J., 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, 160-167.

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

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P., 1992. A practical part-of-speech

tagger. In Third conference on applied natural language processing, 133-140.

Dalkılıç, G., & Çebi, Y. (2004, October). Zipf's law and Mandelbrot's constants for Turkish language using Turkish corpus (TurCo). In International Conference on Advances in Information Systems (pp. 273-282). Springer, Berlin, Heidelberg.

Demirhan, U. U., 2013. A description of the verb gel-with special reference to pattern grammar (Master's thesis, Sosyal Bilimler Enstitüsü).

Devlin, J., Chang, M. W., Lee, K., and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dinçer, B. T., and Karaoğlan, B., 2004. Sentence boundary detection in Turkish. In International Conference on Advances in Information Systems, Springer, Berlin, Heidelberg. 255-262.

Fillmore, C., Ide, N., Jurafsky, D., and Macleod, C., 1998. An american national corpus: A Proposal. In Proceedings of the First Annual Conference on Language Resources and Evaluation 965-969.

Francis, W. N., and Kucera, H.,1979. Brown corpus manual. Letters to the Editor.

Galves, C., and Britto, H., 1999. A construção do Corpus Anotado do Português Histórico Tycho Brahe: o sistema de anotação morfológica. Proceedings of the IV PROPOR. Évora. Universidade de Évora, 55-67.

Gers, F. A., Schmidhuber, J., and Cummins, F., 2000. Learning to forget: Continual prediction with LSTM. Neural computation, 12(10), 2451-2471.

Gers, F. A., Schraudolph, N. N., and Schmidhuber, J., 2002. Learning precise timing with LSTM recurrent networks. Journal of machine learning research, 3(Aug), 115-143.

Gillick, D., 2009. Sentence boundary detection and the problem with the US. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for

Computational Linguistics, Companion Volume: Short Papers, 241-244.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., ... and Smith, N. A., 2010. Part-of-speech tagging for twitter: Annotation, features, and experiments. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Greenbaum, S., and Nelson, G.,1996. The international corpus of English (ICE) project. World Englishes, 15(1), 3-15.

Grishman, R., 1986. Computational linguistics: an introduction. Cambridge University Press, New York, 193p.

Han, J., and Kamber, M., 2006. Data Mining Concepts and Techniques, 2nd ed., Morgan Kaufmann Publishers, San Francisco, p800.

Han, J., Pei, J., and Yin, Y., 2000. Mining frequent patterns without candidate generation. ACM sigmod record, 29(2), 1-12.

Han, J., Pei, J., Yin, Y., and Mao, R., 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data mining and knowledge discovery, 8(1), 53-87.

Hand, D. J., and Yu, K., 2001. Idiot's Bayes—not so stupid after all?. International statistical review, 69(3), 385-398.

Hilden, J., 1984. Statistical diagnosis based on conditional independence does not require it. Computers in biology and medicine, 14(4), 429-435.

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

Japkowicz, N., 2000. Learning from imbalanced data sets: a comparison of various strategies. In AAAI workshop on learning from imbalanced data sets, AAAI Press Menlo Park, CA., (Vol. 68, pp. 10-15).

Kempe, A., 1993. A stochastic Tagger and an Analysis of Tagging Errors. Internal paper. Institute for Computational Linguistics, University of Stuttgart.

Kim, J. D., Ohta, T., Tateisi, Y., and Tsujii, J. I., 2003. GENIA corpus—a semantically annotated corpus for bio-textmining. Bioinformatics,

19(suppl_1), 180-182.

Kiss, T., and Strunk, J., 2006. Unsupervised multilingual sentence boundary detection. Computational linguistics, 32(4), 485-525.

Koehn, P., 2005. Europarl: A parallel corpus for statistical machine translation. In Proceedings of machine translation summit x: papers, 79-86.

Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

Kucera, H., and Francis, W. N., 1979. Brown corpus manual. Letters to the Editor, 5(2), 7.

Lee, D. Y., 2010. What corpora are available. The Routledge handbook of corpus linguistics, Roudledge Press, New York, 650p.

Liu, B., 2011. Web data mining: exploring hyperlinks, contents, and usage data (Vol. 1). Berlin: springer.

Maggiori, E., Tarabalka, Y., Charpiat, G., and Alliez, P., 2016. Convolutional neural networks for large-scale remote-sensing image classification. IEEE Transactions on geoscience and remote sensing, 55(2), 645-657.

McEnery, T., and Hardie, A., 2011. Corpus linguistics: Method, theory and practice. Cambridge University Press, 294p

Mete, F., 2015. Türkçe kurallı cümle yapısının graf teori ile gösterilmesi. Journal of Turkish Studies, 10(16).

Mikheev, A., 2000. Tagging sentence boundaries. In 1st Meeting of the North American Chapter of the Association for Computational Linguistics.

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

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S., 2010. Recurrent neural network based language model. In Interspeech, Vol. 2, No. 3, 1045-1048.

Mundluru, D., 2008. Automatically constructing wrappers for effective and efficient

Web information extraction. University of Louisiana at Lafayette.

Niuniu, X., and Yuxun, L., 2010. Notice of Retraction: Review of decision trees. In 2010 3rd international conference on computer science and information technology, Vol. 5, 105-109).

Özbey, C., and Dinçsoy, Ö., 2019. Sentence Boundary Detection in Turkish News with Regular Expressions. In 2019 27th Signal Processing and Communications Applications Conference (SIU), 1-4.

Palmer, D. D., and Hearst, M. A., 1997. Adaptive multilingual sentence boundary disambiguation. Computational linguistics, 23(2), 241-267.

Pawlak, Z., 2003. A rough set view on Bayes' theorem. International Journal of Intelligent Systems, 18(5), 487-498.

Polpinij, J., Srikanjanapert, N., and Sopon, P., 2017. Word2Vec approach for sentiment classification relating to hotel reviews. In International Conference on Computing and Information Technology, Springer, Cham, 308-316.

Quinlan, J. R., 1986. Induction of decision trees. Machine learning, 1(1), 81-106.

Quinlan, J. R., 1993. C4. 5: Programming for machine learning. Morgan Kauffmann, 38(48), 49.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A., 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision, Springer, Cham525-542.

Ratnaparkhi, A., 1996. A maximum entropy model for part-of-speech tagging. In EMNLP, 133-142.

Read, J., Dridan, R., Oepen, S., and Solberg, L. J., 2012. Sentence boundary detection: A long solved problem? In Proceedings of COLING 2012: Posters, 985-994.

Reynar, J. C., and Ratnaparkhi, A., 1997. A maximum entropy approach to identifying sentence boundaries. arXiv preprint cmp-lg/9704002.

Riley, M., 1989. Some applications of tree-based modelling to speech and language.

In Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, 15-18.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1985. Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986. Learning representations by back-propagating errors. nature, 323(6088), 533-536.

Sak, H., Güngör, T., and Saraçlar, M., 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In International conference on natural language processing, Springer, Berlin, Heidelberg, 417-427.

Santoso, J., Setiawan, E. I., Purwanto, C. N., Haviluddin, H., and Kurniawan, F., 2021. Indonesian Sentence Boundary Detection using Deep Learning Approaches. Knowledge Engineering and Data Science, 4(1).

Say, B., Zeyrek, D., Oflazer, K., and Özge, U., 2002. Development of a corpus and a treebank for present-day written Turkish. In Proceedings of the eleventh international conference of Turkish linguistic, Eastern Mediterranean Universitys, 183-192.

Schmid, H., 1994. Part-of-speech tagging with neural networks. arXiv preprint cmp-lg/9410018.

Schuster, M., and Paliwal, K. K., 1997. Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11), 2673-2681.

Schweter, S., and Ahmed, S., 2019. Deep-EOS: General-Purpose Neural Networks for Sentence Boundary Detection. In KONVENS.

Sinclair, J., 2005. Corpus and text-basic principles. Developing linguistic corpora: A guide to good practice, 92, 1-16.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B., 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In ACL (1), 1555-1565.

Tateisi, Y., Yakushiji, A., Ohta, T., and Tsujii, J. I., 2005. Syntax Annotation for the GENIA corpus. In Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts.

TDK, 2022.http://www.tdk.gov.tr/icerik/yazim-kurallari/noktalama-isaretleri-aciklamalar, 13/06/2022

Tomanek, K., Wermter, J., and Hahn, U., 2007. Sentence and token splitting based on conditional random fields. In Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, Vol. 49, p 57.

Tosun, C., 2005. Türkçe'nin yabancı dil olarak öğretilmesi. Journal of Language and Linguistic Studies, 1(1), 22-28.

Tür, G., 2000. A Statistical Information Extraction System for Turkish. Bilkent University, Department of Computer Engineering (Doctoral dissertation, Ph. D. Thesis, Ankara, Turkey).

Tyers, F. M., and Alperen, M. S., 2010. South-east european times: A parallel corpus of balkan languages. In Proceedings of the LREC workshop on exploitation of multilingual resources and tools for Central and (South-) Eastern European Languages, 49-53.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... and Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems, 30.

Waltz, D. L., 1978. An English language question answering system for a large relational database. Communications of the ACM, 21(7), 526-539.

Wiki, 2021, https://en.wikipedia.org/wiki/File:Recurrent_neural_network_unfold.svg#filehistory, 13/06/2022

Wong, D. F., Chao, L. S., and Zeng, X., 2014. isentenizer-: Multilingual sentence boundary detection model. The Scientific World Journal.

Wong, F., and Chao, S., 2010. iSentenizer: An incremental sentence boundary classifier. In Proceedings of the 6th International Conference on Natural

Language Processing and Knowledge Engineering (NLPKE-2010), 1-7

Wong, P. C., Whitney, P., and Thomas, J., 1999. Visualizing association rules for text mining. In Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis' 99), 120-123.

Woods, W., 1972, The lunar sciences natural language information system. BBN report.

Zaki, M. J., Parthasarathy, S., Ogihara, M., and Li, W., 1997. Parallel algorithms for discovery of association rules. Data mining and knowledge discovery, 1(4), 343-373.

Zhang, Y., and Wallace, B., 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

**CURRICULUM VITAE**

Yasin BEKTAŞ. He received a B.S. degree from the Department of Computer Engineering from Mersin University, Turkey, in 2002. He earned a M.S. degree from the Department of Electrical and Electronics Engineering at Mersin University in Mersin, Turkey, in 2008.

Since 2002, he has been working as the lecturer at Vocational School of Erdemli, Mersin, Turkey.

His research interests are in machine learning, software engineering, data mining and natural language processing.