

**T.C  
İNÖNÜ ÜNİVERSİTESİ  
FEN BİLİMLERİ ENSTİTÜSÜ**

**BÜYÜK VERİDE DERİN ÖĞRENME ALGORİTMALARI KULLANILARAK  
METİN ANALİZİNİN GERÇEKLEŞTİRİLMESİ**

**MURAT AYDOĞAN**

**DOKTORA TEZİ  
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

**ARALIK 2019**

**T.C**  
**İNÖNÜ ÜNİVERSİTESİ**  
**FEN BİLİMLERİ ENSTİTÜSÜ**

**BÜYÜK VERİDE DERİN ÖĞRENME ALGORİTMALARI KULLANILARAK  
METİN ANALİZİNİN GERÇEKLEŞTİRİLMESİ**

**MURAT AYDOĞAN**

**DOKTORA TEZİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

**ARALIK 2019**

Tezin Başlığı: Büyük Veride Derin Öğrenme Algoritmaları Kullanılarak Metin Analizinin Gerçekleştirilmesi

Tezi Hazırlayan: Murat AYDOĞAN

Sınav Tarihi: 5/12/2019

Yukarıda adı geçen tez jürimizce değerlendirilerek Bilgisayar Mühendisliği Ana Bilim Dalında Doktora Tezi olarak kabul edilmiştir.

### Sınav Jüri Üyeleri

**Tez Danışmanı: Prof. Dr. Ali KARCI**  
İnönü Üniversitesi

.....

**Prof. Dr. A. Bedri ÖZER**  
Turgut Özal Üniversitesi

.....

**Doç. Dr. B. Baykant ALAGÖZ**  
İnönü Üniversitesi

.....

**Dr. Öğr. Üyesi A. Fatih KOCAMAZ**  
İnönü Üniversitesi

.....

**Dr. Öğr. Üyesi Faruk SERİN**  
Munzur Üniversitesi

.....

**Prof. Dr. Kazım TÜRK**  
Enstitü Müdürü

## ONUR SÖZÜ

Doktora Tezi olarak sunduğum “Büyük Veride Derin Öğrenme Algoritmaları Kullanılarak Metin Analizinin Gerçekleştirilmesi” başlıklı bu çalışmanın bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın tarafımdan yazıldığını ve yararlandığım bütün kaynakların hem metin içinde hem de kaynakçada yöntemine uygun biçimde gösterilenlerden oluştuğunu belirtir, bunu onurumla doğrularım.

Murat AYDOĞAN

## ÖZET

Doktora Tezi

### BÜYÜK VERİDE DERİN ÖĞRENME ALGORİTMALARI KULLANILARAK METİN ANALİZİNİN GERÇEKLEŞTİRİLMESİ

Murat AYDOĞAN

İnönü Üniversitesi  
Fen Bilimleri Enstitüsü  
Bilgisayar Mühendisliği Anabilim Dalı  
x + 93 sayfa  
2019

Danışman: Prof. Dr. Ali KARCI

Bilişim dünyasında yüksek hızda yaşanan gelişmeler ve internet kullanımının çok yaygın hale gelmesi ile birlikte, dijital platformlarda üretilen verinin çeşitliliği ve miktarı artmıştır. Üretilen bu büyük verinin çoğunluğu metinlerden oluşan içeriklerdir. Ancak her geçen gün artan metin verilerini geleneksel yöntemlerle işlemek zor bir problem haline gelmiştir. Bu nedenle büyük veri teknolojileri ve özellikle büyük veriler üzerinde oldukça başarılı performans gösteren derin sinir ağları ve sinir ağı temelli kelime temsil yöntemleri geliştirilmiştir.

Bu tez çalışmasında kelime temsil yöntemleri, büyük veri teknolojileri ile birlikte kullanılmış derin öğrenme mimarileri üzerinde detaylı analizler yapılmıştır. Yapılan çalışmalar incelendiğinde başta İngilizce olmak üzere birçok doğal dile özgü çalışmalar yapıldığı ancak Türkçe çalışmaların sayısının yeterli olmadığı görülmüştür. Bu nedenle çalışmanın hedef dili olarak Türkçe tercih edilmiş bununla birlikte tez çalışmasında üç uygulama geliştirilmiştir ve iki özgün yöntem önerilmiştir. İlk uygulamada çalışmaların yapılacağı ortamı belirlemek için büyük veri uygulaması yapılmıştır. İkinci uygulamada metin işleme öncesinde ön işleme çalışmaları yapılmıştır. Bu kapsamda TF (Term Frequency) - IDF (Inverse Document Frequency) yöntemi ile ilk kez Türkçe için stopwords listesi oluşturulmuştur. Üçüncü uygulamada ise oldukça büyük Türkçe etiketsiz verilerden oluşan bir veri seti (Dataset-1) üretilmiştir. Bu veri seti üzerinde kelime temsil yöntemleri kullanılarak kelime vektörleri eğitilmiş ve farklı kelime temsil yöntemlerinin başarımları kıyaslanmıştır. Üçüncü uygulama için de yaklaşık 1,5 milyon veri ve 10 sınıftan oluşan ikinci bir Türkçe veri seti (Dataset-2) üretilmiştir. Bu veri seti üzerinde, kelime vektörleri, önceden eğitilmiş kelime vektörleri olarak

transfer öğrenme yöntemi ile farklı derin öğrenme mimarileri üzerinde metin sınıflandırma problemi için kullanıldığı bir yöntem önerilmiştir. Önerilen bu yöntem ile hemen hemen tüm modeller üzerinde mevcut başarımlar %5-%7 arasında geliştirilmiştir. İkinci yöntem olarak sözlük yöntemi adı verilen yeni bir yöntem önerilmiştir. Türkçe için hali hazırda geliştirilmiş bir yazım denetleyicisi olmadığı için bu yöntem ile Dataset-2 üzerindeki yanlış yazılmış kelimeler tespit edilmiş ve bir derin öğrenme modeli olan LSTM (Long Short Term Memory) ile bu kelimelerin yerine doğru kelimeler tespit edilmeye çalışılmıştır. Uygulama sonucunda elde edilen sınıflandırma başarımları analiz edildiğinde yaklaşık 55.000 hatalı kelimenin doğru kelime ile değiştirildiği ve başarımların %8.68 geliştirildiği görülmüştür.

Bu tez çalışmasıyla birlikte Türkçe metin işleme çalışmalarına katkı sağlamak amacıyla iki adet büyük Türkçe veri seti üretilmiştir. Ayrıca bu veri setleri üzerinde şimdiye kadar eğitilmiş en büyük Türkçe kelime vektörleri üretilerek araştırmacıların kullanımına açık halde paylaşılmıştır.

**ANAHTAR KELİMELER:** Büyük Veri, Derin Öğrenme, Metin İşleme, Kelime Temsili, Önceden Eğitilmiş Kelime Vektörleri, Sözlük Yöntemi

## **ABSTRACT**

Ph.D. Thesis

### **APPLICATION OF TEXT ANALYSIS PROCESSING USING DEEP LEARNING ALGORITHMS IN BIG DATA**

Murat AYDOĞAN

İnönü University  
Graduate School of Natural and Applied Sciences  
Department of Computer Engineering  
x + 93 pages  
2019

Supervisor: Prof. Dr. Ali KARCI

With the high-speed developments in the IT world and the widespread use of the Internet, the diversity and amount of data produced on digital platforms has increased. The majority of this big data generated is textual content. However, it has become a difficult problem to process the increasing text data with traditional methods. For this reason, deep neural networks and neural network-based word embedding methods have been developed that perform highly successfully on big data technologies and especially big data.

In this thesis, detailed analysis has been made on deep learning architectures used word embedding methods with big data technologies. When the studies were examined, it was seen that there were many natural language specific studies, especially English, but the number of Turkish studies was not sufficient. Therefore, Turkish was chosen as the target language of the study. However, three applications were developed in the thesis and two novel methods were proposed. In the first application, a big data application was made to determine the platform in which the studies would be conducted. In the second application, preprocessing studies were performed before text processing. In this context, the stopwords list for Turkish was generated for the first time by TF (Term Frequency) - IDF (Inverse Document Frequency) method. In the third application, a dataset (Dataset-1) consisting of very large Turkish unlabeled data has been generated. Word vectors were trained on this dataset using word embedding methods and the performances of different word embedding methods were compared. For the third application, a second Turkish dataset (Dataset-2) consisting of approximately 1,5 million data and 10 classes were generated. A method has been proposed on this data set where word vectors are used for the problem of text classification on different deep learning architectures with the transfer

learning method as pre-trained word vectors. With this proposed method, current performance values on almost all models have been improved between 5-7%. As a second method, a new method called the dictionary method has been proposed. Since there is no spelling checker developed for Turkish, the misspelled words on Dataset-2 have been identified and LSTM (Long Short Term Memory), which is a deep learning model, has tried to identify the correct words instead. When the classification performance obtained as a result of the analysis was analyzed, it was seen that approximately 55.000 incorrect words were replaced with the correct words and the performance value was improved by 8.68%.

With this thesis, two large Turkish datasets were generated in order to contribute to Turkish text processing. In addition, the largest Turkish word vectors ever trained on these datasets were generated and shared open to researchers.

**KEYWORDS:** Big Data, Deep Learning, Text Processing, Word Embedding, Pre-trained Word Embeddings, Dictionary Method

## TEŞEKKÜR

Tez çalışmamın her aşamasında bilgi ve tecrübeleriyle bana destek olan, öğrencisi olmaktan onur duyduğum bir danışmandan ziyade bir baba gibi gördüğüm danışman hocam Sayın Prof. Dr. Ali KARCI 'ya;

Çalışmalarım boyunca, bana destek olan ve beni teşvik ederek yönlendiren Tez İzleme Komitesinde yer alan değerli Doç. Dr. B. Baykant ALAGÖZ ve Dr. Öğr. Üyesi A. Fatih KOCAMAZ hocalarıma;

İhtiyaç duyduğumda her zaman bana vakit ayırıp yardımcı olan Doç. Dr. M. Fatih TALU, Doç. Dr. Yılmaz KAYA ve Veysel KOCAMAN hocalarıma;

Birlikte görev yaptığım Öğr. Gör. Abdullah ŞENER, Öğr. Gör. Doygun DEMİROL, Arş. Gör. Muammer TÜRKOĞLU, Öğr. Gör. Cihangir KAÇMAZ ve destekleri için kıymetli Genç Meslek Yüksekokulu akademik personeline;

Son olarak aileme, adeta benimle birlikte doktora yapan, doktora eğitimim süresince karşılaştığım tüm zorluklarda yanımda olan ve beni her zaman destekleyen anlayış ve sabır gösteren eşim Duygu AYDOĞAN'a ve dünyaya gelmesiyle bana büyük şans getiren sevgili kızım Miray'a

Sonsuz teşekkürlerimi sunarım...

## İÇİNDEKİLER

|                                                                           |      |
|---------------------------------------------------------------------------|------|
| ÖZET .....                                                                | i    |
| ABSTRACT .....                                                            | iii  |
| TEŞEKKÜR .....                                                            | v    |
| ŞEKİLLER DİZİNİ .....                                                     | viii |
| ÇİZELGELER DİZİNİ .....                                                   | ix   |
| SİMGELER VE KISALTMALAR .....                                             | x    |
| 1. GİRİŞ .....                                                            | 1    |
| 1.1. Amaç ve Kapsam .....                                                 | 2    |
| 1.2. Literatür Çalışması .....                                            | 4    |
| 1.2.1. Türkçe İçin Yapılmış Çalışmalar .....                              | 4    |
| 1.2.2. Türkçe Dışındaki Dillerde Yapılmış Çalışmalar .....                | 6    |
| 1.3. Tezde Yapılan Çalışmalar ve Literatüre Katkısı .....                 | 9    |
| 1.4. Tezin Yapısı .....                                                   | 10   |
| 2. BÜYÜK VERİ .....                                                       | 12   |
| 2.1. Büyük Veri Bileşenleri .....                                         | 12   |
| 2.2. Büyük Veri Teknolojileri .....                                       | 14   |
| 2.2.1. Hadoop .....                                                       | 14   |
| 2.2.2. Apache Spark Projesi .....                                         | 15   |
| 3. METİN İŞLEME .....                                                     | 18   |
| 3.1. Metin İşlemeye Hazırlık ve Ön İşleme .....                           | 19   |
| 3.1.1. İşaretleme (Tokenization) .....                                    | 20   |
| 3.1.2. Gövdeleme (Stemming, Lemmatization) .....                          | 22   |
| 3.1.3. Sözlük Oluşturma .....                                             | 22   |
| 3.2. Kelime Temsil Yöntemleri .....                                       | 22   |
| 3.2.1. TF (Term Frequency) - IDF (Inverse Document Frequency) .....       | 24   |
| 3.2.2. Kelime Çantası ( Bag of Words) .....                               | 25   |
| 3.2.3. Gömme (Embedding) Yöntemleri .....                                 | 25   |
| 3.2.4. CBOW (Continous Bag of Words) .....                                | 27   |
| 3.2.5. SG (Skip-Gram Model) .....                                         | 28   |
| 3.2.6. Glove .....                                                        | 30   |
| 4. DERİN ÖĞRENME .....                                                    | 31   |
| 4.1. Yapay Zekâ .....                                                     | 31   |
| 4.2. Makine Öğrenmesi .....                                               | 31   |
| 4.3. Derin Sinir Ağları .....                                             | 32   |
| 4.4. Sinir Ağlarının Yapısı .....                                         | 32   |
| 4.4.1. Evrişimli Sinir Ağları - Convolutional Neural Networks (CNN) ..... | 34   |
| 4.4.2. Yinelemeli Sinir Ağları-Recurrent Neural Networks (RNN) .....      | 35   |
| 4.4.3. Uzun Kısa Süreli Bellek-Long Short Term Memory (LSTM) .....        | 36   |
| 4.4.4. Kapılı Tekrarlı Birim - Gated Recurrent Unit (GRU) .....           | 37   |
| 4.4.5. İki Yönlü RNN -Bidirectional RNN (BRNN) .....                      | 37   |
| 4.4.6. İki Yönlü LSTM - Bidirectional LSTM (BLSTM) .....                  | 38   |
| 4.5. Derin Sinir Ağlarında Hiper-parametre Ayarlamaları .....             | 39   |
| 4.5.1. Mini-batch Boyutu .....                                            | 39   |
| 4.5.2. Aktivasyon Fonksiyonları .....                                     | 39   |
| 4.5.3. Öğrenme Hızı .....                                                 | 40   |
| 4.5.4. Optimizasyon Algoritmasının Seçimi .....                           | 40   |
| 4.5.5. Tur (Epoch) Sayısı .....                                           | 41   |
| 4.5.6. Seyreltme (Dropout) Değeri .....                                   | 41   |

|        |                                                              |    |
|--------|--------------------------------------------------------------|----|
| 4.5.7. | Katman Sayısı ve Nöron Sayısı .....                          | 41 |
| 4.5.8. | Hata ve Kayıp Fonksiyonu .....                               | 42 |
| 4.6.   | Metin Verileri İçin Derin Öğrenme .....                      | 42 |
| 4.6.1. | Kelime Temsillerinin Kullanımı .....                         | 42 |
| 4.6.2. | Kelime Vektörleri .....                                      | 43 |
| 4.6.3. | Önceden Eğitilmiş Kelime Vektörleri .....                    | 43 |
| 5.     | GELİŞTİRİLEN UYGULAMALAR .....                               | 45 |
| 5.1.   | Uygulama 1: Apache Spark ile Büyük Veri Uygulaması .....     | 45 |
| 5.1.1. | Uygulamanın Amacı .....                                      | 45 |
| 5.1.2. | Uygulamanın Geliştirilmesi .....                             | 46 |
| 5.1.3. | Uygulamanın Sonuçları .....                                  | 50 |
| 5.2.   | Materyal .....                                               | 51 |
| 5.2.1. | Dataset-1 .....                                              | 51 |
| 5.2.2. | Dataset-2 .....                                              | 51 |
| 5.3.   | Uygulama 2: Önışlem Uygulaması .....                         | 52 |
| 5.3.1. | Uygulamanın Amacı .....                                      | 52 |
| 5.3.2. | Uygulamanın Geliştirilmesi .....                             | 53 |
| 5.3.3. | Uygulamanın Sonuçları .....                                  | 54 |
| 5.4.   | Uygulama 3: Kelime Vektörlerinin Üretilmesi .....            | 54 |
| 5.4.1. | Uygulamanın Amacı .....                                      | 55 |
| 5.4.2. | Uygulamanın Geliştirilmesi .....                             | 55 |
| 5.4.3. | Uygulamanın Sonuçları .....                                  | 63 |
| 6.     | ÖNERİLEN YÖNTEMLER VE DENEYSEL SONUÇLAR .....                | 64 |
| 6.1.   | Önerilen Yöntem 1: Önceden Eğitilmiş Kelime Vektörleri ..... | 64 |
| 6.1.1. | Yöntemin Amacı .....                                         | 64 |
| 6.1.2. | Uygulamanın Geliştirilmesi .....                             | 64 |
| 6.1.3. | Uygulamada Elde Edilen Bulgular .....                        | 72 |
| 6.1.4. | Uygulamanın Sonuçları .....                                  | 76 |
| 6.2.   | Önerilen Yöntem 2: Sözlük Yöntemi .....                      | 76 |
| 6.2.1. | Problemin Tanımı .....                                       | 76 |
| 6.2.2. | Sözlük Yöntemi .....                                         | 77 |
| 6.2.3. | Uygulamanın Sonuçları .....                                  | 80 |
| 7.     | SONUÇ VE ÖNERİLER .....                                      | 82 |
| 8.     | KAYNAKLAR .....                                              | 85 |
|        | ÖZGEÇMİŞ .....                                               | 93 |

## ŞEKİLLER DİZİNİ

|             |                                                                     |    |
|-------------|---------------------------------------------------------------------|----|
| Şekil 1.1.  | Çalışma konularının işlenme biçimi .....                            | 2  |
| Şekil 2.1   | Büyük veri bileşenleri .....                                        | 13 |
| Şekil 2.2   | HDFS yapısı .....                                                   | 14 |
| Şekil 2.3.  | Map-Reduce çalışma şekli .....                                      | 15 |
| Şekil 2.4.  | Spark ve Hadoop karşılaştırılması .....                             | 16 |
| Şekil 2.5.  | Apache Spark Mimarisi .....                                         | 17 |
| Şekil 3.1.  | Kelime temsil yöntemleri ile ilişkilerin tespit edilmesi .....      | 27 |
| Şekil 3.2.  | CBOW modelinin yapısı .....                                         | 27 |
| Şekil 3.3.  | Skip-Gram modelinin yapısı .....                                    | 28 |
| Şekil 3.4.  | Skip-Gram yönteminin uygulanması .....                              | 29 |
| Şekil 4.1.  | Yapay zekânın görünümü .....                                        | 31 |
| Şekil 4.2.  | Sinir ağının çalışma mantığı .....                                  | 33 |
| Şekil 4.3.  | Yapay sinir ağının mimarisi .....                                   | 33 |
| Şekil 4.4.  | RNN modelinin yapısı .....                                          | 36 |
| Şekil 4.5.  | LSTM modelinin yapısı .....                                         | 36 |
| Şekil 4.6.  | GRU modelinin yapısı .....                                          | 37 |
| Şekil 4.7.  | BRNN modelinin yapısı .....                                         | 38 |
| Şekil 4.8.  | BLSTM modelinin yapısı .....                                        | 38 |
| Şekil 5.1.  | Tez çalışmasının genel yapısı .....                                 | 45 |
| Şekil 5.2.  | Verisetindeki mail formatı .....                                    | 46 |
| Şekil 5.3.  | RDD Yapısı .....                                                    | 47 |
| Şekil 5.4.  | Ortalama çalışma süreleri .....                                     | 50 |
| Şekil 5.5.  | Mikolov tarafından tanımlanmış kelime vektörlerinin gösterimi ..... | 57 |
| Şekil 5.6.  | Vektörün 2boyutlu düzlemde gösterimi .....                          | 59 |
| Şekil 5.7.  | Vektörler arasındaki yakınlık ve ilişkiler .....                    | 62 |
| Şekil 5.8.  | Kelime vektörlerinin eğitim süreleri .....                          | 62 |
| Şekil 6.1.  | Önerilen yöntemin genel yapısı .....                                | 65 |
| Şekil 6.2.  | Uygulanan CNN modelinin yapısı .....                                | 66 |
| Şekil 6.3.  | CNN modelinin doğruluk grafiği .....                                | 66 |
| Şekil 6.4.  | Uygulanan LSTM & BLSTM modelinin yapısı .....                       | 67 |
| Şekil 6.5.  | LSTM & BLSTM modellerinin doğruluk grafikleri .....                 | 68 |
| Şekil 6.6.  | Uygulanan GRU modelinin yapısı .....                                | 68 |
| Şekil 6.7.  | GRU modelinin doğruluk grafiği .....                                | 69 |
| Şekil 6.8.  | Uygulanan RNN modelinin yapısı .....                                | 69 |
| Şekil 6.9.  | RNN modelinin doğruluk grafiği .....                                | 70 |
| Şekil 6.10. | Uygulanan CNN-LSTM modelinin yapısı .....                           | 71 |
| Şekil 6.11. | CNN-LSTM modelinin doğruluk grafiği .....                           | 71 |
| Şekil 6.12. | Modellerin doğruluk ve kayıp fonksiyon grafikleri .....             | 72 |
| Şekil 6.13. | Tüm modellerin PWV yöntemi ile ROC eğrisi grafiği .....             | 73 |
| Şekil 6.14. | Farklı öğrenme hızı değerleri için doğruluk değerleri .....         | 75 |
| Şekil 6.15. | Sözlük yönteminin çalışma yapısı .....                              | 78 |
| Şekil 6.16. | Transfer öğrenme ile sözlük kullanımı .....                         | 78 |
| Şekil 6.17. | LSTM ile kelime tahmini .....                                       | 79 |
| Şekil 6.18. | Benzer kelimelerin listesi .....                                    | 79 |

## ÇİZELGELER DİZİNİ

|               |                                                                         |    |
|---------------|-------------------------------------------------------------------------|----|
| Çizelge 2.1   | Ölçü Birimleri .....                                                    | 13 |
| Çizelge 3.1.  | CBOW modeli .....                                                       | 28 |
| Çizelge 3.2.  | Skip-Gram modeli .....                                                  | 30 |
| Çizelge 4.1.  | Aktivasyon fonksiyonları .....                                          | 40 |
| Çizelge 5.1.  | Dönüştürülen veri seti .....                                            | 47 |
| Çizelge 5.2.  | Karışıklık matrisi .....                                                | 48 |
| Çizelge 5.3.  | Başarım oranları .....                                                  | 48 |
| Çizelge 5.4.  | Çalışma süreleri .....                                                  | 49 |
| Çizelge 5.5.  | Dataset-2'nin yapısı .....                                              | 52 |
| Çizelge 5.6.  | Dataset-2'nin kategorileri .....                                        | 52 |
| Çizelge 5.7.  | TF-IDF ağırlık matrisleri .....                                         | 54 |
| Çizelge 5.8.  | Word2Vec yönteminde kullanılan bazı parametreler .....                  | 56 |
| Çizelge 5.9.  | Word2Vec yöntemi ile eğitilen kelime benzerliklerinin incelenmesi ..... | 58 |
| Çizelge 5.10. | Word2Vec yöntemi ile eğitilen kelime benzerliklerinin incelenmesi ..... | 60 |
| Çizelge 5.11. | Eşleştirme sorguları .....                                              | 61 |
| Çizelge 6.1.  | Tüm modeller için PWV ile elde edilen sonuçlar .....                    | 74 |
| Çizelge 6.2.  | PWV ile sınıflandırma sonuçları .....                                   | 75 |
| Çizelge 6.3.  | Sözlük yöntemi kullanılmadan elde edilen sonuçlar .....                 | 78 |
| Çizelge 6.4.  | Sözlük yöntemi kullanılarak elde edilen sonuçlar .....                  | 80 |

## SİMGELER VE KISALTMALAR

|       |                                                                 |
|-------|-----------------------------------------------------------------|
| BLSTM | : İki yönlü LSTM (Bidirectional LSTM)                           |
| BOW   | : Kelime Torbaları (Bag of Words)                               |
| BRNN  | : İki yönlü RNN (Bidirectional RNN)                             |
| CBOW  | : Devamlı Kelime Torbaları (Continuous Bag of Words)            |
| CNN   | : Evrişimli Sinir Ağı (Convolutional Neural Network)            |
| DL    | : Derin Öğrenme (Deep Learning)                                 |
| DVM   | : Destek Vektör Makinesi (Support Vector Machine)               |
| FN    | : Yanlış Negatif (False Negative)                               |
| FP    | : Yanlış Pozitif (False Positive)                               |
| GPU   | : Grafiksel İşleme Birimi (Graphics Processing Unit)            |
| GRU   | : Kapılı Tekrarlı Birim (Gated Recurrent Unit)                  |
| HDFS  | : Hadoop Dağıtık Dosya Sistemi (Hadoop Distributed File System) |
| IDF   | : Ters Doküman Sıklığı (Inverse Document Frequency)             |
| KA    | : Karar Ağacı (Decision Tree)                                   |
| LSTM  | : Uzun Kısa Süreli Bellek Ağı (Long Short-Term Memory)          |
| MR    | : Map Reduce                                                    |
| NB    | : Naive Bayes                                                   |
| NLP   | : Doğal Dil İşleme (Natural Language Processing)                |
| NLTK  | : Doğal Dil İşleme Aracı (Natural Language Toolkit)             |
| PWV   | : Önceden Eğitilmiş Kelime Temsili (Pre-trained Word Vectors)   |
| RDD   | : Esnek Dağıtık Veri Setleri (Resilient Distributed Datasets)   |
| ReLU  | : Doğrultulmuş Lineer Birim (Rectified Linear Unit)             |
| RNN   | : Tekrarlayan Sinir Ağı (Recurrent Neural Network)              |
| RO    | : Rastgele Orman (Random Forest)                                |
| SG    | : Atlama-Gram (Skip-Gram)                                       |
| SGD   | : Olasılıksal Gradyan Azaltma (Stochastic Gradient Descent)     |
| TF    | : Terim Sıklığı (Term Frequency)                                |
| TL    | : Transfer Öğrenme (Transfer Learning)                          |
| TN    | : Doğru Negatif (True Negative)                                 |
| TP    | : Doğru Pozitif (True Positive)                                 |

# 1. GİRİŞ

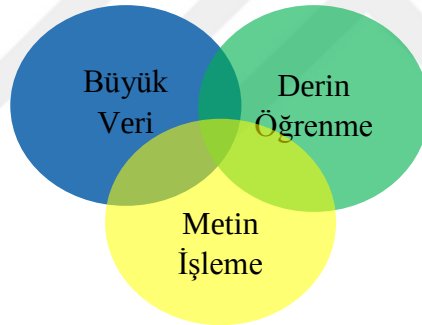
Bilişim teknolojilerindeki gelişmelere bağlı olarak internet teknolojileri de büyük bir gelişme göstermiş ve bu gelişmelerle birlikte gelen sosyal medya çağı mobil cihazların kullanımının yaygınlaşmasına yol açmıştır. Bilişim teknolojilerindeki bu ilerlemelere paralel olarak donanım maliyetlerinin giderek ucuzlaması, üretilen ve depolanan veri miktarında devasa denilebilecek boyutta bir artış meydana getirmiş böylece büyük veri kavramı ortaya çıkmıştır [1]. Günümüze kadar gelen süreçte bilgi ve veri miktarı her geçen gün artmıştır. Özellikle son yıllarda bu miktar katlanarak büyümüştür. Hatta Sintef verilerine göre tüm insanlık tarihi boyunca oluşturulan verinin çok daha fazlası son iki yıl içerisinde üretilmiştir [2]. Bununla birlikte sosyal medya devrimi olarak tanımlanan gelişmeler bu artışın oluşmasında çok önemli bir rol oynamıştır. Veri bilimciler, 2020 yılına gelindiğinde ise yaşayan her insan için saniyede ortalama 1.7 megabaytlık veri üretileceğini tahmin etmektedirler [3]. Bu bilgiler ışığında verinin hem miktar hem çeşitlilik olarak artması “Büyük Veri” kavramını ortaya çıkarmıştır.

“Büyük Veri” olarak adlandırılan bu büyük veri kümeleriyle ilgili olarak IBM’e göre bir günde yaklaşık 2,5 kentilyon veri üretilmektedir. Üretilen bu verinin çok büyük kısmını ise (%90-%95) metin verileri oluşturmaktadır. Forbes verilerine göre 2 yıl içerisinde üretilen metin verileri 40 zettabayt boyutuna ulaşması öngörülmektedir [4]. Ayrıca Google her saniyede 40 binden fazla arama sonucunu işlemekte, bu durum yaklaşık olarak günde 3,5 milyar aramaya tekabül etmektedir [5]. Bu nedenle metin işleme çalışmaları son dönemde oldukça önemli hale gelmiş ve yapılan çalışmalar incelendiğinde oldukça popüler olmuştur. Metin işleme çalışmaları; metin sınıflandırma (text classification), duygu analizi (sentiment analysis), metin özetleme (text summarization), makine çevirisi (machine translation), bilgi çıkarımı (information retrieval), adlandırılmış varlık tanıma (named entity recognition) başta olmak üzere birçok farklı alanda yapılmaktadır [6]. Bu tez çalışmasında da yapılan deneysel çalışmalarda metin sınıflandırma işlemi uygulanmıştır.

Metin Sınıflandırması, doğal dil metinlerinin önceden belirlenmiş kategorilere göre sınıflandırılması işlemidir. Metin sınıflarının sayısı metnin boyutuna veya metne göre değişebilir. Doğal dil işlemede (Natural Language Processing-NLP), duygu analizi, konu etiketleme, spam tespiti ve amaç tespiti gibi geniş uygulamalara sahip temel görevlerden biridir. Metin sınıflandırma işleminin temelinde, kategorizasyon işlemi doğru kullanıldığında, metin mümkün olabilecek en doğru sınıfa atanır.

Bu tez çalışmasında büyük veri miktarındaki metin verilerinin analiz edilmesinde büyük veri teknolojilerinden ve derin öğrenme tekniklerinden faydalanılmıştır. Özellikle son yıllarda daha hızlı ve güçlü işlem yapabilen grafik işlem birimlerinin kullanılmasıyla birlikte hızlı işlem yapabilmenin maliyeti düşmüştür. Bu gelişmede derin sinir ağlarını oldukça popüler hale getirmiştir. Derin öğrenme ile başta görüntü işleme alanında olmak üzere uygulandığı birçok çalışma alanında başarılı sonuçlar elde edilmiştir. Son zamanlarda yapılan çalışmalar ile derin öğrenme doğal dil işleme alanında da önemli katkılar sağlamıştır. Özellikle büyük veriler üzerinde oldukça başarılı performans göstermiştir. Bu nedenle, bu tez çalışmasında da derin öğrenme yaklaşımı tercih edilmiştir.

Söz konusu kavramlar günümüz bilgisayar bilimleri alanında oldukça popüler olan ve son yıllardaki akademik çalışmalarda oldukça tercih edilen çalışma alanlarıdır. Bu tez çalışmasında, birbirinden farklı çalışma alanları gibi görülse de büyük veri, derin öğrenme ve metin işleme konuları birlikte işlenmiş problemin çözümünde bu üç popüler alan Şekil 1.1.'de görüldüğü gibi birbirine entegre olarak düşünülmüş ve bu alanların teknolojilerinden ve yöntemlerinden faydalanarak daha etkili çözümler geliştirilmeye çalışılmıştır.



Şekil 1.1. Çalışma konularının işlenme biçimi

## 1.1. Amaç ve Kapsam

Metin işleme çalışmaları, büyük veri kavramının ortaya çıkmasıyla ve üretilen bu büyük verinin çok büyük kısmının metinlerden oluşmasıyla birlikte son yıllarda en önemli çalışma alanlarından biri haline gelmiştir. Bu nedenle bu tez çalışmasında metin işleme alanında araştırmalar yapılması tercih edilmiştir. Literatür incelendiğinde ise yapılan metin işleme çalışmalarında Türkçe metin işleme çalışmalarının yeterli sayıda olmadığı görülmüştür. Bu nedenle Türkçe metin işleme çalışmasının yapılması hedeflenmiştir.

Derin öğrenme yaklaşımı ise özellikle büyük veri ile oldukça başarılı sonuçlar göstermiştir. Üzerinde çalışılacak metin verilerinin büyüklüğü göz önünde bulundurulduğunda

ise derin öğrenme modelleri ve büyük veri teknolojilerinin birlikte kullanılmasına karar verilmiştir. İşlemler bir büyük veri teknolojisi olan Apache Spark üzerinde geliştirilmesi düşünülmüştür. Bunun sebebi Apache Spark büyük veri teknolojileri içerisinde en hızlısıdır. En büyük rakibi olan Map Reduce'dan yaklaşık 100 kat daha hızlı olduğunu iddia etmektedir. Bu nedenle bu platformun kullanılması ve geliştirilen uygulamalar ile gerçek bir metin işleme görevi üzerinde test edilmesi hedeflenmiştir.

Metin işleme çalışmalarına bakıldığında Türkçe çalışmalar ve dolayısıyla Türkçe veri setleri sınırlı sayıdadır. Bu sebeple bu tez çalışmasında Türkçe dili tercih edilmiştir. Bunun için iki adet Türkçe veriseti (Dataset-1 ve Dataset-2) üretilmiştir. Dataset-1, Türkçe Wikipedia makaleleri kullanılarak elde edilen şimdiye kadar Türkçe için üretilmiş en büyük derlem (yaklaşık 80 gb) olması hedeflenmiştir. Bu derlem üzerinde yine şu ana kadarki çalışmalar içinde en büyük kelime vektörleri üretilmesi planlanmıştır. İlk etapta kelime temsil yöntemleri Word2Vec ve onun iki yöntemi olan CBOW (Continuous Bag of Words) ve SG (Skip-Gram) algoritmaları ile Glove yöntemleri analiz edilerek yakın anlamlı kelimelerin benzerlikleri incelenmesi yapılmıştır. Bu sayede en başarılı kelime temsil yönteminin belirlenmesi hedeflenmiştir.

Metin işleme çalışmalarında en kritik adımlardan biri önışlem aşamasıdır. Bu evre içinde etkisiz (stopwords) kelimelerin temizlenmesi modelin performansını önemli derecede etkilemektedir. Ancak Türkçe için geliştirilmiş bir kelime listesi bulunmamaktadır. Bu çalışma kapsamında TF-IDF yöntemi ile önışlem aşamasında etkisiz (stopwords) kelimeler listesi üretilmesi hedeflenmiştir.

Tezin temel amacı ise, derin öğrenme mimarileri olan CNN (Convolutional Neural Network), RNN (Recurrent Neural Network), LSTM (Long Short Term Memory), GRU (Gated Recurrent Unit) ve bu mimarilerin çeşitli varyasyonları kullanılarak oluşturulan sinir ağları ile Dataset-2 üzerinde Apache Spark ile metin sınıflandırma işlemleri yapılmasıdır. Aynı zamanda derin sinir ağlarının başarımları ve kelime temsil yöntemlerinin bu başarıma etkileri kıyaslanarak sonuçlarının analiz edilmesidir. Önerilen yöntemler ile Dataset-1 üzerinde eğitilen kelime vektörleri Dataset-2 üzerinde sınıflandırma aşamasında transfer öğrenme ile kullanılmış ve doğruluk değerleri geliştirilmesi hedeflenmiştir. Önerilen ikinci yöntem ile Dataset-2 üzerindeki yazım yanlışları tespit edilmiş ve bu hatalar giderilmeye çalışılmıştır. Türkçe için uygulaması olmayan bu yöntem ile de doğruluk değerlerinin iyileştirilmesi amaçlanmıştır.

## 1.2. Literatür Çalışması

Bu bölümde, kelime temsil yöntemleri ve sinir ağı temelli yaklaşımlar ile ilgili literatürde yapılan çalışmalar kapsamlı bir şekilde araştırılmıştır. Yapılan incelemeler Türkçe dilinde yapılan çalışmalar ve Türkçe dışındaki dillerde yapılmış çalışmalar olmak üzere iki alt başlıkta incelenmiştir.

### 1.2.1. Türkçe İçin Yapılmış Çalışmalar

Amasyalı ve arkadaşları, yaptıkları duygu analizi çalışmasında kelime, anlam ve karakter bazlı temsil yöntemlerini analiz etmişlerdir. Çalışmaları için 3 sınıf ve yaklaşık 17 bin tweetten oluşan bir veri seti üretmişlerdir. Hem kelime temsil yöntemlerinde hem de sınıflandırıcılarda geleneksel ve yenilikçi yöntemleri birlikte kullanmışlardır. Metin temsili için BoW (Bag of Words), n-gram ve fasttext yöntemlerini kullanmışlardır. Sınıflandırıcı olarak ise klasik yöntemlerden DVM (Destek Vektör Makinesi) ve RO (Rasgele Orman) kullanırken, CNN (Convolutional Neural Networks) ve LSTM (Long Short Term Memory) gibi derin öğrenme yaklaşımlarını tercih ederek sonuçları kıyaslamışlardır. Çalışmanın sonucunda yenilikçi yaklaşımların daha başarılı olduklarını, gösterim metotlarında ise karakter bazlı gösterimin daha iyi sonuçlar verdiğini tespit etmişlerdir [7].

Yıldırım ve Yıldız, kelime temsil yöntemi olarak BoW, Word2Vec, Doc2Vec ve Glove yöntemlerini uyguladıkları çalışmalarında 7 sınıftan ve 4900 dokümandan oluşan bir veri seti üzerinde Türkçe metin sınıflandırma uygulaması yapmışlardır. Çalışmalarında sınıflandırıcı olarak LR (Lojistik Regresyon), DVM, KNN (K- Nearest Neighbour) gibi geleneksel yöntemler ve CNN, RNN (Recurrent Neural Network), LSTM ve GRU gibi yenilikçi derin öğrenme yöntemlerini kullanmışlardır. Çalışmalarında geleneksel tabanlı bir temsil yöntemi olan BoW yönteminin, daha başarılı bir performans gösterdiği gözlemlenmiştir [8].

Ay Karakuş ve arkadaşları, Word2Vec temsil yöntemi ile derin sinir ağları üzerinde Türkçe duygu analizi çalışması yapmışlardır. Bu çalışma için film yorumlarından oluşan yaklaşık 13 milyon kelime içeren Türkçe bir veri seti ürettir. CNN, LSTM, BLSTM (Bidirectional Long Short Term Memory), ve CNN-LSTM gibi çeşitli derin sinir ağı mimarilerini kullanarak 40 bin yorumu sınıflandırmaya çalışmışlardır. Farklı mimari ve parametrelerin denenerek bu işlemlerin başarıma etkisinin analiz edilmeye çalışıldığı bu çalışmada 4000 test verisi üzerinde CNN-LSTM mimarisi %98.07 doğruluk skoru ile diğer yöntemlerden daha başarılı sonuç göstermiştir [9].

Şahin, 22.729 kayıt ve 7 sınıftan oluşan bir Türkçe veri seti üzerinde farklı kelime temsil yöntemleri ile bir çalışma yapmıştır. BoW ve Word2Vec kelime temsil yöntemlerini kullandığı çalışmada bu yöntemlerin sınıflandırma başarımına etkisini araştırmıştır. 12 milyon cümle ve 500 milyon kelimeden oluşan bir Türkçe veri seti üzerinde kelime vektörlerini eğitmiştir. Sınıflandırıcı olarak ise DVM kullanmıştır. Ayrıca Word2Vec yönteminin farklı mimarilerinin de kullanıldığı çalışmada negatif örnekleme (negative sampling) uygulanmış Skip-Gram algoritması %91 doğruluk skoru elde etmiş ve BoW yönteminden daha başarılı olmuştur [10].

Ayata ve arkadaşları, çalışmalarında 5 bin kayıt ve 4 sınıftan oluşan Türkçe bir veri seti üretmişler ve bu veri seti üzerinde DVM ve RO sınıflandırıcılarını kullanarak Türkçe sınıflandırma çalışması yapmışlardır. Kelime temsil yöntemi olarak Word2Vec kullanarak kelime vektörlerini eğitmişlerdir [11].

Polat ve Körpe, yaptıkları çalışmalarında TBMM (Türkiye Büyük Millet Meclisi) tutanaklarından oluşturdukları derlem üzerinde Word2Vec ve Glove kelime temsil yöntemlerini kullanarak deneyler yapmışlardır. Yakın anlamlı kelimeleri, kelime temsil yöntemleri ile tespit etmeye çalışmışlardır. Çalışma sonunda, oluşturdukları derlem üzerinde Word2Vec yönteminin benzer kelimeleri tespit etmede Glove yöntemine göre daha başarılı olduğu görülmüştür [12].

Acı ve Çırak, CNN derin sinir ağı mimarisi ve Word2Vec yöntemini kullanarak TTC-3600 veri seti üzerinde Türkçe haber metinlerini sınıflandırma çalışması yapmışlardır. Önerdikleri yöntem ile önceki çalışmalarda kullanılan yöntemler ve başarımları kıyasladıkları çalışmalarında CNN ve Word2Vec ile geliştirdikleri yöntem ile daha başarılı sonuçlar elde etmişlerdir. %93.3 doğruluk skoru ile Word2Vec yönteminin sınıflandırma işleminde kullanılmasının önemli bir etken olduğu ortaya çıkmıştır [13].

Erşahin ve arkadaşları Türkçe duygu analizi için duygu sözlüğü tabanlı yeni bir yöntem önermişlerdir. NB (Naive Bayes), DVM ve KA (Karar Ağacı) algoritmalarını ise sınıflandırıcı olarak kullanmışlardır. Önerilen yöntem, sınıflandırıcılar üzerinde test edilerek doğruluk oranının %7 oranında iyileştirildiği görülmüştür [14].

Amasyalı ve Diri çalışmalarında Türkçe metin sınıflandırma için bir n-gram yaklaşımı önermişlerdir. Çalışmalarında sınıflandırıcı olarak NB, DVM, RO ve KA algoritmalarını kullanmışlardır. Deneysel çalışmalar sonucunda bigram yaklaşımının trigram yöntemine göre daha başarılı sonuçlar elde ettiği görülmüştür. Farklı iki görev üzerinde test ettikleri yöntemlerinde en başarılı sınıflandırıcılar NB (%83.3) ve DVM (%93.6) olmuştur [15].

Kılınç ve arkadaşları TTC-3600 isimli çeşitli haber sitelerinden topladıkları veriler ile 6 kategoriden oluşan bir Türkçe veri seti oluşturmuşlardır. Ayrıca veri seti üzerinde farklı makine öğrenmesi algoritmaları ile metin sınıflandırma işlemi yapmışlardır [16].

### 1.2.2. Türkçe Dışındaki Dillerde Yapılmış Çalışmalar

CNN, çok boyutlu girdiler için ve özellikle iki boyutlu görsel veriler için önerilmiş bir derin öğrenme yöntemidir [17]. Ancak doğal dil işleme alanındaki çalışmalarda da başarılı sonuçlar elde edilmiştir.

Collobert ve Weston ilk kez yaptıkları çalışma ile başlangıçta görüntü işleme çalışmaları için tasarlanmış CNN mimarilerinin birçok doğal dil işleme probleminde kullanılabileceğini ifade etmişlerdir [18].

Kim ve arkadaşları tarafından yapılan çalışmada, çeşitli veri setleri üzerinde metin sınıflandırma işlemi yapılmıştır. Tek katmanlı CNN mimarisinin kullanıldığı çalışmada başarılı sonuçlar elde edilmiştir [19].

Zhang ve arkadaşlarının, karakter seviyesinde CNN mimarisi kullanarak metin sınıflandırma yaptığı çalışmada ise bu yöntemin güçlü bir yöntem olduğu ve metin sınıflandırma problemlerinin çözümünde kullanılabileceği anlatılmıştır [20].

Lai tarafından yapılan bir çalışmada ise yinelemeli CNN mimarisi ile metin sınıflandırma yapılmıştır. Çalışmada önerilen model de bağlamsal bilgi için RNN mimarisi, metnin temsil edilmesi için ise CNN mimarisi kullanılmıştır [21].

RNN, veri akışlarını (stream of data) analiz edebilen gizli katmanlara sahip sinir ağlarıdır ve çıktının bir önceki hesaplamalara bağlı olduğu problemlerin çözümü için çok uygundur [22,23]. RNN, bu nedenle özellikle doğal dil işlemenin değişik problemlerinin çözümünde oldukça başarılı olmuştur.

RNN’lerde öğrenme sırasında ortaya çıkan bazı problemler nedeniyle, bu çalışmada da incelenen uzun kısa-dönem bellek (Long Short-Term Memory - LSTM) [24] ve GRU (Gated Recurrent Units) [25] gibi farklı sürümleri literatüre girmiştir.

Lee ve Dernoncourt tarafından yapılan çalışmada ise bir RNN mimarisi türü olan LSTM modeli, CNN mimarisi ile birlikte kullanılmış ve çeşitli veri setleri üzerinde başarılı sonuçlar elde edilmiştir [26].

Kalchbrenner ve arkadaşları, cümlelerin anlamsal modellemesi için değişken uzunlukta girdi kullanarak dinamik bir CNN mimarisi (DCNN) tanımlanmıştır. Tanımlanan ağ yapısı, dinamik k-max havuzlama (pooling) uygulanan dinamik havuzlama (pooling) katmanları, doğrusal olmayan özellik işlevi, çoklu özellik haritaları ve katlama katmanından oluşturulmuştur. Bu tanımlanmış ağ yapısını test etmek için, İngilizce twitter içerikleri üzerinde deneyler yapılmıştır. Sonuçlar incelendiğinde önerilen mimari ile veri seti üzerindeki en yüksek başarı elde edilmiştir [27].

Severyn ve Moschitti, İngilizce bir derlem olan Semeval-15 derlem (corpus) üzerinde mesaj ve ifade düzeyinde duygu analizini uygulamak için kelime temsil yöntemleri ve CNN kullanmışlardır. Deneysel çalışmalar sırasında; rastgele ağırlıklar ile 50 milyon Tweet üzerinde eğitilmiş Word2Vec vektörleri ağ parametreleri için kullanılmıştır. Deneylerin sonucunda, ifade düzeyinde duygu analizi için önerilen yöntem en iyi yöntem olarak bulunmuş ve cümle düzeyinde duygu analizinin ise, duygu analizi probleminin çözümünde ikinci en iyi yöntem olduğu görülmüştür [28].

Hassan ve Mahmood, CNN ve LSTM mimarilerinin birlikte kullanıldığı ConvLstm adında yapay bir sinir ağı önermişlerdir. Önerilen bu mimaride, CNN girdi verilerinden özellikleri çıkarmak için kullanılmış LSTM ise önemli bilgileri hatırlamak ve uzun vadeli bağımlılıkları yakalamak için bir araya getirme katmanı yerine kullanılır. Çalışmada; İngilizce veri setleri olan IMDB ve Stanford Sentiment Treebank'ta önceden eğitilmiş Word2Vec vektörleri kullanılmıştır. Elde edilen sonuçların, bu veri setlerini kullanan önceki çalışmalara göre daha başarılı olduğu bulunmuştur. Deneyler sonucunda; daha az parametre ile eğitilmiş olan bu ağ mimarisi, diğer modellere göre daha yüksek başarı elde ederek diğer yöntemlere alternatif olarak kabul edilmiştir [29].

Yang ve Xia, metinleri Word2Vec yöntemini kullanarak duygularına göre sınıflandırmak için son katmanı doğrusal bir sınıflandırıcı olan bir CNN mimarisi kullanmışlardır. Çince otel değerlendirmelerinden oluşan 4 farklı veri seti (3 dengeli, 1 dengesiz sınıf dağılımı) üzerinde Çince dili için sınıflandırma çalışması yapmışlardır. Metinleri sınıflandırmak için, bir CNN mimarisi ile birlikte DVM ve NB sınıflandırıcılarını kullanmışlardır. Deneylerin sonunda, dengesiz veri seti de dâhil olmak üzere tüm veri setleri üzerinde NB ile en kötü sonuçlar elde edilirken, en iyi sonuçların CNN mimarisi ile sağlandığı gözlemlenmiştir [30].

Huang ve arkadaşları, çalışmalarında, Çin mikro blog verileri üzerinde Word2Vec kullanarak kelime vektörlerini eğitmişler ve sınıflandırıcı olarak iki katmanlı LSTM, CNN, LSTM ve CNN-LSTM mimarilerini kullanmışlardır. Sonuçlara göre, önerdikleri tek katmandan oluşan CNN ve çift katmandan oluşan LSTM hibrit mimarisi ile %87.2 skoru ile en yüksek doğruluk değerini elde etmişlerdir [31].

Vateekul ve Koomsubha, Tayland dili için ilk metin sınıflandırması çalışmasını yapmışlardır. Topladıkları Tai dilinde yazılmış tweetler üzerinde Word2Vec ile kelime vektörlerini eğitmiş ve bu vektörleri DCNN ve LSTM derin öğrenme modelleri ile birlikte kullanmışlardır. Çalışmada bu iki derin öğrenme mimarisine ek olarak DVM, NB, ME gibi sınıflandırıcılar ile de deneyler yapılmıştır. Sonuçlar incelendiğinde iki derin öğrenme mimarisi

diğer sınıflandırıcılardan daha başarılı olduğu görülmüştür. DCNN mimarisi ile en yüksek skor olan %75.35 değeri elde edilirken bu değeri LSTM mimarisinde %75.30 olarak tespit edilmiştir [32].

Vo ve arkadaşları, Vietnam dilinde içeriklerden oluşan 2 farklı veri seti üzerinde çalışmıştır. Bu veri setleri üzerinde sınıflandırma için, CNN, LSTM, CNN-LSTM derin öğrenme mimarileri ve DVM sınıflandırıcısını kullanmıştır. DVM ile yapılan uygulamada geleneksel kelime temsil yöntemlerinden BoW yöntemi kullanılırken, derin öğrenme mimarileri ile birlikte sinir ağı temelli kelime temsil yöntemleri tercih edilmiştir. Çalışma sonunda en yüksek başarımlarını CNN ve LSTM mimarilerinin birlikte kullanıldığı CNN-LSTM modeli elde ettiği görülmüştür [33].

Zhang ve Xu çalışmalarında, Word2Vec ve DVM yöntemlerini kullanarak Çince metinler üzerinde sınıflandırma uygulaması yapmışlardır. Bir giyim firması ile alakalı olarak toplanan yorumlara önce Word2Vec yöntemi uygulanmış ve kelime vektörleri üretilmiştir. Daha sonra bu vektörler DVM ile sınıflandırılmıştır. Önerilen bu yöntem ile %90 doğruluk değeri elde edilmiştir [34].

Hughes ve arkadaşları, CNN mimarisini kullanarak tıbbi metin sınıflandırma uygulaması yapmışlardır. Bu çalışmayı 26 kategoriden ve 4000 veriden oluşan bir veri seti üzerinde yapmışlardır. Çalışmada Word2Vec kelime temsil yöntemi ile vektörler eğitilerek sınıflandırma işleminin yapılması ile doğal dil işleme alanında kullanılan geleneksel yöntemlere kıyasla yaklaşık %15 oranında doğruluk değerinin geliştirildiği ifade edilmiştir [35].

Sohangir ve arkadaşları, kullanıcıların borsa tahmini gibi finansal görüşlerini paylaştıkları platformlardan topladıkları verilerini kullanarak LSTM ve CNN derin öğrenme modelleri ile analiz etmişlerdir. Çalışmalarında %90 doğruluk oranı ile borsa tahmini yapan bir model geliştirmişlerdir [36].

Shoro ve Soomro, çalışmalarında, büyük veri teknolojilerinden Apache Spark ile twitter üzerinden akan verilerin gerçek zamanlı analizini yapmışlardır [37].

### 1.3. Tezde Yapılan Çalışmalar ve Literatüre Katkısı

- Türkçe için literatürde olmayan iki Türkçe veriseti üretildi. İlk veriseti (Dataset-1) Türkçe Wikipedia makalelerinden üretilmiştir ve şimdiye kadar yapılan Türkçe çalışmalar içerisindeki etiketsiz en büyük derlemidir.
- Dataset-1 üzerinde kelime temsil (word embedding) metotları ile şimdiye kadar yapılan Türkçe çalışmalar içerisindeki en geniş kelime vektörleri eğitildi.
- Çağrı merkezleri, müşteri yorumları ve mesajları ile kullanıcıların bir moda firması hakkındaki görüşleri toplanarak çok sınıflı ikinci bir veriseti (Dataset-2) üretildi. Bu veriseti üzerinde kelime vektörleri ve önerilen yöntemler test edildi.
- Öncelikle üretilen veri setleri üzerinde önışlem çalışmaları yapıldı. Önışlem adımında metinlerdeki gürültüyü ortadan kaldırmanın en önemli yöntemlerinden biri olan etkisiz (stopwords) kelimelerin tespit edilerek bu kelimelerin derlemden kaldırılması hedeflendi. Bunun için TF-IDF ile etkisiz (stopwords) kelimeler üretildi ve bu liste ile yaklaşık 1 milyar kelime derlemden çıkarıldı.
- Neredeyse tüm doğal dillerin NLP kütüphanelerinde entegre halde bir etkisiz kelime (stopwords) aracı bulunurken, bu araç bu tez çalışması kapsamında ilk kez Türkçe için üretildi. Ayrıca bu listenin açık kaynak topluluğu ile paylaşılarak NLTK (Natural Language Toolkit), Spacy ve Gensim gibi büyük NLP kütüphanelerine entegre edilmesi planlanmaktadır.
- Üretilen kelime vektörleri, Dataset-2 üzerinde derin sinir ağı mimarileri ile birlikte kullanıldı ve metin sınıflandırma işlemi uygulandı. Bu aşamada, derin sinir ağları üzerinde Türkçe dilinin doğasını en iyi şekilde temsil edecek özel ayarlar ile eğitildi.
- Çalışmada hemen hemen tüm derin sinir ağı mimarileri kullanıldı. Her bir ağın başarımı grafiklerle görselleştirilerek analiz edildi ve elde edilen sonuçlar kıyaslanarak detaylandırıldı. Ayrıca doğal dil işleme çalışmalarında en çok tercih edilen kelime temsil (word embedding) metotlarının da başarımları incelendi ve bu yöntemlerin modellerin başarımları üzerindeki etkileri incelendi.
- Dataset-1 üzerinde eğitilen en geniş kelime vektörleri transfer öğrenme ile Dataset-2 üzerinde kullanıldı. Önceden eğitilmiş bu kelime vektörleri ile bir önceki adımda bahsedilen başarımlar değerlendirildi. Hemen hemen tüm modeller üzerinde sağlanan yaklaşık olarak %5 - %7 arasındaki bu iyileştirme değerleri tablolastırılarak sunuldu ve sonuçlar kıyaslanarak analiz edildi.

- Dataset-2 doğal ortamlardan toplanmış gerçek verilerden oluştuğu için bu verilerin büyük kısmının yanlış ve hatalı yazılmış kelimelerden oluşan kirli veriler olduğu tespit edildi ve bu durumun modelin performansını önemli derecede olumsuz olarak etkilediği görüldü.
- Literatürde hali hazırda bir yazım denetleyicisi olmadığı için veri kalitesini artırmak amacıyla, Türkçe için literatürde olmayan yeni bir yaklaşım önerildi ve bu yöntem ile yanlış yazılmış kelimeler tespit edildi. Bu yöntemle sözlük yöntemi adı verildi.
- Sözlük yöntemi ile tespit edilen hatalı kelimelerin yerine, derin sinir ağı modellerinden LSTM kullanılarak doğru kelimeler tahmin edilmeye çalışıldı ve tahmin edilen yeni kelimeler yanlış yazılmış kelimeler ile değiştirildi.
- Yanlış yazılmış kelimeler sözlük yöntemi ile düzeltildikten sonra yeniden sınıflandırma işlemi uygulandı. Yaklaşık 55 bin yanlış yazılmış kelime tespit edildi ve tahmin edilen doğru kelimeler ile değiştirildi. Doğruluk değeri, % 77.12 den % 85.80'e yükseldi ve böylece % 8.68 oranında bir iyileştirme sağlandı.

#### 1.4. Tezin Yapısı

Doktora tezi olarak hazırlanan bu çalışma altı bölümden oluşmuştur. Bölümlerin özeti aşağıdaki gibidir. Tez çalışması büyük veri, metin işleme ve derin öğrenme temellerine dayanmaktadır. Bu nedenle alt bölümlerde bu kavramlar detaylı olarak ele alınmıştır.

Birinci bölümde, teze giriş yapılmış, problem tanımlanmış, geliştirilecek çözümün amacı, etkileri, kapsamı, yapılan çalışmalardan ve literatüre katkısından bahsedilmiştir.

İkinci bölümde, büyük veri kavramı açıklanmıştır. Öncelikle büyük veri kavramının ne olduğu, ortaya çıkış nedeni ve büyük veri bileşenleri ele alınmıştır. Neden bu tez çalışmasında büyük veri teknolojilerine ihtiyaç duyulduğundan ve bu çalışmada tercih edildiğinden bahsedilmiştir.

Üçüncü bölümde, metin işleme başlığı incelenmiştir. Doğal dil işlemenin temel prensipleri açıklanmış, metin işlemeden önce önışlem adımları incelenmiş ve bu çalışmada hangi önışlem adımlarının uygulandığına değinilmiştir. Tez çalışmasının en önemli noktalarından biri olan kelime temsili konusuna detaylı olarak yer verilmiştir. Kelime temsillerinin arka planındaki matematiksel detaylara yer verilmiş ve çalışma prensipleri açıklanmıştır.

Dördüncü bölümde, derin öğrenme yaklaşımı incelenmiştir. Bu tez çalışmasında kullanılan tüm derin öğrenme mimarileri ve matematiksel alt yapıları detaylı olarak açıklanmıştır. Ayrıca metin verileri için derin öğrenme konusu bu bölümde incelenmiştir.

Beşinci bölümde, tez çalışması kapsamında geliştirilen üç uygulama ve bu uygulamalar sonucunda elde edilen materyaller anlatılmıştır. Geliştirilen Apache Spark ile büyük veri uygulaması, TF-IDF yöntemi ile Türkçe etkisiz kelimeler (stopwords) uygulaması ve Türkçe kelime vektörlerinin eğitildiği uygulamalara yer verilmiştir. Ayrıca üretilen veri setlerinden ve özelliklerinden, üretilen kelime vektörlerinden ve yapılarından bahsedilmiştir. Daha sonra geliştirilen ve önerilen yöntemle sunulmuştur. TF-IDF yöntemi ile literatüre kazandırılan önışlem uygulaması ve sonuçları açıklanmıştır. Önerilen bir diğer yöntem olan, önceden eğitilmiş kelime vektörlerinin transfer öğrenme ile kullanıldığı yöntem açıklanmış ve sonuçları görselleştirilerek sağlanan iyileştirme detaylandırılmıştır. Tez çalışmasında önerilen son yöntem olarak sözlük metodu ve çalışma prensibi anlatılmıştır.

Altıncı bölümde, doğruluk değerinin artırılması amacıyla önerilmiş iki özgün yöntemden bahsedilmiştir. Önceden eğitilmiş kelime vektörleri ve sözlük yöntemi adı verilen iki yöntem bu bölümde detaylı şekilde anlatılmış elde edilen deneysel sonuçlar incelenmiştir.

Yedinci bölümde ise tez çalışması ile elde edilen sonuçlar ve öneriler yer almaktadır.

## 2. BÜYÜK VERİ

Günümüzde depolama kapasitesi ne kadar büyük olursa olsun bu büyüklüğü doldurmaya yetecek büyüklükte veri üretilmektedir. Bilişim teknolojilerinde yaşanan gelişmeler ile birlikte internet çağının gelmesi, buna paralel olarak mobil cihazların günlük yaşamda yaygın olarak kullanılması, bu gelişmeyi ise sosyal medya devriminin izlemesi ile veri miktarı her geçen gün artmaktadır. Üretilen bu verinin hem miktar hem de çeşitlilik olarak artması ile birlikte günümüzde *büyük veri* kavramı ortaya çıkmıştır [38].

Büyük veri, çeşitli ortamlardan toplanan yapısal olan veya yapısal olmayan tüm verinin anlamlı ve işlenebilir hale getirilmiş biçimidir. Son yıllarda bilişim teknolojilerinde yaşanan gelişmeler ve internetin yaygın kullanımı ile birlikte üretilen verinin türü de çeşitlik kazanmıştır. Ses, görüntü, metin tabanlı her türlü yapısal olan ve yapısal olmayan verilerin miktarında ciddi oranda artış meydana gelmiştir [39].

### 2.1. Büyük Veri Bileşenleri

Devasa miktardaki verinin anlamlandırılabilmesi, işlenebilmesi literatürdeki tanımlanmasıyla bir “bilgi çöplüğü” olmaktan çıkarılması için büyük verinin bileşenlerinin ayrıştırılması gerekmektedir. Bunun için bu bileşenler Volume, Velocity ve Variety olarak tanımlanmış ve kelimelerin baş harfleri dikkate alınarak 3V olarak modellenmiştir. Bileşenler aynı mantıkla Türkçeleştirilirse sırasıyla Çokluk, Çabukluk ve Çeşitlilik olarak 3Ç modeliyle tanımlanabilir [40].

Volume (Çokluk) : Bu bileşen saklanan verinin çok büyük miktarlarda olabileceğini belirtmektedir. Ayrıca veri büyüklüğüyle alakalı olarak ölçü birimleri Çizelge 2.1’de gösterilmiştir.

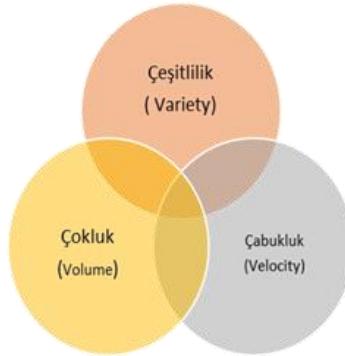
Çizelge 2.1 Ölçü Birimleri

|                 |              |
|-----------------|--------------|
| 8 Bit           | 1 Bayt       |
| 1024 Bayt       | 1 Kilobayt   |
| 1024 Kilobayt   | 1 Megabayt   |
| 1024 Megabayt   | 1 Gigabayt   |
| 1024 Gigabayt   | 1 Terabayt   |
| 1024 Terabayt   | 1 Petabayt   |
| 1024 Petabayt   | 1 Exabayt    |
| 1024 Exabayt    | 1 Zettabayt  |
| 1024 Zettabayt  | 1 Yottabayt  |
| 1024 Yottabayt  | 1 Brontobayt |
| 1024 Brontobayt | 1 Geopbayt   |

Çizelge 2.1 analiz edildiğinde 1 zettabayt 1 trilyon gigabayta eşittir. Diğer bir deyişle 1,2 zettabayt boyutunda bir veriyi saklamak için 16 gigabayt depolama hafızasına sahip 75 milyar adet sayıda tablet bilgisayar gerekmektedir. 75 milyar adet tablet bilgisayar ise yaklaşık 41 adet futbol sahası alanı büyüklüğündedir [41].

Velocity (Çabukluk) : Büyük veride verinin üretilme hızı çok büyüktür bu büyüklüğü tanımlar. Daha doğru bir ifadeyle veri çok çabuk artmaktadır.

Variety (Çeşitlilik) : Bu bileşen incelendiğinde ise akla ilk olarak “veri tipi” kavramı gelmelidir. Çeşitlilik tanımlaması ise veri tiplerindeki farklılıklardan dolayı yapılmıştır.



Şekil 2.1 Büyük veri bileşenleri (3Ç– 3V)

Literatürde Büyük veri kavramına daha sonra eklenmiş 2V daha bulunmaktadır. Bunlar Verification ve Value kavramlarıdır.

Verification (Doğrulama): Verinin gürültüden arındırılmış doğruluk bileşeni olarak kabul edilmektedir. Olması gerektiği güvenlik seviyesinde izlenmesi doğru kişiler tarafından görünebilir veya gizli kalmasıdır.

*Value (Değer):* Bu kavram ise aslında bir bileşen değil büyük verinin şimdiye kadar anlatılan 4 temel bileşeni sonunda ortaya çıkan anlamlandırılmış değerli bilgidir [42].

## 2.2. Büyük Veri Teknolojileri

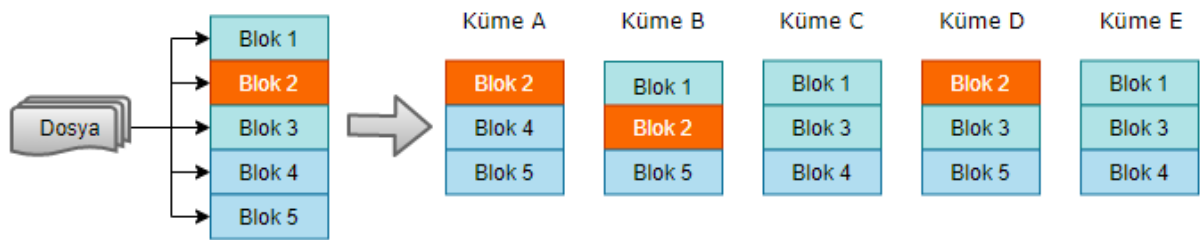
### 2.2.1. Hadoop

Hadoop, gigabayttan petabaytlara kadar geniş veri kümelerini verimli bir şekilde depolamak ve işlemek için kullanılan açık kaynaklı bir yazılımdır. Verileri depolamak ve işlemek için büyük bir bilgisayar kullanmak yerine, Hadoop birden çok bilgisayarın büyük veri kümelerini paralel olarak daha hızlı analiz etmesini sağlamaktadır. Hadoop teknolojisi, Java ile geliştirilmiş Map-Reduce tekniği ve HDFS (Hadoop Distributed File System) adı verilen dağıtık dosya sisteminden oluşan bir kütüphanedir [43].

#### 2.2.1.1. HDFS (Hadoop Distributed File System)

Hadoop uygulamalarının depolama birimlerini yönetmektedir. Petabayt boyutundaki verilerin cluster denilen sunucu kümeleri üzerinde saklayabilmek için tasarlanmış dağıtık bir dosya sistemidir. Böylece çok büyük boyuttaki dosyalar bu dosya sisteminde depolanabilir ve bu verilere birçok kaynak üzerinden aynı anda ve oldukça hızlı erişim sağlanabilir.

HDFS sisteminde dosyalar bloklar şeklinde depolanmaktadır. Olumsuz bir durumla karşılaşıldığında veri kaybının yaşanmaması için her bir blok çoklanarak depolanır. Bu durumda HDFS mimarisinin bir kez yaz, birçok kez oku (write-once, read-many-times) yaklaşımıdır [44].

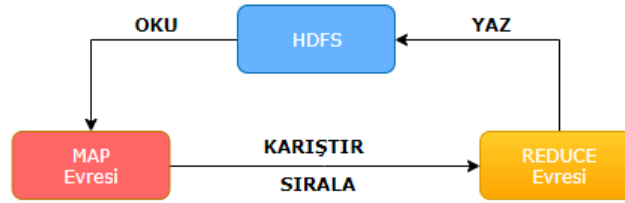


Şekil 2.2 HDFS yapısı

Şekil 2.2’de görüldüğü gibi HDFS sistemine girdi olarak dosya önce bloklara bölünmüş daha sonra her blok farklı bilgisayar kümelerinde depolanarak çoklanmıştır.

### 2.2.1.2. Map Reduce

Map Reduce, dağıtık dosya sistemleri üzerinde (HDFS) veri işleme amacıyla Google tarafından 2004 yılında geliştirilmiş bir yöntemdir. İsmi aldığı Map ve Reduce fonksiyonlarından oluşmaktadır. İstenilen verilerin filtrelenmesini sağlayan Map fonksiyonu ve filtrelenmiş bu verileri işleyerek sonuç üretmeye yarayan Reduce fonksiyonlarından oluşan program geliştirildikten sonra Hadoop üzerinde çalıştırılmaktadır. Bu durum Şekil 2.3.'te görülmektedir. Map Reduce yapısı Java, C++, Python, Perl, Ruby, C gibi birçok dille geliştirilebilmektedir [45].



Şekil 2.3. Map-Reduce çalışma şekli

### 2.2.2. Apache Spark Projesi

Apache Spark, iteratif makine öğrenimi görevleri için çok uygun olan büyük ölçekli veri işleme için popüler bir açık kaynak platformudur. Modern veri kümeleri boyut ve karmaşıklık açısından hızla büyümekte ve bu veri zenginliğini kullanmak için istatistiksel yöntemler kullanılarak çözümler geliştirmek için acil bir gereksinim ortaya çıkmıştır. MapReduce teknolojisini genelleştiren bazı 'yeni nesil' veri akışı motorları, büyük ölçekli veri işleme için geliştirilmiş ve bu motorlarda makine öğrenimi işlevselliği oluşturmak oldukça popüler bir çalışma konusu olmuştur [46].

Apache Spark, 2009 yılında Matei Zaharia tarafından Berkley Üniversitesi laboratuvarlarında geliştirilmeye başlanan açık kaynak kodlu bir büyük veri teknolojisidir. Veriyi saklayan değil sadece işleyen bir teknoloji olduğu için Hadoop projesinde HDFS 'ye değil Map Reduce teknolojisine alternatif olduğu söylenilebilir. Spark, Map Reduce benzeri bir programlama modeline sahiptir, ancak "Resilient Distributed Datasets" veya kısa hali ile RDD denilen bir veri paylaşım soyutlamasıyla genişletir. Bu basit eklentiye (RDD) kullanarak Spark, SQL, streaming, makine öğrenimi ve grafik işleme dâhil olmak üzere daha önce ayrı bileşenlere ihtiyaç duyan çok çeşitli görevlerin üstesinden gelebilir. Paralel veri işleme yaygınlaştıkça, işleme fonksiyonlarının birleştirilebilirliği hem kullanılabilirlik hem de performans açısından en önemli konulardan biri olacaktır.

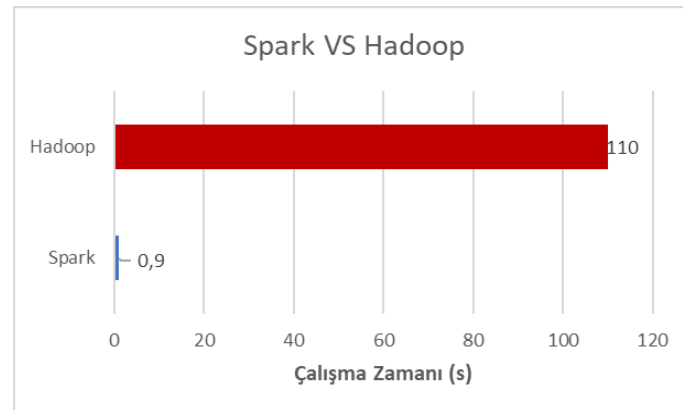
Büyük veri kümeleri üzerinde paralel olarak işlem yapmayı sağlayan Scala programlama dili ile geliştirilmiş bir kütüphanedir [47].

Apache Spark mimari olarak ise Efendi (Master) – Köle (Slave) prensibi ile verileri işlemektedir. Master olarak tanımlanan bilgisayar görevleri Slave bilgisayarlara iletirken tamamlanan görevler ise tekrar Master bilgisayarlara ileterek veri işleme işi sonuçlandırılır [48]. Spark core Apache Spark temelidir. Büyük ölçekli veri kümelerini, RDD API'sini işlemek için basit bir programlama arayüzü sağlar. Spark core, Scala'da uygulanır, ancak Scala, Java, Python ve R'deki API'lerle birlikte gelir. Bu API'ler, üst düzey kütüphanelerde veri analizi algoritmaları için gerekli olan birçok işlemi (yani veri dönüşümleri ve eylemleri) destekler. Buna ek olarak, Spark core, bellek yönetimi, iş planlaması, veri karıştırma ve hata kurtarma dâhil olmak üzere bellek içi küme hesaplaması için ana işlevler sunar. Bu işlevlerle, bir Spark uygulaması, bir bilgi işlem kümesinin CPU, bellek ve depolama kaynakları kullanılarak geliştirilebilir.

Apache Spark'ın hız bakımından diğer teknolojilerden avantajlı olmasının en önemli sebeplerinden biri de verileri diskten bir kez okuyarak RAM üzerinde depolama mimarisi ile geliştirilmiş olmasıdır [49].

Apache Spark'ın Avantajları:

- Kullanımı kolaydır. Map Reduce ile geliştirilen projeler Apache Spark ile daha az eforla geliştirilebilir.
- Map Reduce 'a göre daha hızlıdır (Şekil 2.4.).



Şekil 2.4. Spark ve Hadoop karşılaştırılması

- Python, Scala ve R, Java gibi farklı programlama dilleri ile geliştirilebilir.
- Ölçeklenebilirdir. Böylece Spark birden fazla sistem üzerinde çalıştırılırken herhangi bir sebeple bilgisayar sayısı rahatlıkla artırılabilir.

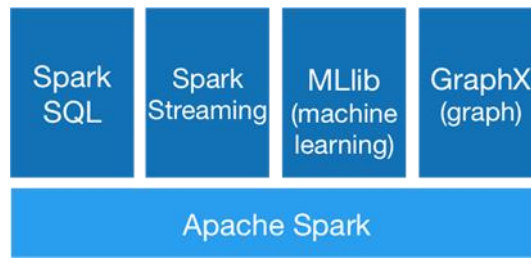
Apache Spark teknolojisi bileşenleri Şekil 2.5.'te görüldüğü gibidir.

Spark SQL, DataFrame: En yaygın veri işleme paradigmalarından biri ilişkisel sorgulardır. Spark SQL2 ve selefi Shark, analitik veritabanlarına benzer teknikleri kullanarak Spark üzerinde bu tür sorguları uygular ve bu yapılarla büyük veriler üzerinde SQL tabanlı analizler yapılabilir.

Spark Streaming: “ayrık akışlar” adı verilen bir model kullanarak bir akış işleme modeli uygular. Spark üzerinde veri akışının implementasyonu için, giriş verileri (inputs) küçük parçalara bölünür (her 200 milisaniyede gibi) ve yeni sonuçlar üretmek için rdd'lerin içinde saklanan durumlarla düzenli olarak birleştirilir. Bu bileşen ile akan veriler anlık olarak analiz edilebilir, gerçek zamanlı işlenebilir [50].

MLlib, Spark: teknolojinin makine öğrenmesi kütüphanesidir. MLlib, geniş ölçekli bir öğrenme ortamı için etkin işlevsellik sağlar ve temelde yatan istatistiksel, optimizasyon ve doğrusal cebir ilkelerini içerir. Spark ile birlikte gelen MLlib, birkaç dili destekler ve uçtan uca makine öğrenme boru hatları (pipelines) geliştirilmesini kolaylaştırmak için Spark'ın zengin ekosistemini güçlendiren üst düzey bir API sağlar. MLlib, 140'ın üzerinde katkıda bulunan açık kaynak topluluğu nedeniyle hızlı bir büyüme yaşadı ve daha fazla büyümeyi desteklemek ve kullanıcıların hızlı bir şekilde hızlanmasına izin vermek için kapsamlı dokümantasyonlar içermektedir. Sınıflama, Kümeleme gibi temel makine öğrenmesi uygulamaları yapılabilir.

Spark GraphX: Pregel ve Graphlab'a benzer bir grafik hesaplama arabirimi sağlar; bu sistemler (vertex bölümleme şemaları gibi), oluşturduğu RDD'ler için bölümleme işlevi seçimi ile aynı yerleştirme optimizasyonlarını uygular. Bu kütüphane ile graf tabanlı analizler yapılabilir. Sosyal ağlar için tercih edilen bir bileşen olabilir [51].



Şekil 2.5. Apache Spark Mimarisi

### 3. METİN İŞLEME

İnsanlar yapılandırılmamış az sayıda verinin işlenmesinde başarılıdırlar. Bilgisayarlarda çok sayıda yapılandırılmış verinin işlenmesinde iyidirler. Bu nedenle, metinlerin bilgisayarlar tarafından anlaşılmasını sağlayabilmek için bu verilerin işlenmesi gerekmektedir. Metinlerin ön işlenmesi doğal dil işleme alanının görevi ve metin işlemenin de öncül aşamasıdır [52].

Dil yeteneği, insan beyninin çalışma mekanizmasına ışık tutan insan türüne ait bir özellik olduğu için dilbilim, bilişsel bilimlerde önemli bir yer tutar. Dil, bilgisayar ortamında modellenenabilirse iletişim bakımından faydalı olabilecek bir araç geliştirilmiş olur. Doğal dil işleme, ana işlevi bir doğal dil çözümleme, anlama, yorumlama ve üretme olan bilgisayar sistemlerinin tasarımını ve gerçekleştirilmesini konu alan bir mühendislik alanıdır. Doğal dil işleme alanı yapay zekâ, biçimsel diller kuramı ve bilgisayar destekli dilbilim, psikoloji gibi çok farklı alanlarda geliştirilmiş kuram, yöntem ve teknolojileri bir araya getirmektedir. 1950 ve 1960'larda yapay zekânın küçük bir alt başlığı olarak tanımlanan bu alan, araştırmacılar tarafından geliştirilen başarılı uygulamalar ve sonuçları neticesinde bilgisayar bilimlerinin temel bir çalışma alanı olarak kabul edilmektedir [53]. Doğal dil işlemenin temel çalışma konuları ise şöyle sıralanabilir.

- Yazım yanlışlarının düzeltilmesi
- Bir metnin özetini çıkarma
- Metnin içerdiği bilgiyi çıkarma
- Bilgiye erişim
- Metni anlama
- Bilgisayarla sesli etkileşim
- Konuşmayı anlama (konuşmayı metne dönüştürme)
- Soru yanıt dizgeleri
- Doğal diller arası çeviri

Doğal dil işlemeyi ise beş temel aşamada incelemek mümkündür.

- ❖ Sesbilim
- ❖ Biçimbilim
- ❖ Sözdizimbilim
- ❖ Anlambilim
- ❖ Kullanımbilim

Doğal dil işleme, genellikle NLP (Natural Language Processing) olarak tanımlanmış yapay zekâ ve dilbilim alt disiplini. Türkçe, İngilizce, Fransızca gibi doğal dillerin işlenmesi ve kullanılması amacıyla araştırılma yapılan bilim dalıdır. Doğal dillerin kurallı yapısının çözümlenerek anlaşılması veya yeniden üretilmesi amacını taşımaktadır. Bu çözümlemenin insana getireceği kolaylıklar, yazılı dokümanların otomatik çevrilmesi, soru-cevap makineleri, otomatik konuşma ve komut anlama, konuşma sentezi, konuşma üretme, otomatik metin özetleme, bilgi sağlama gibi birçok başlıkla özetlenebilir. Bilgisayar teknolojisinin yaygın kullanımı, bu başlıklardan üretilen uzman yazılımların gündelik hayatımızın her alanına girmesini sağlamıştır. Örneğin, tüm kelime işlem yazılımları birer imlâ düzeltme aracı taşır. Bu araçlar aslında yazılan metni çözümleyerek dil kurallarını denetleyen doğal dil işleme yazılımlarıdır [54].

### **3.1. Metin İşlemeye Hazırlık ve Önileme**

Metin işleme “derlem” (corpus) olarak da adlandırılan belge koleksiyonunun oluşturulması ile başlar denilebilir. Derlem, metinden oluşan dokümanlar topluluğu olarak tanımlanabilir. Metin işleme çözümleri oldukça büyük derlemlerden, bilinmeyen örüntülerin tespit edilmesini amaçlamaktadır. Bu tür derlemlerdeki doküman sayısı binlerce, milyonlarca hatta günümüzde büyük verinin ortaya çıkması ile birlikte milyarlarca olabilmektedir. Derlem içerisindeki veriler klasik veri tabanı yapısı ile kıyaslandığında yapılandırılmamış ham verilerden oluşmaktadır. Bu veriler özel bir bilgisayar dilinde olabileceği gibi, doğal dilde de yazılmış olabilirler. Derlem içerisinde yer alan dokümanlar paragrafları, paragraflar cümleleri, cümleler ise kelimelerden oluşmaktadır. Bununla birlikte derlemin yapısı statik veya dinamik yapıda da olabilmektedir. Derlemlerin başlangıçtaki durumları değişmeden kalıyorsa statik yapıda olabilir. Buna karşın derleme zaman içerisinde yeni belgeler ekleniyor ve dokümanlar güncelleniyorsa, derlemin dinamik bir yapıda olduğu söylenebilir [55].

Başarılı metin işleme çalışmaları, gelişmiş veri önileme metodolojilerine dayanmaktadır. Metin işleme, yapılandırılmamış ham veri kaynaklarından yapılandırılmış görünümüleri ya anlamsal olarak ya alıntı yoluyla ya da her iki yoldan çıkaran çeşitli ön işleme tekniklerine o kadar bağlıdır ki, metin işlemenin bu ayrıntılı hazırlık teknikleri tarafından, kısıtlı oranda belirlendiği bile iddia edilmektedir [56].

Çok sayıda farklı metin önileme tekniği literatürde mevcuttur. Sonuç olarak hemen hemen tüm yöntemlerin amacı derlemin daha doğru bir ifade ile derlemi oluşturan doküman koleksiyonlarının yapılandırılmasıdır. Çeşitli metin ön işleme teknikleri, yaygın olarak ham

metin verilerinden yapılandırılmış doküman görünümlerini oluşturmak amacıyla birlikte kullanılmaktadır. Verileri yapılandırmaya yönelik önışleme tekniklerinin tümünü kategorize etmenin bir yolu, yapmış olduđu göreve göre kullandıkları algoritma ve biçimsel çerçevelere göre kategorileştirmedir [57].

Görev odaklı ön işleme yaklaşımları, yapılandırılmış doküman görünümünü oluşturma sürecini görev ve alt görevler şeklinde ön görmekte ve bir belgeden başlık ve yazarların alınması gibi, yerine getirilmesi gereken bir hazırlık amacını veya bir problemi içermektedir. Diğer önışleme yaklaşımları, doğal dil metinlerine de uygulanabilecek olan, karmaşık olayları analiz etmeye yönelik biçimsel yöntemlerden türetilen tekniklere dayanmaktadır. Bu tür yaklaşımlar arasında sınıflandırma şemaları, olasılıkçı modeller ve kurallara dayalı sistemler yaklaşımı bulunmaktadır.

Önışleme yapılırken yapılacak işlemler; hazırlık niteliğindeki işleme, genel amaçlı doğal dil işleme işlemleri ve probleme dayalı görevler olmak üzere 3 kategoride toplanabilir. Ön işleme aşaması hazırlık aşamasından sonra gelen ilk adımdır. Önışleme aşaması şu nedenlere bağlı olarak yapılmaktadır [58].

- Veri üzerinde bulunabilen problemleri çözmek,
- Verinin doğal yapısını öğrenerek daha anlamlı ve daha kaliteli analizler gerçekleştirebilmek,
- Veriden daha anlamlı bilginin üretilebilmesi olarak sıralanabilir.

Türkçe gibi bitişken (bükümlü) dillerde ise kelimeler, en küçük anlamlı parçasının sınırlarına dair bir belirti göstermez, üstelik bu parçalar, morfolojik ve fonolojik şartlara bağlı olarak şekil alırlar. Türkçe dili sondan eklemeli bir dil olduđu için ve bir kelimeye eklenecek her bir ek, o kelimenin anlamını değiştirmektedir. Bunun yanında tek bir Türkçe kelimedenden çok sayıda farklı anlamlı kelimeler oluşturulabilmektedir. Bu karmaşık morfolojik yapı nedeniyle, Türkçe metin işleme çalışmalarında, İngilizce ve benzeri dillerden daha farklı metin önışleme tekniklerine ihtiyaç duyulmaktadır [59].

### **3.1.1. İşaretleme (Tokenization)**

Bir derlem üzerinde çalışmaya başlayabilmek için yapılması gereken ilk önışlem adımı, karakter akışını kelimelere veya literatürdeki karşılığı ile “token” lara Türkçeleştirilmiş hali ile de “işaret”lere bölmektir. Bu metin işleme açısından oldukça önemli bir adımdır. Tokenlar elde edilmeden, derlem içerisinden daha üst düzey bilgilerin çıkarılması zordur. Ayrıca daha gelişmiş önışlem aşamalarından evvel bu adım yapılmalıdır [60]. Tokenization işlemi çeşitli düzeylerde yapılabilir. Derlem içerisinde, bölüm, kısım, paragraf, cümle, kelime ve hatta hece

ve karakterlere kadar ayrıştırılabilir. Ancak metin işleme çalışmalarında en fazla tercih edilen yöntem, metnin cümle ve kelimelere ayrılmasıdır.

Tokenizasyon işlemi, bir insan için basit bir görevken bir bilgisayar sistemi için daha zor bir görevdir. Bunun sebebi ise belirli karakterlerin kullanımına bağlı olarak, bazı durumlarda tokenların sınırlayıcı olurken bazı durumlarda ise sınırlayıcı olmamasıdır. Boşluk (space), sekme ve satırbaşı karakterlerinin hep sınırlayıcı olduğu, işaret olarak sayılmadığı varsayılmaktadır. Bunlara sıkça toplu olarak beyaz boşluklar (White spaces) adı verilmektedir. “( )”, “< >”, “!”, ve “?” karakterleri her zaman sınırlayıcı olup, aynı zamanda token olabilmektedir. “.”, “,”, “:” ve “-“ karakterleri ortamlarına bağlı olarak sınırlayıcı olabilir veya olmayabilirler [61].

Numaralar arasında nokta, virgül veya üst üste iki nokta işaretleri normalde sınırlayıcı değil, numaraların bir parçası olarak kabul edilmektedir. Her türlü başka virgül veya iki nokta sınırlayıcı olup token olabilmektedir. Nokta, kısaltmanın bir parçası olabilir. Aynı zamanda ardından boşluk geldiğinde ise kısaltmanın bir parçası olabilir. Ancak bunlardan bazıları aslında cümle sonudur. Tokenizasyon amaçları için belki de en iyi çözüm, belirsiz her noktayı kelime sınırlayıcı ve aynı zamanda token olarak ele almaktadır.

Kesme işaretinin de çeşitli kullanımları mevcuttur. Öncesinde ve sonrasında sınırlayıcı olmayanlar geldiğinde, ilgili işaretin bir kısmı olarak görülmelidir. Sonrasında anlamı kesin olan sonlayıcı geldiğinde, iç alıntı kapama tırnağı olabilir veya iyelik durumu gösterebilir. Öncesinde sonlayıcı gelen kesme işareti, kesin olarak iç alıntı açılış tırnağıdır, dolayısıyla içi alıntı açılış ve kapanış tırnakları izlenerek iki durum arasındaki fark görülebilmektedir.

Tire işareti öncesinde veya sonrasında başka bir tire işareti geldiğinde ise sonlayıcı ve token olduğu kabul edilir. İki numara arasındaki tire işareti, eksi işareti veya ayraç olarak kullanılan tire işareti olabilir. Belki de başka bir tire işaretine komşu olmayan tire işaretinin, sonlayıcı ve token olarak görülmesi en iyi yoldur fakat bazı uygulamalarda, çift tire durumu dışında bir tire işaretinin yalnızca bir karakter olarak görülmesi daha yerinde olacaktır [62].

Mümkün olan en iyi özellikleri elde etmek için işaretleme (tokenizasyon) işlemi için kullanılan (tokenizer), mevcut olan metne uyarlanmalıdır. Aksi halde, tokenlar elde edildikten sonra fazladan yük getirebilmektedir. Tokenizasyon sürecinin dile özgü bir süreç olduğunu unutmamak gerekir. Farklı diller için genel prensipler değişmeyebilir ancak ayrıntılar da muhakkak farklılıklar olacaktır.

### 3.1.2. Gövdeleme (Stemming, Lemmatization)

Karakter akımı token dizisine ayrıldıktan sonra, atılabilecek bir sonraki adım token olarak adlandırılan işaretlerden her birini standart forma dönüştürmektir. Bu sürece genellikle stemming veya lemmatization Türkçeleştirilmiş adı ile gövdeleme işlemi denilmektedir. Bu işlemin gerekli olup olmadığı uygulamaya bağlıdır. Doküman sınıflandırılması amacı için stemming bazı hallerde avantaj sağlayabilmektedir. Frekansın dikkate alındığı sınıflandırma algoritmaları için bazı durumlarda fark oluşturabilmektedir. Farklı amaçlar için ise, fazladan yapılacak stemming işlemi önemli oranda bir avantaj getirmeyebilir [63].

### 3.1.3. Sözlük Oluşturma

Kelimeleri ve bazı işaretleri bir arada barındıran kümeye genellikle *sözlük* adı verilir. Sözlüğün boyutu ise geliştirilen uygulamanın performansını etkileyebilecek bir kısıttır. Bu nedenle gereksiz kelimelerin temizlenmesi gerekmektedir. Sözlük boyutunu azaltmanın en temel yöntemlerinden birisi etkisiz kelimeleri (stopwords) denilen durdurucu kelimeler olarak Türkçeleştirilmiş bir çeşit etkisiz kelimeler listesinin derlem içerisinden çıkarılmasıdır. Bu kelimeler, genellikle geliştirilen uygulamanın tahmin performansını etkilemeyecek kelimelerdir. Etkisiz kelimeleri (stopwords) kelimelere örnek olarak İngilizce için “a”, ”an”, “the” gibi tokenlar ve Türkçedeki “ve”, “veya” gibi kelimeler “ben”, “sen” gibi zamirler verilebilir.

Kelime sayısının azaltılarak sözlük boyutunun küçültülmesi modelin performansına olumlu bir etki yapacağı gibi bazı durumlarda tahmin yapma performansını iyileştirebilir. Derlem içerisinde en sık karşılaşılan kelimeler etkisiz (stopwords) kelimeler olduğu için bu kelimelerin temizlenmesinin modelin performansını iyileştirebilir. Bununla birlikte çok nadir karşılaşılan kelimelerinde yazım hatası olma olasılığı yüksek olduğu için bu kelimelerde derlem içerisinden çıkarılabilir. Bu tez çalışmasında da Türkçe dili için etkisiz (*stopwords*) kelimeler TF-IDF denilen yöntem ile tespit edilerek derlem içerisinden temizlenmiştir.

## 3.2. Kelime Temsil Yöntemleri

Bu bölümde metin gösterim metotları açıklanmıştır. Kelime temsil yöntemleri, Frekans Bazlı Temsil ve Tahmin Bazlı temsil olmak üzere ikiye ayrılmaktadır. Daha geleneksel yöntemler olarak tanımlanan Frekans bazlı kelime temsilleri, dokümanlar içerisinde bulunan kelimeler ve bu kelimelerin frekanslarının tespit edilmesi prensibine dayanmaktadır [64].

Frekans bazlı kelime temsiline dayalı olarak geliştirilmiş ve en çok tercih edilen yöntem Kelimeler Çantası (Bag of Words) metodudur. Bu yönteme göre, doküman içerisindeki her cümle benzersiz (unique) kelimelere ayrılarak, benzersiz kelime boyutunda bir matrise dönüştürülür. Matrisin sütunları doküman içerisindeki kelimelerden oluşurken (N), satırları ise doküman sayısından meydana gelir (D). Sonuç olarak tüm derlem  $D \times N$  boyutunda bir matris olarak temsil edilmiş olur [65]. Frekans bazlı temsil yöntemlerinden bir diğeri ise TF-IDF (Term Frequency—Inverse Document Frequency) yöntemidir. Terim frekansı (Term Frequency), bir doküman içerisinde geçen terim frekanslarını tespit etmek için kullanılır. Ters Doküman Frekansı (Inverse Document Frequency) yöntemi ise birden fazla dokümanda kelimenin geçme sayısını bularak bu kelimenin terim olup olmadığını bağlaç vb (stopwords-etkisiz kelime-) olduğu anlamaya çalışır [66]. Bu çalışmada da oluşturulan modelin verimliliğini artırmak için Türkçe kelime etkisiz kelimeler (stopwords) listesine ihtiyaç duyulmuş ancak bu konuda tatmin edici bir liste bulunmadığı için çalışmanın bir katkısı olarak Inverse Document Frequency yöntemi kullanılarak bir stopwords (etkisiz kelimeler) listesi oluşturulmuş bununla ilgili bilgiler çalışmanın ilerleyen kısımlarında verilmiştir. Eş-oluşum matrisi (co-occurrence matrix) ise benzer kelimelerin cümle içerisinde birlikte geçme eğiliminde olması ilkesine dayanmaktadır. Bu yöntemde oluşturulan matrisin hem satırlarında hem sütunlarında benzersiz (unique) kelimeler yer almaktadır. Matrisin hücrelerini ise, satır ve sütunda yer alan kelimelerin birlikte geçme frekanslarını içermektedir [67].

Frekans bazlı yöntemlerin en önemli iki dezavantajı vardır. Birincisi, kullanılan yönteme göre satır ve/veya sütunlarda benzersiz kelimeler bulunduğu için kelime sayısı boyutunda matrisler meydana gelir ve bunların birçoğunun değeri sıfır olacağından seyrek matris (sparse matrix) denilen durum ortaya çıkmaktadır. Bir diğeri önemli dezavantajı ise, kelimeler arasındaki anlamsal yakınlıkların tespit edilememesi durumudur [68].

Kelime temsil yöntemlerinden bir diğeri olan tahmin bazlı kelime temsil yöntemi ailesinden olan Mikolov ve arkadaşları [69] tarafından 2013 yılında geliştirilmiş olan temelinde yapay sinir ağı ile kelimelerin eğitilmesi ilkesine dayanan Word2Vec modelidir. Bu model giriş olarak alınan kelimelere dayanarak hedef kelimeyi tahmin etme prensibine dayalıdır. Word2Vec modeli CBOW (Continuous Bag of Words) ve Skip-gram olarak adlandırılan iki farklı algorithmadan oluşmaktadır.

Tahmin bazlı bir kelime temsil yöntemi olan Word2Vec yöntemi, girdi (input) olarak büyük bir derlem içerisindeki benzersiz kelimeleri alır ve belirlenen bir temsil vektörü boyutunda matris oluşturulmaktadır. Word2Vec modeli her defasında dokümandaki bir

cümleyi pencere (window) denilen bir yapı içerisinde kaydırarak taramakta ve penceredeki hedef kelimeye göre genellikle çok sayıda boyuttan oluşan bir vektörü çıktı olarak üretmektedir [70].

Word2Vec yönteminin genel olarak birbirine benzeyen CBOW (Continuous Bag of words) ve Skip – Gram olmak üzere iki tür alt yöntemi vardır. Bu iki modelin temel olarak farklılıkları ise girdi ve çıktıları alma yöntemlerine dayanmaktadır [71].

### 3.2.1. TF (Term Frequency) - IDF (Inverse Document Frequency)

TF-IDF (Term Frequency—Inverse Document Frequency) yönteminde, Term Frequency, bir doküman içerisinde geçen terim frekanslarını tespit etmek için kullanılır. Inverse Document Frequency yöntemi ise birden fazla dokümanda kelimenin geçme sayısını bularak bu kelimenin terim olup olmadığını bağlaç vb (stopwords-etkisiz kelime-) olduğu anlamaya çalışır [72].

Bu tez çalışmasında da oluşturulan modelin verimliliğini artırmak için Türkçe etkisiz (stopwords) kelimeler listesine ihtiyaç duyulmuş ancak bu konuda tatmin edici Türkçe bir liste bulunmadığı için çalışmanın bir katkısı olarak TF-IDF yöntemi kullanılarak bir stopwords (etkisiz kelimeler) listesi oluşturulmuş bununla ilgili bilgiler çalışmanın ilerleyen kısımlarında verilmiştir.

Bu yöntem, genellikle kelimelerin ağırlıkları veya skorlarını hesaplamak için kullanılan bir yöntemdir. Bu aşamada derlem içerisinde, öncelikle tekil (unique) kelimelerin sayımları yapılarak kelimelerin frekansları tespit edilmelidir. Denklem 3.1. ile j kelimesine atanan TF-IDF ağırlığının, kelimenin önemine bağlı olarak ölçek faktörü ile uyarlanmış terim frekansı (term frequency) olduğu görülmektedir. Ölçek faktörü denklem 3.2. ile gösterilen ters doküman frekansı (inverse document frequency) olarak adlandırılır. Frekans, j kelimesini içeren doküman sayısını (yani  $df(J)$  değerini) tespit ederek, ölçeklemeyi tersine çevirir. Böylece, kelime birçok dokümanda geçiyorsa, önemsiz sayılacak ve ölçek, belki de sıfıra yaklaşacak kadar düşecektir. Kelime nispeten nadir olup az sayıda dokümanda geçiyorsa ölçek faktörü yukarı yönde artacak ve kelime önemli olarak görülecektir.

Denklem 3.1’de  $tf_{i,j}$  ile terim sıklığı yani TF değeri hesaplanmaktadır. Bu değer eşitlikte de görüldüğü üzere  $TF = \text{kelimenin doküman da geçme sayısı} / \text{dokümandaki kelime sayısı}$  ile hesaplanmaktadır.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \quad (3.1)$$

3.2. numaralı eşitlik kullanılarak IDF değeri hesaplanmıştır.  $IDF = \log ( \text{toplam doküman sayısı} / \text{kelimeyi içeren doküman sayısı} )$

$$idf(w) = \log \left( \frac{N}{df_t} \right) \quad (3.2)$$

TF-IDF matrisinin ağırlıklarının oluşturulması için ise 3.3 numaralı eşitlik kullanılmıştır.

$$w = tf * idf \quad (3.3)$$

### 3.2.2. Kelime Çantası ( Bag of Words)

Bu yöntemde göre, doküman içerisindeki her cümle benzersiz (unique) kelimelere ayrılarak, benzersiz kelime boyutunda bir matrise dönüştürülür. Matrisin sütunları doküman içerisindeki kelimelerden oluşurken (N), satırları ise doküman sayısından meydana gelir (D). Sonuç olarak tüm derlem DxN boyutunda bir matris olarak temsil edilmiş olur [73].

Eş-oluşum matrisi (co-occurrence matrix) ise benzer kelimelerin cümle içerisinde birlikte geçme eğiliminde olması ilkesine dayanmaktadır. Bu yöntemde oluşturulan matrisin hem satırlarında hem sütunlarında benzersiz (unique) kelimeler yer almaktadır. Matrisin hücrelerini ise, satır ve sütunda yer alan kelimelerin birlikte geçme frekanslarını içermektedir [74].

### 3.2.3. Gömme (Embedding) Yöntemleri

Geleneksel yöntemlerin en önemli iki dezavantajı vardır. Birincisi, kullanılan yöntemde göre satır ve/veya sütunlarda benzersiz kelimeler bulunduğu için kelime sayısı boyutunda matrisler meydana gelir ve bunların birçoğunun değeri sıfır olacağından seyrek matris (sparse matrix) denilen durum ortaya çıkmaktadır. Bir diğer önemli dezavantajı ise, kelimeler arasındaki anlamsal yakınlıkların tespit edilememesi durumudur [75].

Kelime temsil yöntemlerinden bir diğeri olan gömme (embedding) yöntemleri ailesinden olan Mikolov ve arkadaşları [69] tarafından 2013 yılında geliştirilmiş olan temelinde yapay sinir ağı ile kelimelerin eğitilmesi ilkesine dayanan Word2Vec modelidir. Bu model giriş olarak alınan kelimelere dayanarak hedef kelimeyi tahmin etme prensibine dayalıdır. Word2Vec modeli CBOW (Continuous Bag of Words) ve Skip-gram olarak adlandırılan iki farklı algorithmadan oluşmaktadır.

Sinir ağı temelli bir kelime temsil yöntemi olan Word2Vec yöntemi, girdi (input) olarak büyük bir derlem içerisindeki benzersiz kelimeleri alır ve belirlenen bir temsil vektörü boyutunda matris oluşturulmaktadır. Word2Vec modeli her defasında dokümandaki bir cümleyi pencere (window) denilen bir yapı içerisinde kaydırarak taramakta ve penceredeki

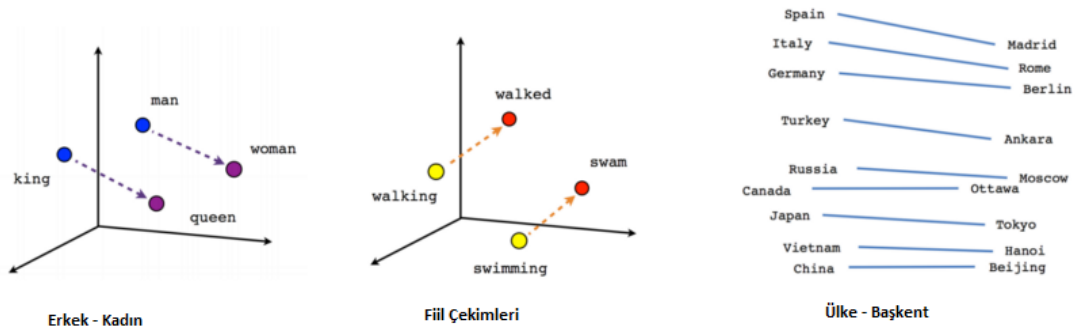
hedef kelimeye göre genellikle çok sayıda boyuttan oluşan bir vektörü çıktı olarak üretmektedir [76].

Word2Vec yönteminin genel olarak birbirine benzeyen CBOW (Continuous Bag of words) ve Skip – Gram olmak üzere iki tür alt yöntemi vardır. Bu iki modelin temel olarak farklılıkları ise girdi ve çıktıları alma yöntemlerine dayanmaktadır [71].

Kelime temsili (word embedding), kelimelerin harflerden vektör biçiminde sayılara dönüştürülmesi olarak açıklanabilir. Kelime temsil metotları ise bu dönüştürme işleminde tercih edilen tekniklerdir. Literatür incelendiğinde, özellikle sinir ağlarında son dönemlerde yaşanan gelişmelerle birlikte kelime temsili metin işleme çalışmalarında en kritik noktalardan bir olduğu kabul edilmektedir [76].

Kelime temsili (word embedding), kelimelerin harflerden vektör biçiminde sayılara dönüştürülmesi olarak açıklanabilir. Bir derlem içerisindeki metinlerin nasıl temsil edileceği metin işleme çalışmalarının en kritik noktalarından biri ayrıca bir ağın en önemli girdisidir. En basit tanımıyla, metinlerin sayısal verilere dönüştürülmesine kelime temsili (word embedding) denilmektedir. Aynı metin farklı şekilde farklı sayısal değerlere dönüştürülebilmektedir. Kelime temsil metotları ise bu dönüştürme işleminde tercih edilen tekniklerdir. Literatür incelendiğinde, özellikle sinir ağlarında son dönemlerde yaşanan gelişmelerle birlikte kelime temsili metin işleme çalışmalarında en kritik noktalardan bir olduğu kabul edilmektedir.

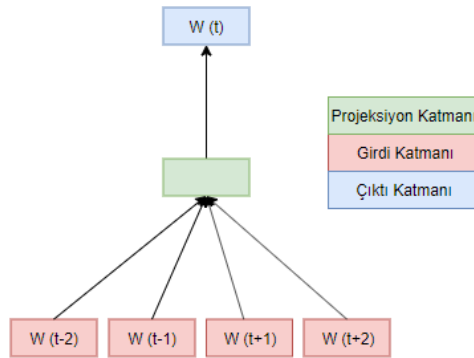
Kelime temsil yöntemlerinden bir diğeri olan tahmin tabanlı kelime temsil yöntemi ailesinden olan Mikolov ve arkadaşları tarafından 2013 yılında geliştirilmiş olan temelinde yapay sinir ağı ile kelimelerin eğitilmesi ilkesine dayanan Word2Vec modelidir. Bu model giriş olarak alınan kelimelere dayanarak hedef kelimeyi tahmin etme prensibine dayalıdır. Şekil 3.1.'de gösterilen bu yöntem ile kelimeler arasındaki ilişkiler tespit edilebilmektedir. Örneğin şekilde cinsiyetler, fiiller ve zamanlar, ülkeler ve başkentler arasındaki ilişkiler görülmektedir. Word2Vec modeli CBOW (Continuous Bag of Words) ve Skip-gram olarak adlandırılan iki farklı modelden oluşmaktadır [77].



Şekil 3.1. Kelime temsil yöntemleri ile ilişkilerin tespiti [69].

### 3.2.4. CBOW (Continuous Bag of Words)

CBOW modelinde pencere boyutu (window size) merkezinde olmayan kelimeler girdi olarak alınıp, merkezinde olan kelimeler çıktı olarak tahmin edilmeye çalışılmaktadır. Bu işlem cümle bitimine kadar devam etmektedir [76].



Şekil 3.2. CBOW modelinin yapısı.

Bu durum Şekil 3.2.'de gösterilmeye çalışılmıştır. Burada  $w(t)$  ile gösterilen değer, cümlenin merkezinde bulunan ve tahmin edilmek istenen çıktı değeri iken,  $w(t-2), \dots, w(t+2)$  ile gösterilen değerler ise tercih edilen pencere boyutuna göre (window\_size) merkezde olmayan çıktı değerleridir.

Şekil 3.2.'de açıklanan CBOW modelinin uygulaması dünyaca ünlü yapay zekâ araştırmacılarından biri olan Andrew Ng'nin söylediği bir söz üzerinden açıklanmak istenirse

**Örnek Cümle:** “Yapay zekâ yeni elektriktir.”

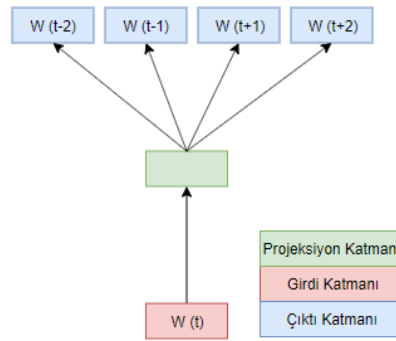
Bu cümleyi girdi olarak alan ve pencere boyutu=1 olan CBOW modeli şu şekilde çalışmaktadır. Önce “yapay” kelimesini pencerenin merkezine oturur, sonra sağındaki ve solundaki 1'er kelimeyi ayrı ayrı girdi olarak alıp (window\_size=1) pencerenin merkezinde bulunan “yapay” kelimesi sinir ağı modeli ile tahmin edilmeye çalışılmaktadır. Sonra pencere 1 sağa kaydırılarak, bu sefer de pencerenin merkezine “zeka” kelimesi gelmektedir. Çizelge 1’de cümlelerin adım adım CBOW modeline göre işletilmesi gösterilmiştir. Kırmızı renkle yazılmış kelimeler çıktıyı, mavi renkli kelimeler ise girdiyi ifade etmektedir.

Çizelge 3.1. CBOW modeli

|   | Cümle (window_size=1) |      |      |             | GİRDİ                | ÇIKTI         |
|---|-----------------------|------|------|-------------|----------------------|---------------|
| 1 | Yapay                 | zeka | yeni | elektriktir | (zeka)               | (yapay)       |
| 2 | Yapay                 | zeka | yeni | elektriktir | (yapay) (yeni)       | (zeka) (zeka) |
| 3 | Yapay                 | zeka | yeni | elektriktir | (zeka) (elektriktir) | (yeni) (yeni) |
| 4 | Yapay                 | zeka | yeni | elektriktir | (yeni)               | (elektriktir) |

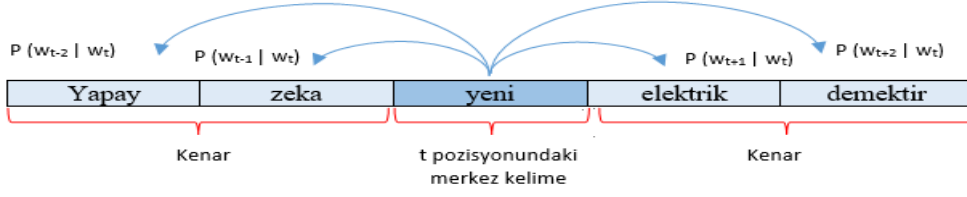
### 3.2.5. SG (Skip-Gram Model)

Skip-Gram modelinde ise CBOW modelinin tersi şeklinde bir işleyişe sahiptir. Merkezdeki kelime girdi olarak alınıp merkezde olmayan kelimeler çıktı olarak tahmin edilmeye çalışılır. Bu işlem cümle bitene kadar devam eder. Bu durum Şekil 3.3.’te gösterilmiştir [71].



Şekil 3.3. Skip-Gram modelinin yapısı.

CBOW modelinde uygulaması yapılan örnek cümle Skip-Gram modelinde uygulanması ise Şekil 3.4.’te ki gibi olmaktadır.



Şekil 3.4. Skip-Gram yönteminin uygulanması

Şekil 3.4.'te anlatılan skip-gram algoritmasının optimize edilecek hata fonksiyonu ise 3.4 numaralı eşitlikte gösterilmiştir. Bu denklemden pencere büyüklüğü  $m$  ile ifade edilmiştir. Denklem, derlem içerisindeki tüm kelimeler için  $t$  pozisyonundaki bir kelimenin solunda ve sağında bulunan  $m$  (pencere boyutu) kadar kelimenin tahmin edilmesine dayanmaktadır.

$$J = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m} \log P(w_{t+j} | w_t) \quad (3.4)$$

$t=1, \dots, T$  ile ifade edildiği gibi bu işlem derlem içerisindeki tüm kelimeler için yapılarak, her kelimenin etrafındaki kelimeleri yüksek olasılıkla tahmin edebilen vektörler oluşturulmaktadır. Olasılıkların hesaplandığı kısım ise eşitlik 3.5. ile daha ayrıntılı olarak gösterilmiştir.

$$P(w_{t+j} | w_t) = \frac{\exp(u_{w_{t+j}}^T v_{w_t})}{\sum_{w=1}^v \exp(u_w^T v_{w_t})} \quad (3.5)$$

Burada dikkat edilmesi gereken husus büyüklüğü  $m$  olan bir pencere içerisindeki kelimenin tahmin edilmesi işleminde, her kelimenin biri söz konusu kelimenin merkezde yani  $w_t$  durumunda iken bir diğeri ise kelime pencere içerisinde kenar kelime durumunda yani  $w_{t+j}$  durumunda olmak üzere iki adet vektörünün olmasıdır. Her kelime için kenar kelimelere göre olasılıkların nasıl hesaplandığını gösteren 3.5. numaralı formülde  $v$  sembolü o kelimenin merkez kelime iken vektör değerini,  $u$  ise kelimenin pencere içerisinde kenar kelime durumundayken vektörel değerini ifade etmektedir.

Pencere boyutunun 2 olduğu bir olasılık hesaplama işlemi örneği aşağıdaki gibidir. Bölme işleminin paydasında ise, bu işlemin derlem içerisinde yer alan tüm kelimeler için yapılarak değerlerin toplandığı görülmektedir.

$$p(\text{yapay} | \text{yeni}) = \frac{\exp(u_{\text{yapay}}^T v_{\text{yeni}})}{\sum_{w=1}^v \exp(u_w^T v_{\text{yeni}})}$$

$$p(\text{zeka} | \text{yeni}) = \frac{\exp(u_{\text{zeka}}^T v_{\text{yeni}})}{\sum_{w=1}^v \exp(u_w^T v_{\text{yeni}})}$$

Çizelge 3.2. Skip-Gram modeli

|   | Cümle (window_size=1) |      |      |             | GİRDİ         | ÇIKTI                |
|---|-----------------------|------|------|-------------|---------------|----------------------|
| 1 | Yapay                 | zeka | yeni | elektriktir | (yapay)       | (zeka)               |
| 2 | Yapay                 | zeka | yeni | elektriktir | (zeka) (zeka) | (yapay) (yeni)       |
| 3 | Yapay                 | zeka | yeni | elektriktir | (yeni) (yeni) | (zeka) (elektriktir) |
| 4 | Yapay                 | zeka | yeni | elektriktir | (elektriktir) | (yeni)               |

Çizelge 3.1. ve Çizelge 3.2. incelendiğinde Skip-Gram modelinin esasen CBOW modeli ile çok benzer olduğu daha öncede ifade edildiği gibi temel farklılığın kelimeleri girdi olarak alışı şekillerinden kaynaklandığı net olarak ortaya çıkmaktadır [78].

### 3.2.6. Glove

Glove, adını “Global Vectors for Word Representation” kelimelerinin baş harflerinden alan bir diğer kelime temsil yöntemidir. Doğal dil işleme alanında Word2Vec yönteminden sonra en çok kullanılan yöntemdir. Stanford üniversitesinde Pennington ve arkadaşları tarafından geliştirilmiştir [79].

Bir derlem üzerinde yakın anlamlı kelimelerin tespit edilmesinde Word2Vec yöntemine göre daha az başarılıdır. Glove, global kelime-kelime sayıları üzerinde eğitilen ve böylece istatistiklerin daha etkili olarak kullanılmasına olanak sağlayan bir yöntemdir. Glove modeli, analogik veri seti üzerinde %75 doğruluk oranına sahip bir kelime uzay modeli üretmektedir. Bu da Word2Vec yönteminin analogi tespitlerinde daha başarılı olduğunu göstermektedir. Glove yöntemine göre optimize edilecek hata fonksiyonu 3.6. numaralı denklem ile gösterilmiştir.

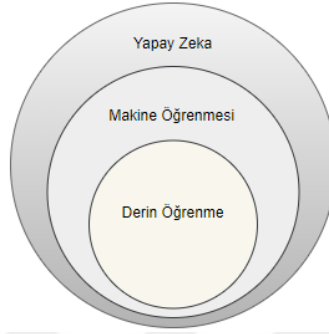
$$J = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij}) (u_i^T v_j - \log P_{ij})^2 \quad (3.6)$$

3.6. numaralı eşitliğe göre  $u$  ve  $v$  simgeleri global kelime-kelime sayıları ile oluşturulan matrisin satır ve sütun değerlerini göstermektedir. Eşitlikte ki  $f(P_{ij})$  tanımlanmış ağırlık fonksiyonudur.  $W$  ise oluşturulan sözlüğün boyutudur [80].

## 4. DERİN ÖĞRENME

### 4.1. Yapay Zekâ

Yapay zekâ, 1950’li yılların başında ortaya çıkmaya başlamış ve o zamanlar makinelerin düşünüp düşünemeyeceğini araştıran bir bilim dalı olarak görülmekteydi. Günümüzde ise yapay zekâ kavramı, insanlar tarafından yapılması beklenen zihinsel aktivitelerin otonom hale getirilmesi olarak tanımlanmaktadır. Yapay zekâ, makine öğrenmesi ve derin öğrenmeyi kapsayan oldukça geniş bir araştırma alanıdır. Bu durum Şekil 4.1.’de görülmektedir.



Şekil 4.1. Yapay zekânın görünümü

1950’li yıllarda kullanılan yapay zekâ, satranç oyunu gibi kuralları insanlar tarafından teker teker yazılan bir model olduğu için bu yaklaşım makine öğrenmesi olarak değerlendirilmemiştir. Çünkü bu yaklaşımda herhangi bir öğrenme işleminin olmadığı görülmüştür. Bu nedenle bu yaklaşıma sembolik yapay zekâ adı verilmiştir. Dahası bu tür yaklaşımlar satranç gibi belirli kuralları olan problemlerde başarılı sonuçlar ortaya koysa da görüntü işleme veya metin işleme gibi daha kompleks problemlerde pek başarılı olamamıştır. Ancak uzman sistemlerin gelişmesiyle birlikte donanım birimlerinde yaşanan gelişmeler sonrasında makine öğrenmesi kavramı ortaya çıkmıştır [81].

### 4.2. Makine Öğrenmesi

Bu tez çalışmasında kullanılan derin öğrenme modelleri ve sinir ağı mimarileri makine öğrenmesinin özel bir türüdür. Derin öğrenme, son yıllarda yapay zekâ kavramının ortaya çıkmasıyla birlikte özellikle endüstri ve sanayii alanında oldukça kullanılan yaygın kullanıma sahip teknolojilerdir. Makine öğrenmesinin temel prensibi, veri ve beklenen çıktılardan faydalanan yeni verilere özgün cevaplar üretebilecek kuralları öğrenmesidir. Yani makine öğrenmesinde tıpkı insanlarda olduğu gibi bir öğrenme söz konusudur. Üzerinde çalışılan

göreve göre eldeki örneklere bakarak istatistiksel örüntüleri tespit ederek görevi otomatikleştirecek kuralları ortaya çıkarmaktadır [82].

Makine öğrenmesi 1990’larda ortaya çıksada donanım alanında yaşanan gelişmeler ve büyük veri setlerinin üretilmeye başlanması ile birlikte yapay zekânın en çok tercih edilen alt alanı olmuştur.

### 4.3. Derin Sinir Ağları

Derin öğrenme, yapay zekâ kavramının içinde barındırdığı makine öğrenmesi konusunun özel bir halidir. Literatür incelendiğinde ilk sinir ağı perceptron algoritmasıdır [83]. Bu ağ bir girdi ve bir çıktı katmanından oluşmaktaydı. Bu sinir ağlarına birden fazla katmanın eklenmesiyle daha karmaşık bağlantıların belirlenmesi hedeflenmiştir. Oluşturulan bu yeni mimariye ise derin sinir ağı mimarisi denilmiştir. Yani derin sinir ağları için klasik yapay sinir ağlarının çok katmanlı ve çok nöronlu özel bir hali denilebilir.

Son yıllarda daha hızlı ve güçlü işlem yapabilen grafik işlem birimlerinin kullanılmasıyla birlikte hızlı işlem yapabilmenin maliyeti düşmüştür. Bu gelişme de derin sinir ağları oldukça popüler hale getirmiştir [84]. Derin öğrenme, alanında yaşanan gelişmeleri sağlayan sebepler,

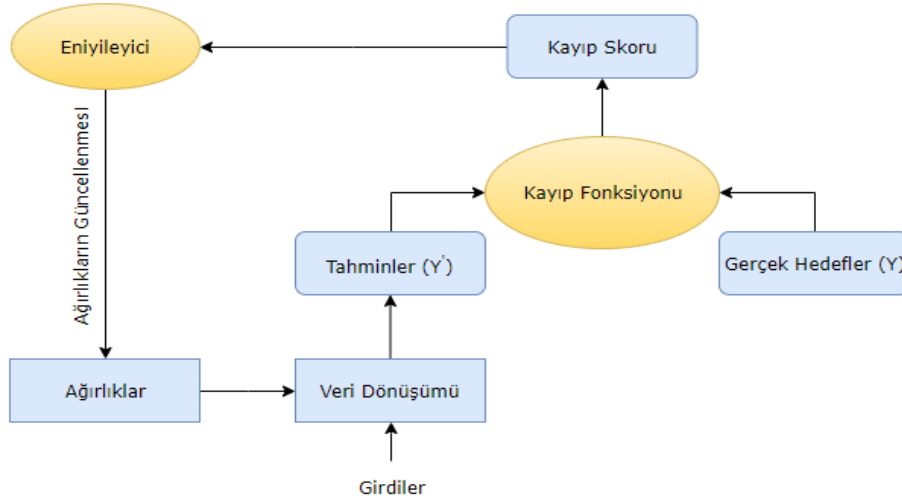
- Donanım alanında özellikle grafik kartlarında sağlanan ilerlemeler
- Çok çeşitli veri setlerinin üretilmesi ve bu veri setlerinin araştırmacılarla paylaşımın kolay hale gelmesi
- Algoritmalarda yapılan iyileştirmeler olarak 3 temel başlıkta toplanabilir.

Derin öğrenmenin günümüzde çok popüler olmasının nedeni çeşitli problemler için başarılı performanslar sunmasıdır. Bunun yanı sıra, derin öğrenmede makine öğrenmesinin en önemli aşamalarından biri olan özellik çıkarımının otomatik olarak yapılabilmesi de önemli bir avantajdır.

### 4.4. Sinir Ağlarının Yapısı

Bir sinir ağı yapısı Şekil 4.2.’de görülen şu bölümlerden oluşmaktadır.

- Katmanlar
- Girdi verisi ve hedefler
- Kayıp fonksiyonu
- Eniyileme algoritması



Şekil 4.2. Sinir ağının çalışma mantığı

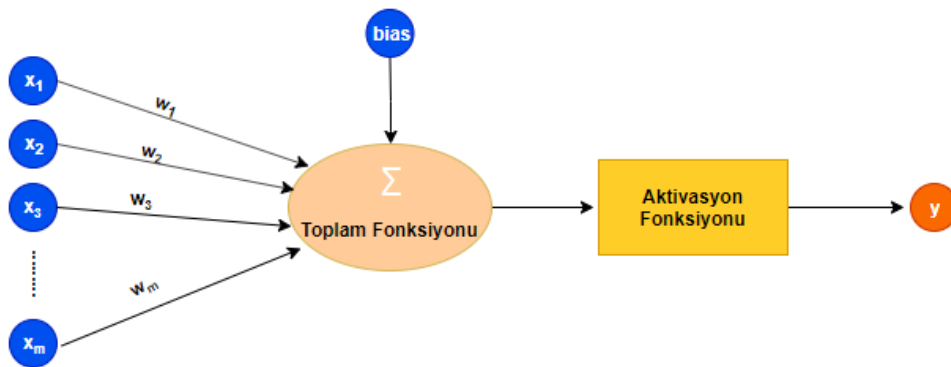
Yapay sinir ağları, en basit tabirle insan beyninin çalışma prensipleri esas alınarak, bilgisayar ortamında matematiksel metodolojilerle modellenmesi olarak tanımlanabilir. Yapay sinir ağları var olan verileri işleyerek, bu verileri öğrenme algoritmaları ile kullanarak var olmayan bilgileri elde etmeyi amaçlamaktadır [85].

Sinir sisteminde 100 milyardan fazla nöron bulunmaktadır. Bir nöron dentritten aksone doğru sinyalleri taşımaktadırlar. Nöronlar girdi olarak bir elektrik sinyalini alır ancak çıktı olarak farklı bir sinyal üretirler. Yapay sinir ağlarında kullanılan bazı terimler şöyle açıklanmıştır.

Threshold (Eşik değeri): Bir çıktı üretilmeden önce aşılması gereken eşik değeridir.

Activation Function (Aktivasyon fonksiyonu): Eşik değerini dikkate alarak çıktı üreten fonksiyonlardır.

Şekilde 4.3.'te basit bir yapay sinir ağı mimarisi örneği görülmektedir.



Şekil 4.3. Yapay sinir ağının mimarisi

Şekil 4.3.'te  $x$  değerleri ağırlık girdileri  $w$  değerleri ise ağırlıkları göstermektedir. Bu değerler toplam fonksiyonunda 4.1 numaralı eşitlikte olduğu gibi bias değeri ile birlikte toplanarak hesaplanır.

$$\sum_{i=1}^m (w_i x_i) + bias \quad (4.1)$$

Hesaplanan bu değer belirlenmiş bir eşik değeri ile aktivasyon fonksiyonuna sokulur ve tercih edilen aktivasyon fonksiyonuna göre  $y$  değerini yani çıktıyı üretir. Yapay sinir ağları doğrusal olmayan problemlerin çözümünde tercih edilir. Yapay sinir ağlarında, üzerinde çalışılan veriler eğitim verisi ve test verisi olmak üzere iki küme olarak ele alınmaktadır. Eğitim işleminin amacı, beklenen çıktı değeri üretilinceye kadar ağırlık  $w$  ile gösterilen ağırlıkların güncellenerek hata değerinin en alt seviyeye indirilmesidir. Bu kademeli işleme eğitim yada öğrenme denilmektedir. Eğitim işlemi tamamlandıktan sonra sinir ağına hiç kullanılmayan test verileri ile ağırlık başarımı ölçülür [86].

#### 4.4.1. Evrişimli Sinir Ağları - Convolutional Neural Networks (CNN)

Çok katmanlı algılayıcıların özel bir türü olarak geliştirilen CNN mimarileri insan görme sisteminden esinlenerek tasarlanmıştır. Düzenli zaman aralıklarında girdiler alan bir boyutlu bir ızgara, zaman dizisi verisini ve iki boyutlu bir piksel ızgarası olarak düşünülebilecek görüntü verisini içermektedir. *Evrişim* adını matematiksel özel bir tür doğrusal işlemden almaktadır. Evrişimli sinir ağları, ağırlık katmanlarından en az birinde genel matris çarpımı yerine evrişim işlemini kullanan ağlar olarak tanımlanabilir. Başlangıçta görüntü işleme çalışmaları için önerilmiş bir derin sinir ağı mimarisidir [87].

Ancak günümüzde Görüntü Sınıflandırma (Image Classification), Nesne Bulma (Object Detection), Nesne Takip Etme (Object Tracking), Doğal Dil İşleme (Natural Language Processing), Stil Transferi (Style Transferring) gibi pek çok alanda kullanılmış ve başarılı sonuçlar elde edilmiştir [88].

CNN mimarileri görüntü işleme alanında değil de doğal dil işleme alanında kullanıldığında görüntü işlemedeki gibi matrislerle ifade edilen değerler pikseller değil cümleler veya dokümanlar olmaktadır. Matrisin her bir satırı bir kelimeyi temsil eden bir vektördür. Yani bu vektörler word embedding olarak bilinen kelime vektörleridir. 10 kelimeden oluşan bir cümle için her bir kelimenin 100 boyutlu bir vektörle ifade edildiği varsayılırsa 10x100 boyutunda bir giriş matrisi olacaktır [89,90].

Görüntü işleme çalışmalarında filtre görüntü üzerinde adım adım kaydırılırken, metin işleme de filtre matrisin (kelime) tüm satırı üzerinde kaydırılmaktadır. Bu nedenle matrisin genişliği genellikle kelime vektör boyutu ile aynıdır [91,92].

Evrişim işlemi 4.2 de yer alan formülle gösterilebilir.

$$s(t) = \int x(a)w(t - a)da \quad (4.2)$$

Bu işleme evrişim adı verilmektedir. Evrişim işlemi 4.3. denklemindeki gibi tipik olarak yıldız işareti ile gösterilir.

$$s(t) = (x * w)(t) \quad (4.3)$$

Evrişimsel ağ terminolojisine göre, evrişim işleminin ilk değişkeni (x fonksiyonu) genellikle girdi olarak ikinci değişkeni (w fonksiyonu) kernel olarak isimlendirilebilir.

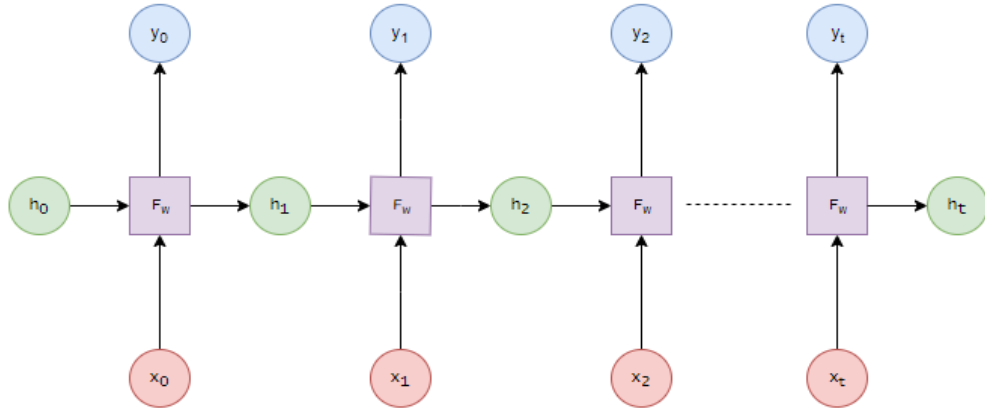
#### 4.4.2. Yinelemeli Sinir Ağları-Recurrent Neural Networks (RNN)

Yinelemeli sinir ağları, zamana bağlı olarak değişen sıralı (sequence) bilgi işlemede kullanılan bir yapay sinir ağı türüdür. Sıralı bir akışa sahip verilerde kullanılmasından dolayı başta doğal dil işleme problemleri olmak üzere konuşma, müzik, video işleme gibi alanlarda yaygın olarak tercih edilmektedir. Klasik yapay sinir ağı yapısından farklı olarak katmanlardan sonraki katmanların girişlerine geri besleme yapılmaktadır [93,94]. Şekil 4.4.'te RNN yapısı verilmiştir.

$$h_t = fw(h_{t-1}, x_t) \quad (4.4)$$

Denklem 4.4'te gösterildiği üzere  $fw$  değeri  $w$  parametrelili bir fonksiyonu gösterirken, bu fonksiyon denklemde  $x_t$  ile ifade edilen bir giriş vektörü ve bir önceki durumdan gelen  $h_{t-1}$  ile birlikte o anki durumun çıkış değeri yani  $h_t$  olmaktadır.

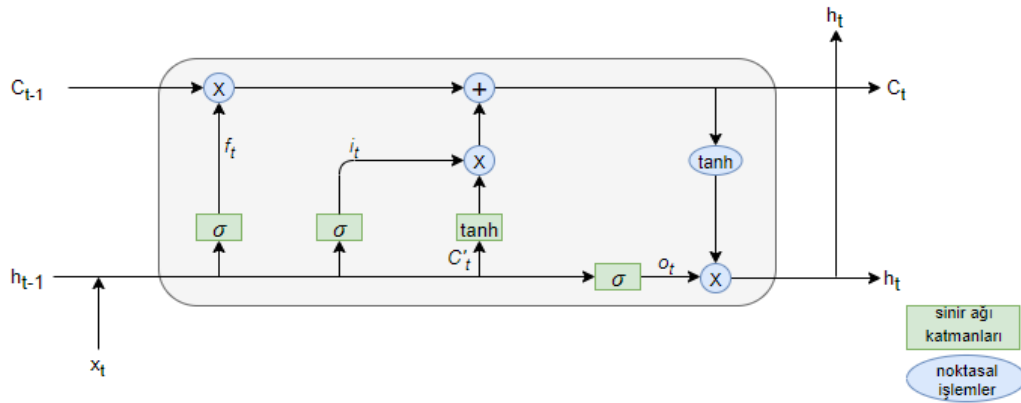
Ancak basit RNN mimarisinin karşılaştığı iki temel problem vardır. Gradient, tüm ağırlık değerlerini ayarlamamızı sağlayan gradyan olarak Türkçeleştirilmiş bir değerdir. Ancak birbirine bağlı uzun ağlarda kayıp (loss) etkisi oldukça düşerek gradyan değeri kaybolmaya başlayabilir. Bütün katmanlar birbirine çarpımla bağlı olduğundan, türevlerinin kaybolma durumuna gradyanların yok olması (vanishing gradients), aşırı yükselme tehlikesine ise gradyanların patlaması (exploding gradients) durumu denir [95].



Şekil 4.4. RNN modelinin yapısı.

#### 4.4.3. Uzun Kısa Süreli Bellek-Long Short Term Memory (LSTM)

LSTM metodu, basit RNN mimarisinin karşılaştığı gradyanların yok olması ve patlaması problemlerine çözüm getirmekte ayrıca basit RNN mimarisinden genellikle daha iyi sonuçlar verdiği için son yıllarda oldukça tercih edilen bir yaklaşımdır. Şekil 4.5.'te LSTM hücresinin içyapısı görülmektedir [96,97].



Şekil 4.5. LSTM modelinin yapısı.

**$c_t$  (Cell State)** : Bilginin kesintisiz olarak bir hücreden diğer hücreye aktığı kanaldır. Art arda LSTM hücrelerinin sıralandığı düşünülürse hücreler arası bilgi akışı bu yolla sağlanmaktadır.

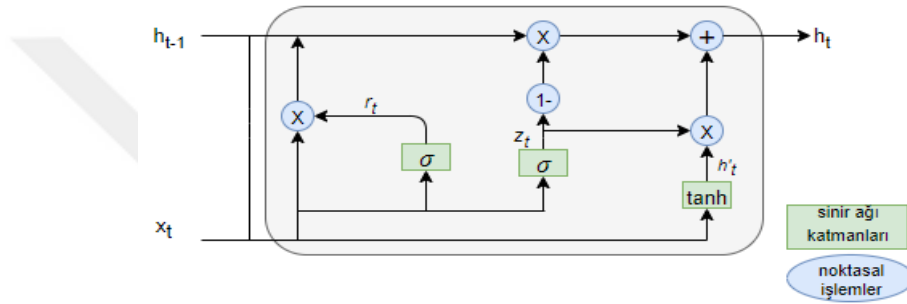
**$f_t$  (Forget Gate)** : LSTM hangi bilginin unutulup cell stateler ile diğer hücrelere aktarılmayacağına bu kapı ile karar verir. Önceki durumdan gelen bilgi ile girdi olarak gelen bilgiye sigmoid işlemi uygulanarak unutulacak bilgiye karar verilir.

**$i_t$  (Input Gate)** : Bu kapı ile hangi bilgelerin cell state'ler ile saklanıp diğer hücrelere aktarılacağına karar verilir. Ayrıca bu adımdaki işlemler de tamamlandıktan sonra cell state güncellenir ve bir sonra ki hücreye aktarılmak üzere hazır hale gelir.

**$o_t$  (Output Gate)** : Output değerinin ne olması gerektiğine bu kapı ile karar verilir. Ancak cell state 'dan gelen tüm değer Output olarak verilmez. Bunun için cell state  $\tanh$  işleminden geçirilerek bir filtre uygulanmış olur [98-101].

#### 4.4.4. Kapılı Tekrarlı Birim - Gated Recurrent Unit (GRU)

GRU, en çok kullanılan bir çeşit LSTM türüdür. Şekil 4.6.'da yer alan GRU yapısı incelendiğinde en önemli değişikliğin cell state'in kaldırılması olduğu görülmektedir. İkinci önemli değişiklik ise iki farklı kapı olan forget gate ve input gate kaldırılarak update gate şeklinde birleştirilmiştir. Bu iki değişiklik ile LSTM yapısına göre daha basit bir yapısı vardır. GRU modelinde kullanılan eşitlikler, denklem (4.5 - 4.8) de görülmektedir [102-105].



Şekil 4.6. GRU modelinin yapısı.

$$z_t = (W_z[h_{t-1}, x_t]) \quad (4.5)$$

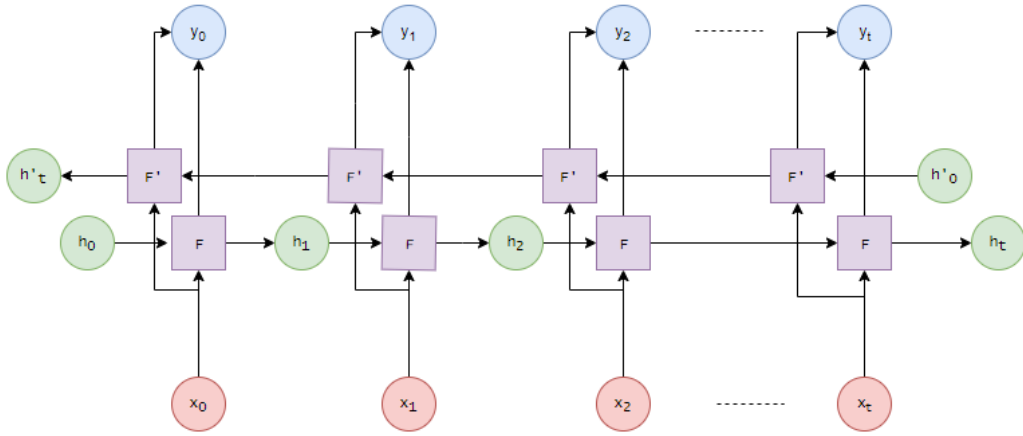
$$r_t = (W_r[h_{t-1}, x_t]) \quad (4.6)$$

$$h'_t = \tanh(W[r_t * h_{t-1}, x_t]) \quad (4.7)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * h'_t \quad (4.8)$$

#### 4.4.5. İki Yönlü RNN -Bidirectional RNN (BRNN)

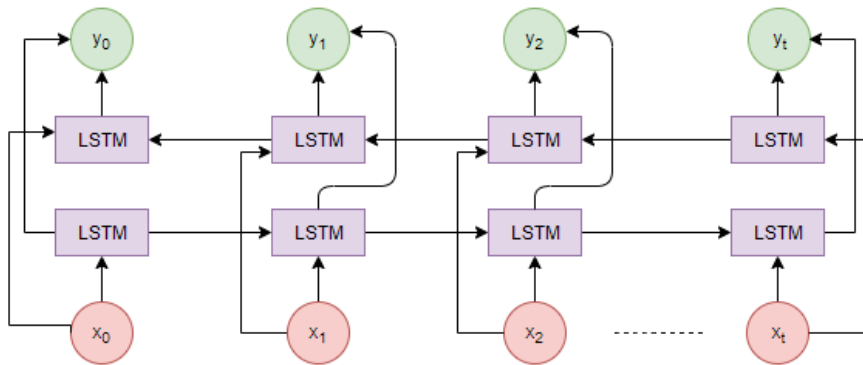
İki yönlü RNN mimarisi, basit RNN yapısında t zamanındaki bir çıktı değeri sadece tekrarlı ağlara bağlı olan hücrelere değil aynı zamanda gelecekteki hücrelere de bağlı olabileceği prensibine dayanmaktadır. Örneğin bir dizide (sequence) yer alan eksik bir kelime tahmin edilmek istendiğinde kelimenin hem sağındaki hem de solundaki içeriğe bakılması istenir. İki yönlü RNN yapısı için üst üste sıralanmış iki basit RNN mimarisi denilebilir. Şekil 4.7.'de BRNN mimarisi görülmektedir. Çıktı her iki RNN'in gizli durumuna göre (hidden state) hesaplanır [106-108].



Şekil 4.7. BRNN modelinin yapısı

#### 4.4.6. İki Yönlü LSTM - Bidirectional LSTM (BLSTM)

İki yönlü LSTM, özellikle dizi (sequence) yapısındaki verilerin sınıflandırılması işlemlerinde model performansını daha da artırabilmek amacıyla geliştirilmiş LSTMlerin bir türüdür. Giriş dizisinin tüm zaman aralıklarının bulunduğu problemlerde, BLSTM yapısında giriş dizisindeki bir LSTM yerine çift LSTM eğitilir.  $x_0, x_1, x_2, \dots, x_n$  değerleri giriş vektörlerini temsil etmektedir [109]. Burada eğitilen LSTM'lerde ilk LSTM' in girdisi giriş dizisinin kendisi ikincisi ise ilk dizinin ters çevrilmiş şeklidir. Bu sayede problemin daha hızlı ve daha iyi bir eğitim yapılması hedeflenmektedir. Eğitim işlemi tamamlandıktan sonra LSTM hücrelerinden gelen değerler birleştirilir ve sonuç çıktısı üretilir. Şekil 4.8.'de BLSTM modelinin yapısı görülmektedir [110].



Şekil 4.8. BLSTM modelinin yapısı.

#### **4.5. Derin Sinir Ağlarında Hiper-parametre Ayarlamaları**

Hiper-parametreler, eğitim işlemi başlamadan önceden belirlenmiş bir değere sahip olan ve manuel olarak yapılan parametrelerdir ve üzerinde çalışılan göreve, donanımına ayrıca kullanılan veri setlerine göre ciddi farklılıklar göstermektedirler. Bu nedenle en uygun hiper-parametre değerlerinin seçilmesi önemli bir problemdir. Genellikle hiper-parametrelerin seçimi derin öğrenme modelini tasarlayan kişilerin deneyim ve sezgilerine göre değişim göstermektedirler. Ancak son yıllarda yapılan çalışmalarla, en uygun değerlerin seçilebilmesi için farklı yöntemler geliştirilmektedir [111-113].

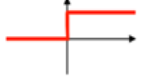
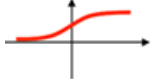
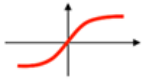
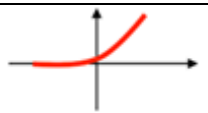
##### **4.5.1. Mini-batch Boyutu**

Derin öğrenme modellerinin eğitim sürecinde, üzerinde çalışılan verilerin aynı anda işlenmesi zaman ve bellek kullanımı açısından oldukça maliyetlidir. Çünkü eğitim aşamasının her iterasyonunda geriye yayılım işlemi ağda gradyan hesaplanması yapılmakta ve ağırlık değerleri güncellenmektedir. Bu işlemde, işlenen veri miktarı ne kadar fazlaysa doğal olarak işlem yükü de o derecede fazla olmaktadır. Bu sorunun aşılması için veri setinin tümünün aynı anda işlenmesi yerine veriler küçük gruplara ayrılarak, eğitimin bu gruplar üzerinde yapılmasıdır. Bu yönteme mini-batch denilmektedir [114].

##### **4.5.2. Aktivasyon Fonksiyonları**

Bir sinir ağının performansını, ağ modellenirken tercih edilen aktivasyon fonksiyonu oldukça etkilemektedir. Yapay sinir ağında en sık kullanılan aktivasyon fonksiyonları, denklemleri ve grafikleri Çizelge 4.1’te gösterilmiştir [115-117].

Çizelge 4.1. Aktivasyon fonksiyonları

| AKTİVASYON FONKSİYONU         | DENKLEM                                                                            | ARALIK              | GRAFİK                                                                              |
|-------------------------------|------------------------------------------------------------------------------------|---------------------|-------------------------------------------------------------------------------------|
| Doğrusal Fonksiyon            | $f(x) = x$                                                                         | $(-\infty, \infty)$ |  |
| Basamak Fonksiyonu            | $f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$ | $\{0,1\}$           |  |
| Sigmoid Fonksiyonu            | $f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$                                          | $(0,1)$             |  |
| Hiperbolik Tanjant Fonksiyonu | $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$                              | $(-1,1)$            |  |
| ReLu                          | $f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ | $[0, \infty)$       |  |
| Softmax                       | $f(x) = \text{softmax}(x_i) = \frac{e^{x_i}}{\sum e^{x_j}}$                        | $(0,1)$             |  |

#### 4.5.3. Öğrenme Hızı

Öğrenme hızı, bir derin sinir ağının önemli parametrelerinden biri olduğu hatta bazı araştırmacılara göre en önemli parametresidir. Sinir ağı mimarilerinde öğrenme hızı parametresi genellikle başlangıçta 0.01 veya 0.001 değerleri atanmaktadır. Öğrenme hızı parametresi sabit bir değer alabildiği gibi, adaptif yöntemler ile zamana bağlı olarak değişen değerleri de alabilir. Ancak diğer hiper-parametrelerde olduğu gibi öğrenme hızında da en iyi performans için kesin veya belli bir değer yoktur. Bunun için genelde ağı eğitim aşamasındaki performansı takip edilerek karar verilmelidir [118].

#### 4.5.4. Optimizasyon Algoritmasının Seçimi

Optimizasyon algoritmaları, üretilen çıktı ile gerçek değer arasındaki farkın yani bir başka deyişle hata değerinin minimum olabilmesi için geliştirilmiş yöntemlerdir. Derin sinir ağlarında da eğitim süreci temelde bu prensibe dayandığı için bu işlemde bir optimizasyon problemidir [119]. Sinir ağlarında en çok tercih edilen optimizasyon algoritması gradyan iniş (gradient descent) algoritmasıdır. Derin öğrenme modellerinde de en çok kullanılan diğer algoritmalar ise stochastic gradient descent, adagrad, adadelata, adam, adamax gibi

optimizasyon algoritmaları kullanılmaktadır. Bu algoritmalar arasında başarımlar ve hız bakımından farklılıklar bulunmaktadır [120].

#### **4.5.5. Tur (Epoch) Sayısı**

Eğitim sürecinde daha öncede bahsedildiği gibi modelin üzerinde çalıştığı verilerin tümü aynı anda öğrenme işleminde kullanılmaz. Mini-batch boyutunda veriler eğitilir ve modelin performansı doğrulama verileri ile sınanır. Elde edilen doğruluk değerlerine göre geri yayılım işlemi ile ağırlıklar güncellenir. Daha sonra yeni eğitim kümesi ile eğitim işlemi tekrar yapılır ve ağırlıklar tekrar güncellenir. Yani modelin ileri ve geri yayılım işlemlerinin gerçekleştiği eğitim adımlarının her birine tur denilmektedir [121]. Ancak bu parametre seçiminde karşılaşılabilecek iki durum söz konusudur. İlkinde, modelde tercih edilen tur sayısı parametresi gereğinden fazla bir değer seçildiğinde model gereksiz detayları bile ezberleyerek aşırı öğrenme buda modelin başarımlarını olumsuz etkileyecektir. İkinci durumda ise tercih edilen değer gereğinden az olmasıdır. Bu durumda da eksik öğrenme denilen durum gerçekleşecek bu da modelin performansını düşürecek diğer bir durum olacaktır. Model oluşturulup eğitim süreci başlatıldıktan sonra beklenen durum, ilk turlarda başarımlar değerinin düşük olması ancak tur sayısı arttıkça başarımlarında artmasıdır. Belli bir tur sayısından sonra modelin öğrenme sürecinin oldukça yavaşladığı görülecektir. Bu tur sayıları parametrenin seçimi için ideal değerler olabilmektedir [122].

#### **4.5.6. Seyreltme (Dropout) Değeri**

Dropout tekniği, tur sayısı parametresinin anlatıldığı kısımda yer alan genellikle modelin eğitim sürecinde aşırı öğrenmesi veya ezberlemesi gibi olumsuz durumların engellenmesi için kullanılmaktadır. Çalışma prensibinde bazı nöronların belirli bir eşik değere göre rastgele şekilde pasif hale getirilmesi yatmaktadır. Böylece aşırı öğrenme durumu ortadan kalkmaktadır [123].

Derin öğrenme uygulamalarında genellikle dropout değeri olarak 0.5 tercih edilmesine rağmen rastgele eleme tekniği de kullanılmaktadır. Ancak diğer hiper-parametrelerde olduğu gibi üzerinde çalışılan probleme ve veriye göre değişkenlik göstermektedir [124].

#### **4.5.7. Katman Sayısı ve Nöron Sayısı**

Derin öğrenme modellerini klasik yapay sinir ağlarından en önemli farkının özellikle de karmaşık problemlerin çözümünde başarılı sonuçlar elde etmesinin katman sayısından geldiği bilinmektedir. Katman sayısı arttıkça modelin daha iyi öğrendiği görülmüştür. Ancak bu

durumda model, eğitim sürecinde üzerinde çalıştığı eğitim verisini aşırı öğrenerek ezberlemektedir. Bunun tersi durumda ise eksik öğrenme durumu ile karşılaşmaktadır. Bu nedenle önemli bir parametredir [125].

#### 4.5.8. Hata ve Kayıp Fonksiyonu

Sinir ağlarındaki eğitim sürecinde beklenen çıktı ile üretilen çıktı arasındaki farka hata denilmektedir.

$$J(w) = p - \hat{p} \quad (4.9)$$

Bu hatayı hesaplamak için kayıp fonksiyonu denilen 4.9.'da gösterilen eşitlikte  $J$  fonksiyonu ile gösterilen fonksiyon kullanılmaktadır. Derin sinir ağlarında çeşitli kayıp fonksiyonları tercih edilmektedir. Farklı kayıp fonksiyonları, aynı tahmin için farklı hatalar vermekte, bu nedenle modelin performansı üzerinde önemli etkileri olmaktadır. En yaygın kullanılan kayıp fonksiyonlarından biri, gerçek değer ile öngörülen değer arasındaki farkın karesini hesaplayan ortalama karesel hatadır. Bununla birlikte farklı kayıp fonksiyonları da, regresyon ve sınıflandırma gibi farklı türdeki problemleri çözebilmek için kullanılmaktadır [126,127].

### 4.6. Metin Verileri İçin Derin Öğrenme

Metinler, dizi tipindeki verilerin en yaygın biçimlerinden biridir. Bir karakter dizisi veya bir kelime dizisi olarak düşünülebilir, ancak en yaygın şekli kelime seviyesinde yapılan çalışmalardır. Derin öğrenmenin doğal dil işlemede uygulanması, örüntü tanımlamanın kelimelere, cümlelere ve paragraflara uygulanması şeklinde düşünülebilir [128].

Derin sinir ağlarında en kritik noktanın, ağlara input olarak verilen girdilerin olduğu söylenebilir. Bilinmektedir ki derin öğrenme modelleri de sinir ağlarında olduğu gibi ham metin verisini girdi olarak alamazlar. Bu aşamadaki en kritik nokta ise metinlerin vektör haline yani sayısal tensörlere dönüştürülmesi işlemidir [129,130].

#### 4.6.1. Kelime Temsillerinin Kullanımı

Kelime ve vektörleri birlikte kullanma konusunda en çok tercih edilen yöntem kelime vektörü yöntemidir. Bu yöntem literatürde kelime gömmesi (word embedding) olarak tanımlanmaktadır. Kelime vektörleri iki şekilde üretilir.

- İlk yöntem, kelime vektörleri değerlerini ana görev ile birlikte öğrenmektir. Bu yöntemde vektörlerin ağırlıkları ilk başta rastgele küçük değerler ile veya sıfırlar ile başlar.

- Diğer yöntem ise başka bir görevde eğitilen kelime vektörleri, ana görevde kullanılan modele yüklenir. Bu yöntem ise önceden eğitilmiş kelime vektörleri olarak adlandırılır. Bu tez çalışmasında da önerilen yöntemler bölümünde önceden eğitilmiş kelime vektörleri ile deneysel çalışmalar yapılarak sonuçları analiz edilmiştir [131].

#### 4.6.2. Kelime Vektörleri

Kelimeye vektör ilişkilendirmenin en kolay yolu rastgele bir vektör seçmektir. Üretilen vektörlerin arasındaki geometrik ilişkinin kelimelerin anlamlarında da yakın olması beklenmektedir. Yani bir başka deyişle kelime vektörlerinin insan dilinin vektör uzay modelinde eşlenmesidir. Bu durumda yakın anlamlı kelimelerin vektör uzayında vektörlerinde birbirlerine yakın olması beklenmektedir. Bu bilgiler ışığında iki kelime vektörü arasındaki geometrik mesafe, ilgili kelimeler yakın anlamlı ise yakın olmalı, yakın anlamlı değilse uzak olmalıdır. Örneğin büyük bir derlemden elde edilmiş kelime vektörleri, vektör uzayına yerleştirildikten sonra haftanın günleri olan *pazartesi*, *salı*, *çarşamba* gibi kelimelerin vektörleri adeta bir küme gibi birbirlerine daha yakın bulunurken, bu kelimelerle bazı hayvan isimleri uzayda birbirlerine uzakta bulunurlar.

#### 4.6.3. Önceden Eğitilmiş Kelime Vektörleri

Önceden eğitilmiş kelime vektörleri daha öncede bahsedildiği üzere, başka bir görev üzerinde eğitilmiş kelime vektörlerinin ana görev üzerinde çalışan modele yüklenmesidir. Bu yöntem genellikle az miktarda eğitim verisi ile çalışıldığında kullanılmaktadır [132].

Kelime vektörlerinin üzerinde çalışılan veriler yerine, dilin genel yapısını kavramış, kullanışlı özellikler sunan değerleri hesaplanmış kelime vektörlerinin yüklenmesi modelin performansını geliştirebilecek önemli bir avantajdır. Çünkü bir görev üzerinde çalışılan veriler müşteri-satış, bir durumu beğenip-beğenmeme gibi belirli bir konsepti olan verilerdir. Bu veriler üzerinde eğitilen kelime vektörleri genel yapıda vektörler değil daha spesifik vektörlerdir. Ancak bir derlemden öğrenilen vektörlerin olabildiğince genel olması istenen bir durumdur [133,134].

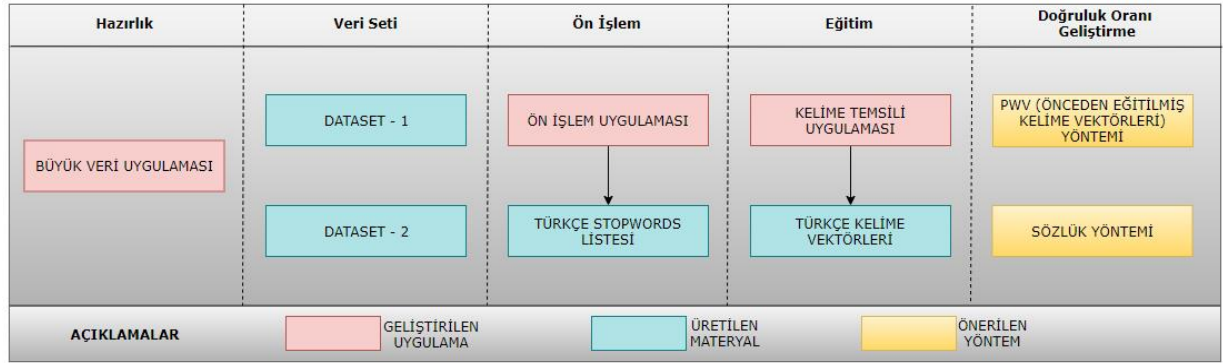
Denetimsiz olarak yoğun, düşük boyutlu kelime vektörü uzayı hesaplama fikri ilk olarak Bengio ve arkadaşları tarafından 2000’li yıllarda çalışılmaya başlanmıştır. Fakat günümüzde de hala en popüler kelime temsil yöntemi olan Mikolov ve arkadaşları tarafından 2013 yılında Google da geliştirilen Word2Vec yöntemidir [135].

Önceden eğitilmiş kelime vektörleri eğitildikten sonra bir çeşit veri tabanında tutulur ve buradan yüklenerek vektörlerin ağırlıklarının saklandığı embedding katmanında kullanılabilir. Önceden eğitilmiş kelime vektörlerinin kullanılması, temelde görüntü işlemede önceden eğitilmiş ağların kullanılmasına benzemektedir.



## 5. GELİŞTİRİLEN UYGULAMALAR

Tezin bu bölümünde yapılan uygulamalar verilmiş ve elde edilen sonuçlar grafiklerle görselleştirilerek analiz edilmiştir. Tez çalışması kapsamında 2 Türkçe veri seti üretilmiş, 3 uygulama gerçekleştirilmiş, bu uygulamalar sonucunda Türkçe stopwords listesi ve Türkçe kelime vektörleri oluşturulmuş ve deneysel çalışmalar ile elde edilen doğruluk skorlarını geliştirmek amacıyla 2 özgün yöntem önerilmiştir. Bu işlemler Şekil 5.1.'de görülmektedir.



Şekil 5.1. Tez çalışmasının genel yapısı

### 5.1. Uygulama 1: Apache Spark ile Büyük Veri Uygulaması

Bu tez çalışmasında daha önceki bölümlerde de bahsedildiği üzere öncelikle hangi ortamın kullanılacağına karar verilmiştir. Tez çalışması için üretilen veri miktarının büyüklüğü ve derin öğrenme yaklaşımı ile ulaştığı başarı sonuçlar dikkate alındığında büyük veri teknolojilerine ihtiyaç duyulduğu tespit edilmiştir. Bunun için uygun bir platform olarak Apache Spark teknolojisi tercih edilmiştir. Ancak bunun iyi bir seçim olup olmadığının görülebilmesi için spam ve spam olmayan maillerin sınıflandırıldığı bir metin sınıflandırma uygulaması geliştirilmiştir.

#### 5.1.1. Uygulamanın Amacı

Spam, kelime olarak istenmeyen anlamına gelmektedir. Özellikle bilişim dünyasında karşımıza “spam mail” kavramı ile sık sık çıkmaktadır. Başka bir ifadeyle kişinin bilgisi ve isteği dışında kendisine gönderilen postalara “spam mail” denilmektedir.

Spam mailler e-posta teknolojisinin kullanılmaya başlandığı ilk zamanlardan itibaren gönderilmeye başlanmıştır. Spam maillerin en büyük zararı ise bilinenin aksine alıcının göndericiden daha fazla maliyetle karşılaşmasıdır. Buna örnek olarak American Online ve

CyberPromotions kuruluşları arasındaki dava gösterilebilir. American Online medya şirketi CyberPromotions kuruluşundan günde 1,8 milyon spam mail aldığını tespit etmiştir. Bir e-posta kullanıcısının bir spam maili tespit edip silmesinin yaklaşık 10 saniyesini aldığı düşünülse bile böyle bir durumda yaklaşık 5000 saat sürmektedir. Buna karşın T1 hattı kullanılan spam kaynağı bunun için günde yaklaşık 100\$ ödemektedir [136].

Bu uygulama ile spam ve spam olmayan maillerden oluşan bir veri seti üzerinde Apache Spark teknolojisinin gücü ve verimliliğinin test edilmesi amaçlanmıştır.

### 5.1.2. Uygulamanın Geliştirilmesi

Çalışmada anlatılan büyük veri teknolojilerinden Spark ve Spark'ın makine öğrenmesi (MLlib) kütüphanesi kullanılarak bir sınıflandırma projesi yapılmak istenmiştir. Bu çalışma için kaggle platformunda erişime açık şekilde paylaşılmış, yaklaşık 10,5 milyon satır kayıt içeren, İngilizce ile yazılmış elektronik postalarından oluşan bir veri seti kullanılmıştır. Veri setinde öncelikle gürültü temizliği yapılarak alakasız veriler temizlenmiştir. Kullanılan veri setinde yer alan elektronik postalardan bir tanesi ve kayıtların formatı Şekil 5.2.'de ki gibidir.

```
From craig@deersoft.com
Return-Path: <craig@deersoft.com>
Delivered-To: yyyy@localhost.netnoteinc.com
Received: fromlocalhost (localhost [127.0.0.1])
Date: Thu, 22 Aug 2012 22:53:44 -0700
Subject: Re: resultsforgiantmass-check (pew)
Cc: yyyy@example.com (Justin Mason),
"Malte S. Stretz" <msquadrat.nospamplease@gmx.net>,
To: Scott A Crosby<scrosby@cs.rice.edu>
From: "CraigR.Hughes" <craig@deersoft.com>
Content: "I neverclaimed it couldlearn *all* combinatorial
possibilities, but it certainly can learnsome."
```

Şekil 5.2. Verisetindeki mail formatı

Uygulamada Apache Spark ile metin sınıflandırma teknikleri kullanılarak Spam Mail Tespiti uygulaması geliştirilmek istendiği için gürültülü veriler temizlendikten sonra bu uygulama için gerekli olmadığı düşünülen sütunlar çıkarılarak sadece mail içerikleri korunmuştur. Çizelge 5.1.'de Sınıflandırma etiketi olarak Spam mailler için "1", spam olmayan mailler için ise "0" değeri verilmiştir.

Çizelge 5.1. Dönüştürülen veri seti

| Sınıf | Mail İçeriği                                                                                                                                                                |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0     | "I never claimed it could learn *all* combinatorial possibilities, but it certainly can learn some"                                                                         |
| 1     | "You have an important customer service announcement. Your gift is ready. Just Call FREEPHONE 0800 542 0825 now and take your surprise gift."                               |
| 1     | "Thanks for your subscription to game of warriors your application will be updated £5/month Please confirm by replying YES or NO. If you reply NO you will not be updated." |

Geliştirilen uygulamada Çizelge 5.1.'deki hale dönüştürülmüş veri seti kullanılmıştır. Spark teknolojisi kullanılarak veri analizi yapılabilmesi için önce Spark Context denilen yapının oluşturulması gereklidir. Bu yapı verilere ulaşılmasını sağlayan bir köprü görevi görmektedir. Spark teknolojisinin diğer bir önemli yapısı ise tez çalışmasında büyük veri bölümünde bahsedilen RDD yapısıdır. Bu yapı sayesinde context nesnesi ile veri setinden çekilen veriler RDD ile işlenebilir hale getirilir. Bu çalışmada spam mailler (1) ve spam olmayan (0) mailler için ayrı ayrı RDD yapıları oluşturulmuştur.

```
JavaRDD<String>spamContent = spam.map(newFunction<String, String>()
publicStringcall(String line) throws Exception
returnline.split(" ")[1];
```

Şekil 5.3. RDD Yapısı

Şekil 5.3.'te gösterilen RDD yapısından da anlaşılacağı gibi RDD ile text şeklindeki okunabilir hale getirilmiş veriler "FlatMap" işlemi ile tırnak işareti içerisinde yer alan string ifadeyi içerik olarak tanımlamıştır. Analiz ortamında önce veriler, ardından cümleler elde edildikten sonra bu kez metinler kelimeler arasındaki boşluklar dikkate alınarak vektör yapısı haline dönüştürür. Vektör dönüşümü yapılan kelimeler ise Spark'ın makine öğrenmesi kütüphanesinde yer alan Naive Bayes yöntemi ile modellenmiştir [137].

Çizelge 5.2. Karışıklık matrisi

|                | SPAM (1)       | SPAM DEĞİL (0) |
|----------------|----------------|----------------|
| SPAM (1)       | 1097602 (TP)   | 796684 (FP)    |
| SPAM DEĞİL (0) | 98785 (FN)     | 8497160 (TN)   |
| Doğruluk Oranı | <b>%91.463</b> |                |

Veri setinde yer alan 10.490.231 kayıt alınmış, bu verilerden %66'lık kısma karşılık gelen 6.923.552 adet veri eğitim için kullanılmıştır. Geriye kalan %33'lük oranı kapsayan 3.461.776 veriden ise modelin test aşamasında faydalanılmıştır. Naive Bayes yöntemi ile %91.463 gibi yüksek bir başarımlık oranı yakalandığı görülmüş geliştirilen model ile 9.594.762 adet veri doğru şekilde sınıflandırılmıştır. Karışıklık matrisi ise Çizelge 5.2.'de ki gibi oluşmuştur.

$$Doğruluk = \frac{(TN+TP)}{(TN+TP+FP+FN)} \quad (5.1)$$

$$Hassasiyet = \frac{TP}{TP+FP} \quad (5.2)$$

$$Özgünlük = \frac{TN}{TN+FP} \quad (5.3)$$

$$F\ skor = 2 \frac{Hassasiyet \cdot Özgünlük}{Hassasiyet + Özgünlük} \quad (5.4)$$

Denklem 5.1, 5.2 ve 5.3' te yer alan formüllerden yararlanılarak Çizelge 5.3.' teki gibi Doğruluk, Hassasiyet Özgünlük ve F-Skor değerleri hesaplanmıştır.

Çizelge 5.3. Başarım oranları

| Doğruluk      | Hassasiyet   | Özgünlük     | F skor       |
|---------------|--------------|--------------|--------------|
| <b>91.463</b> | <b>0.917</b> | <b>0.988</b> | <b>0.951</b> |

Veri setinde bulunan yaklaşık 10 milyon kayıt daha iyi bir şekilde analiz edilebilmesi için Çizelge 5.4.'te olduğu gibi 1,3,5,7 ve 10 milyon olmak üzere 5 parçaya bölünmüş ve bu parçalar üzerindeki çalışma süreleri incelenmiştir. Analizler sonucunda belirgin şekilde görülmüştür ki Apache Spark veri miktarı arttıkça genellikle veri madenciliğinde, veri analizlerinde en çok tercih edilen ortamlardan çok daha performanslı şekilde çalışmaktadır. Veri setinin yaklaşık olarak 1/3 'lük kısmına denk gelen bir boyutu (3 milyon) 1 dakikanın

altında işlemiştir. Veri setinin yarısı işlenmek istediğinde ise Weka bu veriyi işleyememiş ve bellek sorunları ile karşılaşmıştır. Veri setinin tamamı analiz edilmek istendiğinde ise Spark'ın çalışma süresi 2,55dk olarak ölçülmüş, Weka ve Knime yazılımları veriyi işleyememiş Rapid Miner ise yaklaşık 41 dakika da veriyi işleyebilmiştir.

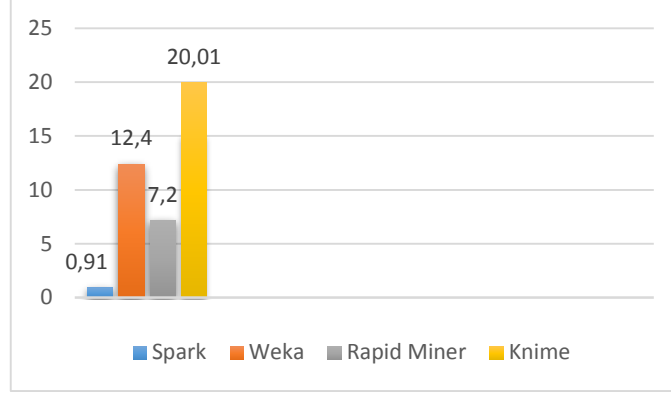
Uygulamada Apache Spark'ın performansını kıyaslayabilmek için Veri analizi, İstatistik, Veri Madenciliği alanlarında en çok kullanılan yazılımlar kullanılmıştır. Yapılan bir çalışmada ise araştırmacılar tarafından bu alanlarda en çok tercih edilen yazılım % 65.08 ile Weka olmuştur. Ardından % 46.03 ile Rapid Miner ve % 26.98 ile Knime gelmektedir [138]. Yapılan bu çalışma ile araştırmacılar tarafından en çok tercih edilen platformlara karşı Apache Spark'ın üstünlüğü net olarak ortaya çıkmaktadır.

Ayrıca bilinmelidir ki bu çalışmada kullanılan veri seti her ne kadar milyonlara varan kayıt sayısında olsa da, günümüzde elde edilen ve işlenmesi gereken veri miktarı düşünüldüğünde asla “Büyük Veri” denilemeyecek kadar da az miktarda bir veri boyutundadır. Bu çalışmayla büyük veriye oranla bu kadar az bir verinin bile işlenme süreleri göz önüne alındığında artık mevcut araçların ne kadar yetersiz kaldığı, büyük veri araçlarına neden ihtiyaç duyulduğu ve bu teknolojileri kullanmanın kaçınılmaz olduğu açıkça ortaya çıkmaktadır.

Çizelge 5.4. Çalışma süreleri (dk)

| KAYIT SAYISI    | SPARK | WEKA  | R.MINER | KNIME | MATLAB |
|-----------------|-------|-------|---------|-------|--------|
| <b>1Milyon</b>  | 0,46  | 8,03  | 4,04    | 19,01 | 10,03  |
| <b>3Milyon</b>  | 0,98  | 23,53 | 11,26   | 28,23 | 29,31  |
| <b>5Milyon</b>  | 1,28  | -     | 19,31   | 39,36 | 51,25  |
| <b>7Milyon</b>  | 2,09  | -     | 29,12   | 53,05 | -      |
| <b>10Milyon</b> | 2,55  | -     | 40,23   | -     | -      |

Platformların ortalama çalışma sürelerinin hesaplanması işlemi ise aynı veri miktarı (2 Milyon kayıt) ve eşit sayıda (5 kez) çalıştırma senaryosu ile yapılmış ve ortaya çıkan sonuçlar Şekil 5.4.' te verilmiştir. Sonuçlar aynı bilgisayar ve donanım üzerinde çalıştırılarak elde edilmiştir.



Şekil 5.4. Ortalama çalışma süreleri (dk)

### 5.1.3. Uygulamanın Sonuçları

Hızla gelişen bilişim teknolojileriyle birlikte devasa boyutlarda veri miktarına çok müthiş bir hızla ulaşıldığı herkes tarafından kabul edilen bir gerçektir. Bu kadar çabuk büyüyen verinin ise hem saklanması hem de işlenmesi ise veri bilimcilerinin karşılarında duran en büyük problemdir.

Uygulamada en önemli büyük veri teknolojilerinden biri olan Spark ile bir metin sınıflandırma işlemi yapılarak yeni teknolojiler ortaya çıkana kadar en çok tercih edilen yazılımlarla performansının ölçülmesi istenmiştir. Bunun içinde gelişen internet dünyası ile çağımızın önemli bir sorunu haline gelen Spam maillerin tespit edilmesi uygulamasının geliştirilmesi, problem olarak seçilmiştir. Oldukça hızlı bir teknoloji olan Spark yine sunduğu makine öğrenmesi kütüphanesi ile (MLlib) uygulamanın geliştirilmesi aşamasında müthiş bir kolaylık sağlamıştır. Uygulamada yöntem olarak ise makine öğrenmesi algoritmalarından denetimli öğrenme temellerine dayanan Naive Bayes yöntemi seçilmiştir.

Devasa boyutlara ulaşan veri miktarlarıyla klasik yöntemlerin ve araçların artık geçerliliklerini yitirdikleri aşikârdır. Hatta veri işlemek bir yana verinin saklanması bile artık sadece büyük veri teknolojileriyle mümkün görülebilmektedir. Yapılan çalışma ile ortaya çıkmıştır ki büyük verinin en yeni teknolojilerinden olan Apache Spark ile diğer yazılımların arasında oldukça büyük bir performans farkı bulunmaktadır.

Ayrıca yapılan bu çalışma, büyük veri sayılamayacak kadar bir veri miktarının dahi en gelişmiş ve en çok tercih edilen yazılımlar tarafından işlenememesi, büyük veriyi işlemek için neden yeni teknolojilere ihtiyaç duyulduğunu ve gerekliliğini açıkça ortaya çıkarmıştır. Bu nedenle tez çalışmasında büyük veri teknolojilerinden Apache Spark teknolojisi tercih edilmiş ve tez çalışmasının bu kısımdan sonra gerçekleştirilen uygulamalarda kullanılmıştır.

## 5.2. Materyal

Tezin bu bölümünde, çalışma kapsamında üretilen Türkçe veri setleri ile ilgili detaylar yer almaktadır. Tez çalışmasının bundan sonraki bölümlerinde yer alan uygulamalarda, üretilen bu veri setleri kullanılmıştır.

### 5.2.1. Dataset-1

Daha öncede bahsedildiği üzere metin işleme hatta daha üst kapsamıyla doğal dil işleme alanında yapılan çalışmalar, büyük veri kavramının ortaya çıkmasıyla hatta bu verilerin çok büyük bir kısmının metin (text) verilerinden oluşması nedeniyle çok büyük önem kazanmış ve son yıllarda oldukça popüler çalışma alanları haline gelmiştir. Ancak bu konuda Türkçe diline özgü yapılan çalışmaların sayısına bakıldığında sayılarının yetersiz olduğu görülmüştür. Çalışmada kullanılan Türkçe derlem (korpus) şöyledir.

Hazırlanan derlem, “teknoloji”, “eğitim”, “tarih”, “Türkiye”, “ekonomi”, “spor” , “siyaset” gibi konu başlıkları ile sorgular yapılmıştır. Bu başlıklar ile alakalı olarak Wikipedia platformundan Türkçe makaleler çekilmiştir. Derlemin çeşitliliği ve genelliği açısından bu konu başlıkları tercih edilmiştir.

Bu çalışma kapsamında, toplamda 25.090.767 satır, 558.087.641 cümle ve 12.562.752.820 adet token haline gelmiş kelimedenden oluşan yaklaşık 80 gb boyutunda, bu alanda yapılmış çalışmalar içerisindeki en büyük derlemlerden biri oluşturulmuştur.

### 5.2.2. Dataset-2

Bu çalışmadaki ikinci veri seti ise Dataset-1 ile eğitilmiş kelime vektörlerinin derin sinir ağı modelleri ile birlikte kullanılarak Türkçe metin sınıflandırma problemi için başarımının test edilmesi amacıyla üretilmiştir. Bu kapsamda bir giyim firması hakkında müşterilerden ve satıcılardan gelen geri bildirimler toplanmıştır. Çağrı merkezleri ve internet sitesi aracılığıyla gelen mesajlar operatörler tarafından etiketlenmiştir. Böylece yaklaşık 1,5 milyon örnekten ve 10 sınıftan oluşan büyük ve kapsamlı bir veri seti oluşturulmuştur. Çizelge 5.5.’te Dataset-2’nin yapısı ve Çizelge 5.6.’da ise kategorileri görülmektedir.

Çizelge 5.5. Dataset-2'nin yapısı

| ID    | Yorum                                                                                                                                                  | Sınıf |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 00328 | Satın aldığım ürün büyük geldi küçük beden ile değiştirmek istiyorum                                                                                   | 103   |
| 00329 | Aldığım ürün için ödememi taksit seçeneği ile yaptım fakat detaylarımda göremiyorum, bilgi verirseniz sevinirim... teşekkürler                         | 106   |
| 00330 | Aldığımız Eteği çok beğendik teşekkürler kargo sorunsuzdu...                                                                                           | 107   |
| 00331 | merhaba,musterimiz 22051399 siparisine ait kargosu alıcı teslim almadığı için musterimiz bu konu hakkında en kısa sure icerisinde dönüş beklemektedir. | 104   |
| 00332 | Müşteri Teslimat Yapılmamış Sipariş: 001245                                                                                                            | 101   |
| 00333 | Ürün gayet güzel Kısa surede elime geçti teşekkürler                                                                                                   | 107   |

Çizelge 5.6. Dataset-2'nin kategorileri

| Kategori                | Sınıf |
|-------------------------|-------|
| Sipariş / Order         | 101   |
| Gecikme / Delay         | 102   |
| İade / Return           | 103   |
| Kargo / Cargo           | 104   |
| Ödeme / Payment         | 105   |
| Şikayet / Complaint     | 106   |
| Teşekkür / Appreciation | 107   |
| İptal / Cancel          | 108   |
| Ürün / Product          | 109   |
| Satıcı / Dealer         | 110   |

### 5.3. Uygulama 2: Önişlem Uygulaması

Tez çalışması kapsamında hazırlık aşamasında bir önişlem uygulaması yapılmıştır. Çalışmanın amacı, geliştirilmesi ve sonuçlar ilerleyen bölümde sunulmuştur.

#### 5.3.1. Uygulamanın Amacı

Metin işleme çalışmalarında önişlem aşaması, çok önemli bir görev ve uygulamanın başarımını doğrudan etkileyen kritik bir noktadır [139]. Bu nedenle, önişlem çalışmalarına önem gösterilmiş ve Türkçe için özgün bir ön işlem uygulamasının geliştirilerek başarımın geliştirilmesi amaçlanmıştır.

### 5.3.2. Uygulamanın Geliştirilmesi

Bir önceki başlıkta tanımlanmış problem doğrultusunda, tez çalışmasında uygulanan önışlem çalışmaları aşağıda yer almaktadır.

- Metin dosyası içerisindeki metinler okunarak, nokta işareti referans alınarak metinler cümlelere bölünmüştür.
- Cümlelerden harfler dışındaki tüm karakterler (sayılar, semboller, noktalama işaretleri) temizlenmiştir.
- Cümleler ise, aralarındaki boşluklar dikkate alınarak kelimelere ayrılmıştır.
- Token haline getirilmiş kelimeler, zemberek ortamı kullanılarak köklerine ayrılmıştır [140].

Literatürde etkisiz kelimeler (stopwords) olarak bilinen bazı Türkçe çalışmalarda durak veya durdurucu kelimeler olarak tanımlanmış esasen metin işleme çalışmalarına herhangi bir etkisi olmayan kelimelerdir.

Bu tür kelimeler çalışmanın başarımına pozitif bir etki yapmadığı gibi üzerinde çalışılan büyük hacimli derlemlerde ise ekstra iş yükü getirerek uygulamanın performansını olumsuz yönde etkilemektedir. Bazı doğal dil işleme araştırmacılarına göre etkisiz (stopwords) kelimelerin temizlenmesinin metin işleme çalışmalarının en önemli önışlem adımıdır. Ancak literatür incelemesi sırasında görülmüştür ki Türkçe için bilimsel yöntem ve metotlarla üretilmiş herhangi bir Türkçe etkisiz kelimeler (stopwords) listesi bulunmamaktadır. Tez çalışması kapsamında TF-IDF yöntemi kullanılarak bir önışlem uygulaması gerçekleştirilmiş ve uygulama sonucunda Türkçe etkisiz (stopwords) kelimeler listesi üretilmiştir.

Etkisiz kelimeler (stopwords) denilen etkisiz kelimeler listesinin oluşturulmasında tez çalışması için hazırlanmış Dataset-1 adı verilmiş derlem kullanılmıştır. Ayrıca etkisiz kelimeler listesinin oluşturulurken metin işleme konusunun anlatıldığı üçüncü bölümde değinilen TF-IDF yöntemi kullanılmıştır [72].

TF-IDF yönteminde öncelikle, 5.5 numaralı eşitlikte  $tf_{i,j}$  ile terim sıklığı yani TF değeri hesaplanmaktadır. Bu değer eşitlikte de görüldüğü üzere  $TF = \text{kelimenin doküman da geçme sayısı} / \text{dokümandaki kelime sayısı}$  ile hesaplanmaktadır.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \quad (5.5)$$

5.6 numaralı eşitlik kullanılarak IDF değeri hesaplanmıştır.  $IDF = \log ( toplam\ doküman\ sayısı / kelimeyi\ içeren\ doküman\ sayısı )$

$$idf(w) = \log \left( \frac{N}{df_t} \right) \quad (5.6)$$

TF-IDF matrisinin ağırlıklarının oluşturulması için ise 5.7 numaralı eşitlik kullanılmıştır.

$$w = tf * idf \quad (5.7)$$

Çizelge 5.7. yorumlanırsa, derlem içerisinde en yaygın kelimeler frekansları göz önüne alınarak hesaplanmıştır. TF-IDF değerleri tablodaki gibi hesaplandığında ağırlık değeri ne kadar sıfıra yakınsa o kelimenin derlem içerisinde o kadar yaygın (common word) bir kelime olduğu anlaşılmaktadır. Bu da derlem içerisinde kelimenin çok spesifik bir kelime olmadığını ortaya koymaktadır. Bu nedenle TF-IDF skoru 0 ile 0,1 aralığında olan kelimelerden etkisiz 250 kelime belirlenmiştir.

Çizelge 5.7. TF-IDF ağırlık matrisleri

| Kelime | TF                    | IDF                             | Ağırlık       |
|--------|-----------------------|---------------------------------|---------------|
| 'için' | 126215/4742527= 0,026 | $\log(387179 / 126215) = 0,486$ | TF*IDF =0,012 |
| 'de'   | 98689/4742527= 0,020  | $\log(387179 / 98689) = 0,593$  | TF*IDF =0,011 |
| 'da'   | 68652/4742527= 0,014  | $\log(387179 / 68652) = 0,751$  | TF*IDF =0,010 |
| 'bu'   | 44037/4742527= 0,009  | $\log(387179 / 44037) = 0,944$  | TF*IDF =0,008 |
| 've'   | 26249/4742527= 0,005  | $\log(387179 / 26249) = 1,168$  | TF*IDF =0,005 |

### 5.3.3. Uygulamanın Sonuçları

Daha önce de belirtildiği gibi bu çalışma için hazırlanan derlem içerisinde yaklaşık 12,5 milyar token kelime bulunmaktaydı. Etkisiz kelimeler listesinin oluşturulması ve uygulanması ile birlikte 848.908.720 adet kelime etkisiz kelimeler listesi içerisinde yer alan kelimelerden oluştuğu için model için hazırlanan derlem içerisinden elenerek temizlenmiştir.

## 5.4. Uygulama 3: Kelime Vektörlerinin Üretilmesi

Bir derlem içerisindeki metinlerin nasıl temsil edileceği metin işleme çalışmalarının en kritik noktalarından biridir. En basit tanımıyla, metinlerin sayısal ifadelerle dönüştürülmesine kelime temsili (word embedding) denilmektedir. Aynı metin farklı şekilde farklı sayısal

değerlere dönüştürülebilmektedir [141]. Bu tez çalışmasında da önışlem adımı uygulanmış kelimeler, kelime temsil yöntemleri kullanılarak dönüştürülmüştür.

#### **5.4.1. Uygulamanın Amacı**

Uygulamada kelime temsil yöntemleri kullanılarak kelimelerin aralarında ki anlamsal ilişkiler kaybolmadan kelimelerin sayısal formlara dönüştürülmesi hedeflenmiştir. Bu nedenle üçüncü bölümde anlatılan kelime temsil yöntemlerinden sinir ağı temelli kelime temsil yöntemleri kullanılmıştır.

Bu uygulamada, Türkçe içeriklerden oluşmuş bir derlem olan Dataset-1 üzerinde Word2Vec yöntemi ile her bir kelime için kelime vektörleri oluşturularak kelime benzerlikleri ve anlam benzerlikleri incelenmiş, ayrıca bulunan sonuçlar Glove yöntemi ile kıyaslanmıştır.

Uygulamanın temel amacı ise kelimelerin anlamsal ilişkilerinin başarılı bir şekilde yakalanarak kelime vektörlerine dönüştürülmesi ve ilerleyen uygulamalarda başka görevler ve veri setleri üzerinde kullanılmasıdır.

#### **5.4.2. Uygulamanın Geliştirilmesi**

##### *5.4.2.1. Vektörlerin Eğitilmesi*

Bu uygulamada daha öncede bahsedildiği gibi kelimelerin analizlerinin yapılmak üzere sayısallaştırılması maksadıyla Google tarafından Mikolov ve arkadaşları [69] tarafından geliştirilen Word2Vec yöntemi kullanılmıştır. Word2Vec yönteminde parametre seçimleri modelin performansını önemli ölçüde etkilemektedir. Yöntemde kullanılan bazı parametreler ve açıklamaları ile bu çalışmada geliştirilen modelde tercih edilen parametre değerleri Çizelge 5.8.'te verilmiştir. Kullanılan parametre değerleri literatürde yapılan çalışmalar incelenerek ve kullanılan değerlerin sonunda elde edilen sonuçlar deneme yanılma ile gözlemlenerek en uygun değer değerler tercih edilmiştir [142].

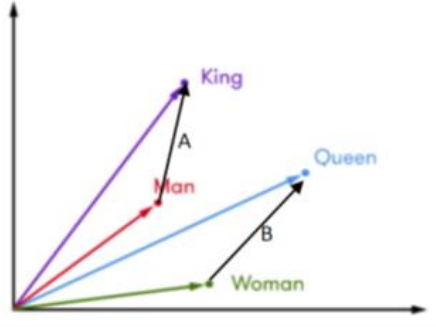
Çizelge 5.8. Word2Vec yönteminde kullanılan bazı parametreler

| Parametre        | Alabileceği Değer Aralığı                                             | Açıklama                                                                                                | Kullanılan Değer                               |
|------------------|-----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|------------------------------------------------|
| <i>sentences</i> | Model için oluşturulmuş, ön işlemden geçirilmiş kelimeler listesidir. | Modelin eğitilmesi için kullanılan veri kümesidir.                                                      | Derlem                                         |
| <i>size</i>      | Opsiyonel olarak integer değer alır.                                  | Kelime vektörlerinin boyutudur.                                                                         | 300                                            |
| <i>window</i>    | Opsiyonel olarak integer değer alır.                                  | Bir cümle içindeki mevcut ve tahmin edilen kelime arasındaki maksimum mesafedir.                        | 10                                             |
| <i>min_count</i> | Opsiyonel olarak integer değer alır.                                  | Parametre değerinden daha düşük frekanstaki tüm kelimeler yok sayılır.                                  | 10                                             |
| <i>workers</i>   | Opsiyonel olarak integer değer alır.                                  | Modeli eğitmek için çalışan thread sayısıdır. Çok çekirdekli makinelerle daha hızlı eğitim gerçekleşir. | 10                                             |
| <i>sg</i>        | 0 ve 1 değerlerini alır.                                              | Skip-gram yöntemiyle eğitim için 1, CBOW yöntemi için 0 değeri kullanılır.                              | Çalışmada iki yöntemde kullanılmıştır.(0 ve 1) |
| <i>alpha</i>     | Opsiyonel olarak float değer alır.                                    | Başlangıç öğrenme oranıdır.                                                                             | 0,025                                          |

Şekil 5.5'te görüldüğü gibi Mikolov ve arkadaşlarına göre, “king” ve “man” kelimelerinin vektörleri arasındaki uzaklık “queen” ve “woman” kelimeleri arasında ki uzaklığa eşit olmalıdır. Başka bir deyişle A vektörü B vektörüne eşit olmalıdır denilebilir. Mikolov ve arkadaşları bu eşitliği şöyle tanımlamışlardır [69].

$$\text{vector} ("king") - \text{vector} ("man") \approx \text{vector} ("queen") - \text{vector} ("woman")$$

Bu eşitlik Şekil 5.5.'te görselleştirilmiştir. Ayrıca bu gösterim yorumlandığında “king” kelimesinin vektörel değerinden “man” kelimesi çıkarılıp “woman” kelimesi eklenirse “queen” kelimesinin cevap olarak alınması beklenmektedir.



Şekil 5.5. Mikolov tarafından tanımlanmış kelime vektörlerinin gösterimi [69].

Bu uygulamada Word2Vec modeline ait CBOW ve Skip-gram yöntemleri Türkçe metinler üzerinde kıyaslanarak modelin performansı incelenmiştir. Word2Vec modelinde kelime vektörleri arasındaki benzerlik genellikle kosinüs benzerliği kullanılarak tespit edilmektedir. Kosinüs uzaklık formülü ile birlikte İbn-i Sina (Öklid) uzaklığı da kullanılarak uzaklık formüllerinin de yakın anlamlı kelimelerin tespitinde nasıl bir rol oynadığı incelenmiştir.

Çizelge 5.9. incelendiğinde örneklerde verilen kelimelerin vektörel karşılığı bulunarak, bu değerin kelime listesinde yer alan diğer benzersiz kelimelerin vektörel değerleri arasındaki kosinüs uzaklıkları incelenmiştir. Çizelge 5'te örnek kelimeler üzerinde bu kelimeye en yakın 5 kelime CBOW, Skip-Gram ve Glove yöntemlerine göre listelenmiştir.

İlk sütunda yakın anlamlı kelimelerin araştırıldığı kelime vektörü verilmiştir. Çizelge 5.9.'da yer alan sonuçlar ilk olarak CBOW yöntemine göre incelendiğinde; "Pazartesi" kelimesi ile yakın anlamlı kelimelerin tespiti için pazartesi kelimesinin vektörüne en yakın vektörler sıralandığında listelenen kelimelerin Salı, Perşembe, Çarşamba şeklinde yine haftanın günlerinin bulunduğu görülmektedir. Yani bu durum modelin pazartesi kelimesini haftanın günlerinden biri olduğunu anladığı ve yakın anlamlı olarak haftanın diğer günlerini bulduğu şeklinde yorumlanabilir. Yine benzer şekilde "ocak" kelimesi ile yakın anlamlı kelimeler olarak yılın diğer ayları listelenmiş, "Ankara" kelimesinde diğer önemli iller, "beş" kelimesinde diğer rakamlar benzer kelimeler olarak ortaya çıkmıştır.

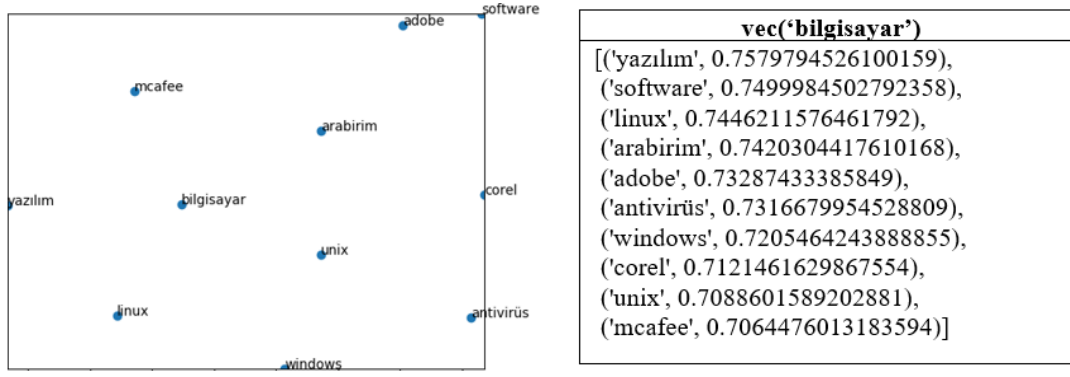
Çizelge 5.9.'da yer alan son örnek incelendiğinde ise "mi" kelimesine benzer kelimeler sorgulanmış cevap olarak mu?, mudur? vb. soru ekleri bulunmuştur. Böylece modelin "mi" sözcüğünün bir soru eki olduğunu tespit ettiği ve bu model kullanılarak bazı gramer tabanlı işlemlerde yapılabileceği görülmektedir. CBOW algoritmasına göre sonuçların başarılı olduğu görülmektedir. Çizelge 5.9 Skip-Gram algoritmasına göre analiz edildiğinde ise, hemen hemen CBOW algoritması ile aynı sonuçlara ulaşılmıştır. Yakın anlamlı kelimelerin tespitinde bir diğer Word2Vec yöntemi olan Skip-Gram algoritmasının da başarılı sonuçlar verdiği görülmektedir.

Çizelge 5.9. Glove yöntemine göre irdelendiğinde ise bu yöntemle de yakın anlamlı kelimelerin başarılı şekilde tespit edildiği ancak sıralanan 5 kelime içerisinde daha önce incelenen CBOW ve Skip-Gram algoritmalarında rastlanmayan bazı alakasız kelimelerinde listelendiği görülmektedir.

Çizelge 5.9. Word2Vec yöntemi ile eğitilen kelime benzerliklerinin incelenmesi

| Kelime            | CBOW                                                                                                                                                                             | Skip-Gram                                                                                                                                                                           | GLOVE                                                                                                                                                                                      |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| vec ('pazartesi') | [('salı', 0.9630045294761658),<br>( 'perşembe', 0.9585877656936646),<br>( 'çarşamba', 0.9553182721138),<br>( 'cuma', 0.9538884162902832),<br>( 'cumartesi', 0.9386611580848694)] | [('perşembe', 0.7971779108047485),<br>( 'çarşamba', 0.7718572616577148),<br>( 'salı', 0.7703690528869629),<br>( 'cumartesi', 0.7620112895965576),<br>( 'cuma', 0.6702909469604492)] | [('çarşamba', 0.5897661447525024),<br>( 'salı', 0.5786328315734863),<br>( 'perşembe', 0.5526260137557983),<br>( 'annem', 0.4993872046470642),<br>( 'cumartesi', 0.4941185712814331)]       |
| vec ('ocak')      | [('ağustos', 0.9286551475524902),<br>( 'mart', 0.9277894496917725),<br>( 'mayıs', 0.9231933355331421),<br>( 'şubat', 0.9167772531509399),<br>( 'temmuz', 0.9152134656906128)]    | [('şubat', 0.5146446824073792),<br>( 'mart', 0.4734412431716919),<br>( 'kasım', 0.4670507311820984),<br>( 'ekim', 0.43222731351852417),<br>( 'eylül', 0.42404061555862427)]         | [('nisan', 0.7427805662155151),<br>( 'mayıs', 0.7262980341911316),<br>( 'kasım', 0.7259864807128906),<br>( 'ekim', 0.7126656770706177),<br>( 'şubat', 0.6853300333023071)]                 |
| vec ('ankara')    | [('izmir', 0.7307801842689514),<br>( 'bursa', 0.6751932501792908),<br>( 'istanbul', 0.6552906036376953),<br>( 'adana', 0.6191496849060059),<br>( 'antalya', 0.5946915149688721)] | [('istanbul', 0.5656736493110657),<br>( 'izmir', 0.5347393751144409),<br>( 'antalya', 0.5179589986801147),<br>( 'bursa', 0.5097886919975281),<br>( 'erzurum', 0.5054399967193604)]  | [('istanbul', 0.5705186128616333),<br>( 'erzincan', 0.5660161375999451),<br>( 'tokyo', 0.5624301433563232),<br>( 'büyükkelçi', 0.5579214692115784),<br>( 'mmo.org.tr', 0.539327502250671)] |
| vec ('beş')       | [('üç', 0.900638222694397),<br>( 'sekiz', 0.8753728866577148),<br>( 'dört', 0.8631222248077393),<br>( 'yedi', 0.8289333581924438),<br>( 'dokuz', 0.779234766960144)]             | [('sekiz', 0.826900839805603),<br>( 'üç', 0.8074360489845276),<br>( 'elli', 0.7847145795822144),<br>( 'dört', 0.7656667232513428),<br>( 'say', 0.7624074220657349)]                 | [('o', 0.6285558342933655),<br>( 'altı', 0.6276718378067017),<br>( 'yirmi', 0.6216707229614258),<br>( 'sekiz', 0.5732786059379578),<br>( 'üç', 0.5651163458824158)]                        |
| vec ('mı')        | [('mu', 0.565196692943573),<br>( 'mıdır', 0.5488075613975525),<br>( 'mi', 0.5180516839027405),<br>( 'ne', 0.5127418041229248),<br>( 'misin', 0.4763811230659485)]                | [('mi', 0.7778604030609131),<br>( 'yoksa', 0.7676596641540527),<br>( 'mıdır', 0.7531024813652039),<br>( 'mu', 0.7458171844482422),<br>( 'ne', 0.7390531301498413)]                  | [('mu', 0.7941142916679382),<br>( 'peki', 0.7562239766120911),<br>( 'evet', 0.7499561309814453),<br>( 'ney', 0.7311218976974487),<br>( 'herhalde', 0.719590425491333)]                     |

CBOW algoritmasına göre “bilgisayar” kelimesi ile tespit edilen yakın anlamlı kelimelerin kosinüs benzerliği kullanılarak tespit edilmesi ve 2 boyutlu gösterimi ise Şekil 5.6.’da verilmiştir.



Şekil 5.6. Vektörün 2 boyutlu düzlemde gösterimi

Word2Vec modeli kullanılarak vektörel değerler üzerinden bir kelime ile yakın anlamlı kelimeler tespit edilebildiği gibi bu yöntemde kelimelerin vektörel değerleri oluşturulduğu için kelimeler üzerinde bazı matematiksel işlemlerde yapılabilmektedir.

Çizelge 5.10. incelendiğinde, 1 numaralı eşitlikte Mikolov ve arkadaşlarının [70] çalışmalarında bahsettikleri eşitliğin Türkçe karşılığı denenmiştir. Buna göre, “kral” ve “erkek” kelimelerinin vektörlerinin arasındaki mesafenin, “kraliçe” ve “kadın” kelimelerinin vektörlerinin arasındaki mesafeye eşit olması gerekmektedir. Yani “kraliçe” kelimesini elde edebilmek için “kadın” kelimesi eşitliğin diğer tarafına atıldığında (kral-erkek+kadın) oluşan eşitliğin vektörel karşılığının yaklaşık olarak “kraliçe” kelimesinin vektörel karşılığına eşit olmalıdır. 1 numaralı eşitlik incelendiğinde ise kral-erkek+kadın denkleminin sonucunda ortaya çıkan değere en yakın 5 değer sıralandığında CBOW ve Skip-Gram algoritmasına göre ilk sırada beklendiği üzere “kraliçe” kelimesi elde edilmiştir. Ancak Glove yönteminde bu sonuca ulaşılamamıştır. 2 numaralı eşitlikte ise “amca” – “erkek” = “teyze” – “kadın” eşitliği sınanmış, bu nedenle “teyze” kelimesi yalnız bırakılmıştır. Bu eşitlikteki amaç ise eşitlikten elde edilen vektörel değerlerin karşılığının “teyze” veya “hala” kelimesinin değerine yaklaşık olmasıdır. Bu eşitlikte de CBOW ve Skip-Gram algoritmasına göre beklenen sonuca ulaşılarak ilk sırada “teyze” kelimesi elde edilmiştir. Ancak “hala” kelimesi daha arka sıralarda yer almıştır. Glove yönteminde ise “teyze” kelimesi son sırada yer almıştır. 3 numaralı eşitlikte ise “doktor” ve “erkek” kelimeleri ile “hemşire” ve “kadın” kelimelerinin vektörel değerlerinin farklarının eşit olduğu düşünülerek daha önceki örneklerde olduğu gibi “hemşire” kelimesinin elde edilmesi amaçlanmıştır. Bu örnekte ise CBOW algoritmasına göre beklenildiği gibi “hemşire” kelimesi cevap kelime vektörü olarak döndürülmüştür. Ancak Skip-Gram algoritmasına göre “hemşire” kelimesi ikinci sırada yer almıştır. Bu durum daha nadir geçen kelimelerin tespitinde kullanılabileceği şeklinde yorumlanabilir. 3 numaralı eşitlik Glove algoritmasına göre

bakıldığında ise hem beklenen sonuçlara ulaşamadığı hem de elde edilen sonuçların alakalı kelimeler olmadığı görülmüştür.

Kelimelerin vektörleştirilerek üzerinde matematiksel işlemlerin yapılabilmesi ve elde edilen sayısal değer ile ilgili vektörün arasındaki mesafeye en yakın kelimenin vektörünün döndürüldüğü bu uygulamada Glove yönteminin Word2Vec algoritmalarına göre daha başarısız olduğu görülmektedir.

Çizelge 5.10. Word2Vec yöntemi ile eğitilen kelime benzerliklerinin incelenmesi

| CBOW                                                                                                                                                                                | Skip-Gram                                                                                                                                                                         | GLOVE                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>1. <math>\text{vec}(\text{"kral"}) - \text{vec}(\text{"erkek"}) + \text{vec}(\text{"kadın"}) = \text{vec}(\text{cevap})</math></b>                                               |                                                                                                                                                                                   |                                                                                                                                                                                     |
| [(‘kraliçe’, 0.5058310031890869),<br>(‘prens’, 0.4587770998477936),<br>(‘imparator’, 0.4512189030647278),<br>(‘prens’, 0.44870319962501526),<br>(‘hükümdar’, 0.43795979022979736)]  | [(‘kraliçe’, 0.6649389863014221),<br>(‘prens’, 0.6619410514831543),<br>(‘hükümdar’, 0.6613924503326416),<br>(‘imparator’, 0.6476287841796875),<br>(‘frigya’, 0.6386576890945435)] | [(‘prens’, 0.36667877435684204),<br>(‘prens’, 0.3593514561653137),<br>(‘napolyon’, 0.342054545879364),<br>(‘aleksandr’, 0.3324899971485138),<br>(‘ferdinand’, 0.31234925985336304)] |
| <b>2. <math>\text{vec}(\text{"amca"}) - \text{vec}(\text{"erkek"}) + \text{vec}(\text{"kadın"}) = \text{vec}(\text{cevap})</math></b>                                               |                                                                                                                                                                                   |                                                                                                                                                                                     |
| [(‘teyze’, 0.6639469861984253),<br>(‘dayı’, 0.5801525712013245),<br>(‘hala’, 0.5743905305862427),<br>(‘abla’, 0.5725879669189453),<br>(‘ağabey’, 0.5711960792541504)]               | [(‘teyze’, 0.46290165185928345),<br>(‘enişte’, 0.39535993337631226),<br>(‘dayı’, 0.38905856013298035),<br>(‘anneanne’, 0.38179734349250793),<br>(‘abla’, 0.3774968683719635)]     | [(‘oğul’, 0.7785995602607727),<br>(‘dede’, 0.7496525049209595),<br>(‘dayı’, 0.743678092956543),<br>(‘baba’, 0.7313336730003357),<br>(‘teyze’, 0.7183244228363037)]                  |
| <b>3. <math>\text{vec}(\text{"doktor"}) - \text{vec}(\text{"erkek"}) + \text{vec}(\text{"kadın"}) = \text{vec}(\text{cevap})</math></b>                                             |                                                                                                                                                                                   |                                                                                                                                                                                     |
| [(‘hemşire’, 0.7681228518486023),<br>(‘hekim’, 0.7603170275688171),<br>(‘hasta’, 0.7203007936477661),<br>(‘kardiyolog’, 0.682590901851654),<br>(‘hastabakıcı’, 0.6806880235671997)] | [(‘hekim’, 0.6817376017570496),<br>(‘hemşire’, 0.6360365152359009),<br>(‘eczacı’, 0.6312180161476135),<br>(‘hasta’, 0.6196988821029663),<br>(‘bakıcı’, 0.6053770184516907)]       | [(‘adam’, 0.37649837136268616),<br>(‘ameliyat’, 0.37092214822769165),<br>(‘hasta’, 0.33870816230773926),<br>(‘hanım’, 0.3268465995788574),<br>(‘bleda’, 0.3250676989555359)]        |

Çizelge 5.10.’da bir önceki uygulamada Word2Vec algoritmaları daha başarılı oldukları için bu uygulamada Word2Vec yöntemine göre sonucular incelendiğinde beklenen sonuçlara ulaşıldığı, özellikle Word2Vec yöntemi ile kelimelerin vektörel değerleri üzerinde matematiksel işlemlerin yapılabilmesi ve sonuçların oldukça başarılı performans gösterdiği görülmüştür.

Çizelge 5.11. Eşleştirme sorguları

|                                                 |                                                                                                                                                             |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| vec(“türkiye”) - vec(“türk”) + vec(“japon”)     | ('japonya', 0.5943777561187744)                                                                                                                             |
| vec(“türkiye”) - vec(“ankara”) + vec(“bakü”)    | ('azerbaycan', 0.5270528793334961)                                                                                                                          |
| vec(“türkiye”) - vec(“osmanlı”) + vec(“bizans”) | ('yunanistan', 0.5426459312438965)                                                                                                                          |
| vec(“islam”) - vec(“türkiye”) + vec(“almanya”)  | ('yahudi', 0.5666460990905762),<br>(‘musevi’, 0.5661754012107849),<br>(‘hristiyan’, 0.5618569254875183)                                                     |
| vec(“lira”) - vec(“almanya”) + vec(“türkiye”)   | ('euro', 0.5543266534805298),<br>(‘sterlin’, 0.5256396532058716),<br>(‘dolar’, 0.5223875641822815),<br>(‘avro’, 0.4999542832374573),                        |
| vec(“odtü”) – vec(“türkiye”) + vec(“ingiltere”) | ('birmingham', 0.4685020446777344),<br>(‘northwestern’, 0.46640270948410034),<br>(‘massachussets’, 0.46634599566459656),<br>(‘sussex’, 0.4653938412666321), |

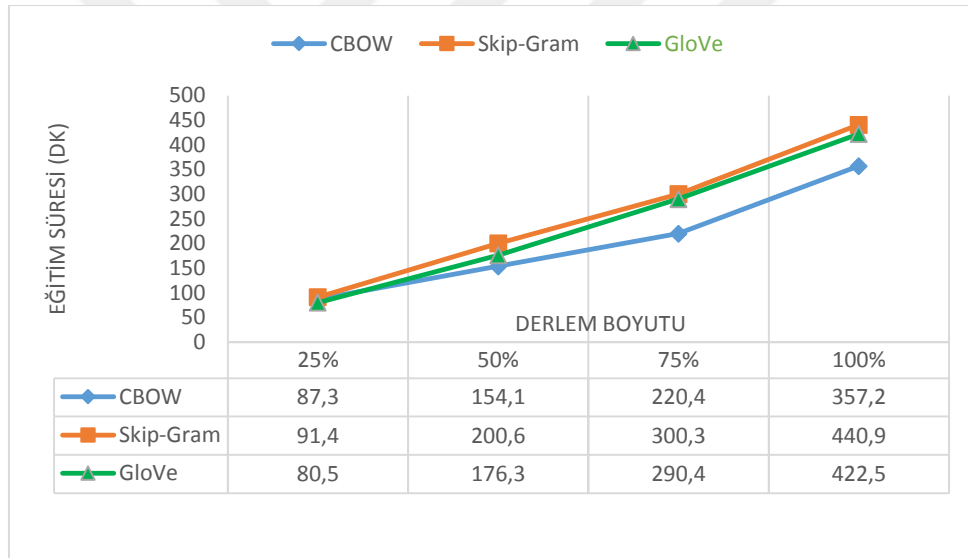
Yine yukarıdaki Çizelge 5.11’de görüldüğü gibi ilk iki satırda ülke-millet ve ülke-başkent sorgulaması yapılmış ikisinde de beklenen sonuçlar ilk sırada dönmüştür. Özellikle “Türkiye-Osmanlı” kelimeleri ile “Yunanistan-Bizans” kelimeleri arasındaki bağlantının tespit edilebilmiş olması heyecan vericidir. Ülke-din sorgulamasında ise, sonuç olarak Almanya’da en çok inanılan din olan Hristiyanlığın ilk sırada çıkması beklenmiş ancak bu kelime üçüncü sırada görülmüştür. Bu durumun da derlem içerisinde geçen kelimeler ile alakalı olduğunun bir örneğidir. Derlem zenginleştikçe sonuçların doğruluk oranı da artmaktadır.

Aşağıda yer alan Şekil 5.7.’de ise verilen ilk iki kelime arasındaki benzerlik nokta çarpımı ile tespit edilerek çıkan sonuç ile son kelime vektörü arasındaki farkın hangi kelimenin vektörüne denk geldiği incelenmiştir. Böylece ünlü bir futbolcu olan Ronaldo ve futbol kelimelerinin vektörleri arasındaki fark ile ressam kelimesi ve bir ressamın isminin (icracı – icra edilen iş) ortaya çıkması beklenmiştir. Sonuç olarak tabloda görüldüğü gibi “Zonaro” kelimesi dönmüştür. Zonaro kelimesi Google da araştırıldığında ise, wikipedia kayıtlarında 19.yy da Osmanlı sarayında hizmet vermiş Türk ressam olarak tanınan İtalyan bir ressam olduğu bilgisi ile karşılaşılmıştır [143]. Yani bizim bile bilmediğimiz bir bilgi model tarafından tespit edilmiştir.

|                                                                                                 |
|-------------------------------------------------------------------------------------------------|
| <b>Input:</b> <code>en_yakin_benzerlik_bul("ronaldo", "futbolcu", "ressam")</code>              |
| <b>Output:</b> ronaldo ve futbolcu kelimeleri, zonaro ve ressam kelimeleri kadar ilişkilidir... |
| <b>Input:</b> <code>eslesmeyi_bul("bursa", "karadeniz", "marmara", "akdeniz")</code>            |
| <b>Output:</b> 'bursa'                                                                          |

Şekil 5.7. Vektörler arasındaki yakınlık ve ilişkiler

Bir diğer örnekte ise tanımlanan bir fonksiyon ile girdi (input) olarak verilen kelimelerin vektörleri incelenerek vektörler arasındaki en düşük değer en uzak mesafe bir başka deyişle anlam bakımından en uzak kelime olması beklenmiştir. Örnek olarak “Karadeniz, Akdeniz, Marmara ve Bursa” kelimeleri verilmiş model tarafından ilk 3 kelimenin deniz veya bölge isimleri olduğu, son kelimenin bu kelimeler ile aynı grupta yer almadığı tespit edilmiştir. Çıktı olarak “Bursa” kelimesi dönmüştür.



Şekil 5.8. Kelime vektörlerinin eğitim süreleri

Derlem oluşturulduktan sonra kelime vektörlerinin CBOW, Skip-Gram ve Glove yöntemlerine göre eğitim süreleri Şekil 5.8.’de gösterilmiştir. Derlem boyutunun %25, %50, %75 ve son olarak derlemin tamamı yani %100’lük kısmı eğitilerek eğitim süreleri incelenmiştir. Bu aşamada eğitim süreleri ilk adımda CBOW algoritması 87,3 dk, Skip-Gram algoritması 91,4 dk ve Glove yöntemi 80,5 dk olarak elde edilmiştir. Yani bu aşamada en hızlı yöntemin CBOW algoritması olduğu bu algoritmayı Glove ve Skip-Gram algoritmasının takip ettiği söylenebilir. Grafik incelendiğinde bu durum son adıma kadar doğrusal olarak bu sıralama ile devam ettiği şeklinde yorumlanabilir.

### 5.4.3. Uygulamanın Sonuçları

Bu uygulamada Dataset-1 üzerinde, son dönemlerde geliştirilen en önemli iki model olan Word2Vec ve Glove yöntemleri ile kelime vektörleri oluşturulmuştur. Bu iki modelin en büyük avantajı olan kelimelerin anlamsal olarak yakınlıklarının tespit edilmesi konusu analiz edilmiştir. Bu analiz sonucunda görülmüştür ki Word2Vec yöntemi Glove yöntemine göre anlamsal benzerliklerin tespiti konusunda daha iyi sonuçlar vermiştir. Özellikle “ $\text{vec}(\text{kral}) - \text{vec}(\text{erkek}) + \text{vec}(\text{kadın}) = \text{vec}(\text{kraliçe})$ ” gibi bir denklem formatında vektörler üzerinde matematiksel işlemler yapılarak beklenen kelimenin model tarafından önerilmesi işlemlerinde Word2Vec yöntemi oldukça üstün bir başarıyı göstermiştir.

Uygulamada Word2Vec yönteminin iki temel algoritması olan CBOW ve Skip-Gram algoritmaları da analiz edilmiştir. Çalışmada kullanılan derlem büyük bir derlem olduğu için bu derlem üzerinde CBOW algoritması daha iyi bir performans göstermiştir. Derlemin daha küçük bir kısmı ile çalışmalar yapıldığında ise, Skip-Gram yönteminin daha başarılı olduğu görülmüştür. Yine derlem içerisinde sık geçen kelimelerin temsilinde CBOW, nadir geçen kelimelerin temsilinde ise, Skip-Gram algoritmasının daha başarılı olduğu görülmüştür. Yani tercih edilecek algoritmanın, üzerinde çalışılan derleme göre tercih edilmesi daha uygun olacaktır.

Uygulamanın son kısmında ise, Word2Vec modeli CBOW ve Skip-Gram algoritmaları ile Glove yöntemi eğitim süreleri bakımından analiz edilmiştir. Öncelikle derlemin tamamı daha sonra derlemin %75’lik kısmı ardından %50 ve son olarak %25’i modellere derlem (corpus) olarak verilmiş ve eğitim süreleri kıyaslanmıştır. Bu inceleme sonunda CBOW algoritmasının eğitim süresinin tüm adımlarda Skip-Gram ve Glove algoritmalarından daha hızlı olduğu görülmüştür. Ayrıca kelime vektörlerinin kullanılarak anlamsal olarak yakın kavramların tespit edilmesinde hem Word2Vec hem de Glove yöntemlerinin başarılı oldukları ancak hem hız hem de başarımları olarak Word2Vec yönteminin daha iyi olduğu görülmüştür.

## 6. ÖNERİLEN YÖNTEMLER VE DENEYSEL SONUÇLAR

### 6.1. Önerilen Yöntem 1: Önceden Eğitilmiş Kelime Vektörleri

Kelimelerin sayısal olarak temsil edilme biçimi metin işleme probleminin belki de en kritik noktası olarak kabul edildiği kelime vektörlerinin eğitildiği önceki kısımlarda bahsedilmişti. Tez çalışmasının ana katkılarından biri olarak Uygulama 3'te eğitilen kelime vektörlerinin, önceden eğitilmiş kelime vektörleri olarak kullanılarak transfer öğrenme tekniği ile başka bir görev üzerinde kullanıldıkları bir yöntem önerilmiştir [144].

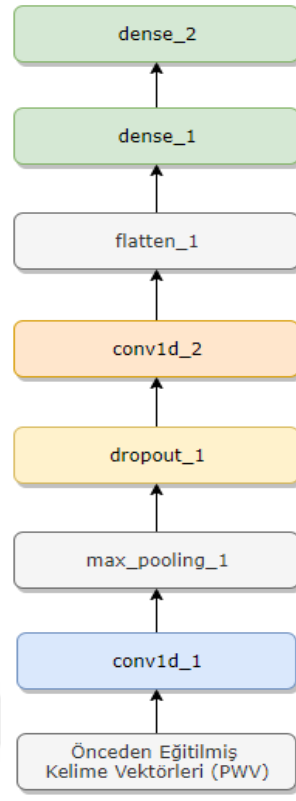
#### 6.1.1. Yöntemin Amacı

Önerilen yöntemin kullanıldığı bu uygulamada önceden eğitilmiş Türkçe kelime vektörleri ve derin sinir ağları (DNN) kullanılmıştır. Önceden eğitilmiş kelime vektörleri ile 1,5 milyon örnek ve 10 sınıftan oluşan Dataset-2 üzerinde derin sinir ağları ile metin sınıflandırma işlemi yapılmıştır. Bu yöntemde, derin sinir ağı mimarilerinden CNN ve RNN ile RNN mimarisinin diğer türleri olan LSTM ve GRU modelleri ve çeşitli varyasyonları kullanılmıştır. Çalışmada kullanılan kelime temsil yöntemlerinin performansları, doğruluk oranına etkileri ve derin sinir ağı mimarilerinin başarımları detaylı olarak analiz edilmiştir. Yöntemin temel amacı derin sinir ağları ile yapılan metin sınıflandırma uygulamasında elde edilen doğruluk değerlerinin önerilen bu yöntem ile daha da geliştirilmesidir. Önceden eğitilmiş kelime vektörleri bu aşamadan sonra İngilizce karşılığından dolayı literatürde yer aldığı gibi PWV (Pre-trained Word Vectors) şeklinde kısaltılarak kullanılacaktır.

#### 6.1.2. Uygulamanın Geliştirilmesi

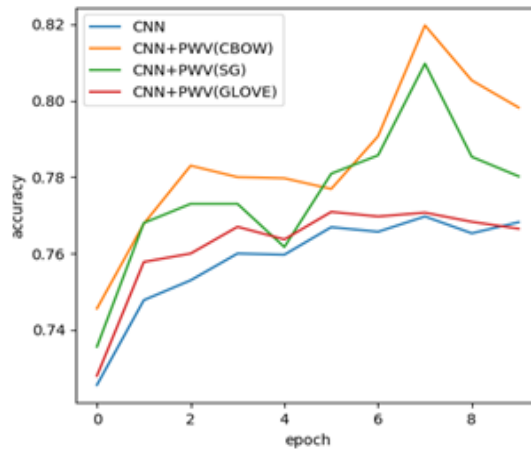
Önerilen yöntemin temel alt yapısı, deneysel çalışmalar sırasında bu aşamaya kadar üretilmiş Dataset-1 ve Dataset-2 veri setleri ve geliştirilen Uygulama 2 ve Uygulama 3 ile oluşturulmuştur. Uygulama 2 ile Dataset-1 ve Dataset-2 üzerinde ön işlem çalışmaları yapılmış daha sonra Uygulama 3 ile kelime vektörleri üretilmiştir. Eğitilen kelime vektörleri önceden eğitilmiş kelime vektörleri (PWV) olarak transfer öğrenme tekniği ile derin sinir ağı modelleri ve varyasyonlarıyla (hyper parameters tuning) Dataset-2 üzerinde sınıflandırma işlemi yapılmıştır. Bu modellerin başarımları kıyaslanmış ve analiz edilmiştir. Son olarak önceden eğitilmiş kelime vektörleri (PWV), derin sinir ağları ile birlikte kullanılmış ve başarıma olan etkisi incelenmiştir. Önerilen yöntemin genel mimarisi şekilde gösterilmiştir.





Şekil 6.2. Uygulanan CNN modelinin yapısı

Modelin başarımı incelendiğinde ise 10 tur (epoch) sonunda validation accuracy değerleri Şekil 6.3.'te gösterilmiştir. Modelde ilk olarak CNN modeli kullanılmış ve bir başarıım değeri elde edilmiştir. Daha sonra bu çalışmanın da hedeflerinden biri olan önceden eğitilmiş kelime vektörlerinin model başarımına etkisi incelenmiştir. Sonuçlar analiz edildiğinde önceden eğitilmiş kelime vektörleri kullanılarak model başarımının geliştirildiği görülmüştür. En başarılı performans ise Word2Vec yöntemi ve CBOW algoritması olmuştur.

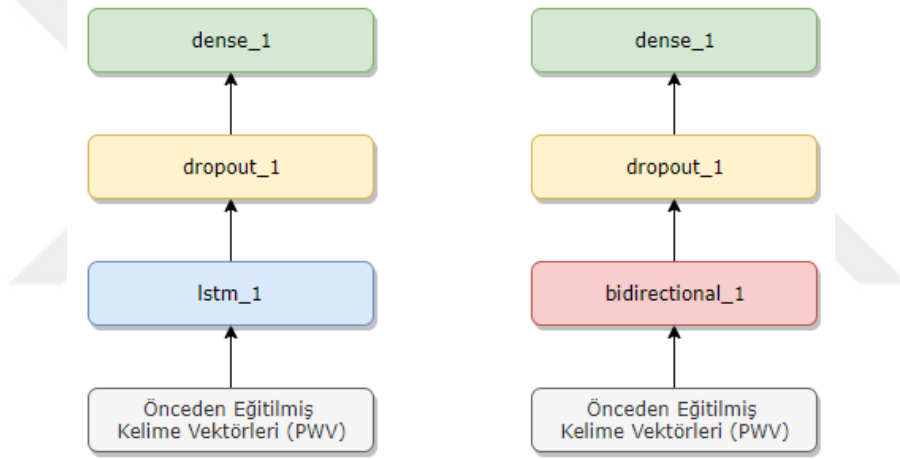


Şekil 6.3. CNN modelinin doğruluk grafiği

Test verileri dikkate alındığında ikinci en iyi performansı bir Word2Vec yöntemi olan skip-gram (SG) algoritması kullanılarak eğitilen kelime vektörleri ile geliştirilen model göstermiştir. Glove yöntemi ile eğitilen kelime vektörleri ise Word2Vec yöntemlerinin gerisinde kaldığı görülmüştür. Ancak dikkat edilmelidir ki hangi yöntemle olursa olsun önceden eğitilmiş kelime vektörlerinin kullanılması model başarımını geliştirmiştir.

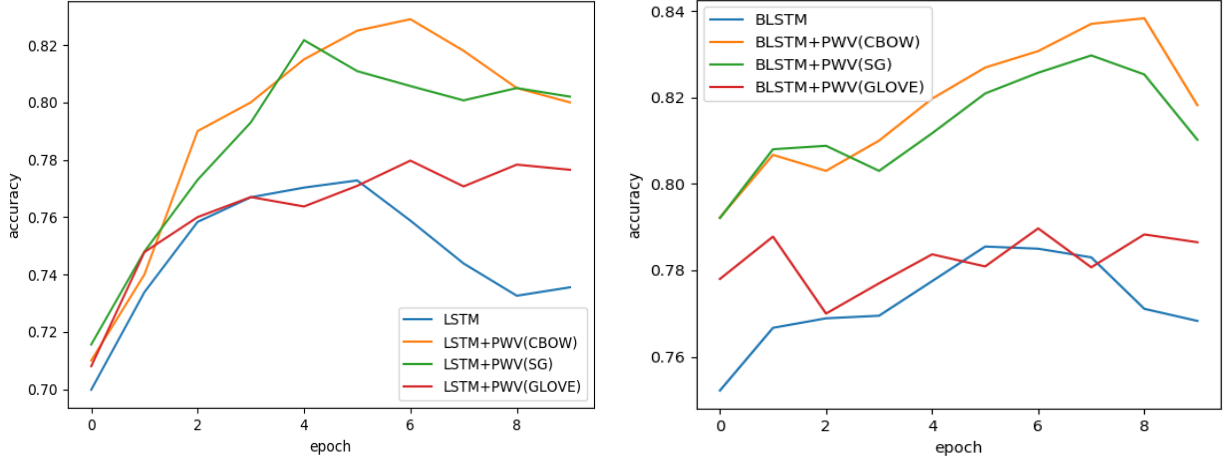
#### 6.1.2.2. LSTM & BLSTM

Çalışmada kullanılan LSTM modelinin yapısı Şekil 6.4.'te gösterilmiştir. Geliştirilen modelde 256 nöron kullanılmıştır. Modelin eğitim işlemi sırasında aşırı öğrenmesini (overfitting) engelleyebilmek için LSTM katmanından sonra Dropout katmanı eklenmiştir. Eklenen dropout değeri ise 0.2 olarak uygulanmıştır. Çıktı değerinin hesaplanmasında ise softmax kullanılmıştır. BLSTM yapısında ise LSTM modelinden farklı olarak LSTM hücreleri iki yönlü olarak eğitilmiştir.



Şekil 6.4. Uygulanan LSTM & BLSTM modelinin yapısı

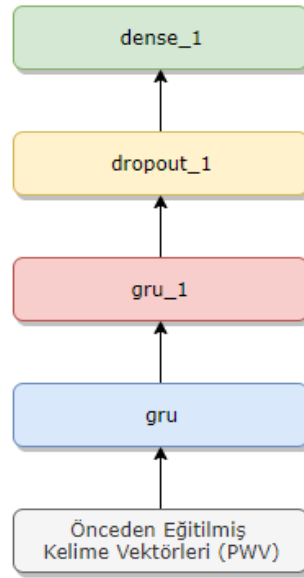
Şekil 6.5.'teki sonuçlar incelendiğinde test verileri dikkate alındığında BLSTM modelinin LSTM modeline göre daha başarılı olduğu görülmüştür. Ayrıca önceden eğitilmiş kelime vektörlerinin kullanılması ile hem LSTM modelinin hem de BLSTM yapısının başarımının iyileştirildiği görülmüştür. Bu modelde de Word2Vec yöntemi Glove yöntemine göre daha başarılı sonuçlar göstermiştir. CBOW algoritması ile Skip-Gram algoritması birbirine yakın sonuçlar verse de CBOW yine daha başarılı olmuştur.



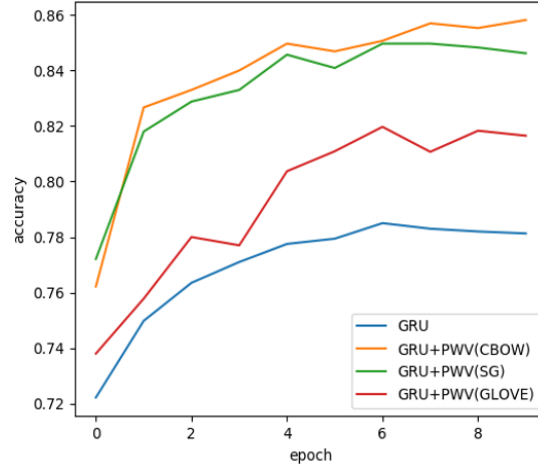
Şekil 6.5. LSTM & BLSTM modellerinin doğruluk grafikleri

### 6.1.2.3. GRU

Çalışmada kullanılan GRU modelinde 3 katmanlı bir yapı kullanılmıştır. İlk katmanda 512 nöron ikinci katmanda 64 nöron yerleştirilmiştir. Son katman olan çıktı katmanında ise veri setindeki sınıf sayısı olan 10 nöron bulunmaktadır. İlk iki katman arasında daha önceki modellerde de kullanıldığı gibi aşırı öğrenmeyi önlemek amacıyla dropout katmanı eklenmiştir. Modelin mimarisi şekil 6.6.'da gösterilmiştir.



Şekil 6.6. Uygulanan GRU modelinin yapısı

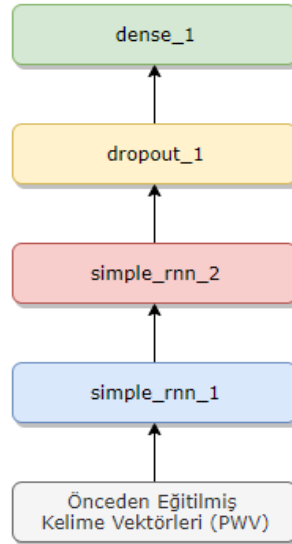


Şekil 6.7. GRU modelinin doğruluk grafiği

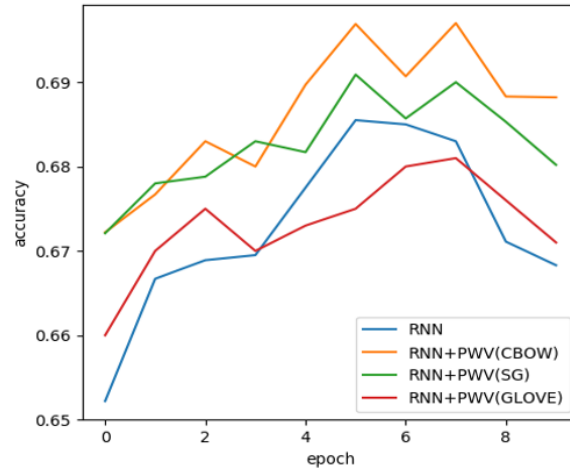
Şekil 6.7. incelendiğinde ise önceden eğitilmiş kelime vektörleri modele eklendiğinde GRU modelinin mevcut başarımının önemli oranda geliştirildiği görülmüştür. CBOW ve Skip-Gram yöntemi ile eğitilen vektörler yüklendikten sonraki modelin başarımı yakın değerler olsa da CBOW yönteminin daha başarılı olduğu görülmüştür.

#### 6.1.2.4. RNN

RNN mimarisi 3 katmanlı Şekil 6.8.'de görüldüğü gibi tasarlanmıştır. İlk katmana 256 ikinci katmana ise 64 nöron yerleştirilmiştir. Son katmanda ise çıktı katmanı olduğu için 10 nöron bulunmaktadır. Son katmandan önce aşırı öğrenmenin engellenebilmesi için dropout kullanılmıştır. Son katmanda softmax ile çıktı değerleri hesaplanmıştır.



Şekil 6.8. Uygulanan RNN modelinin yapısı

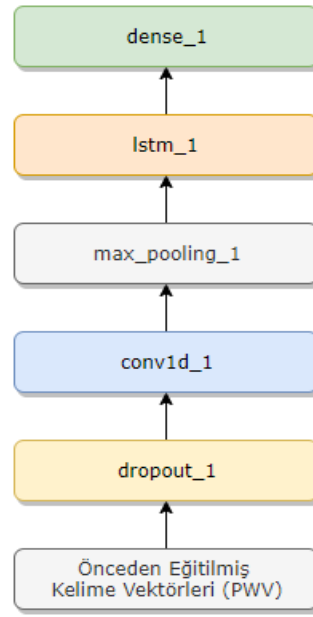


Şekil 6.9. RNN modelinin doğruluk grafiği

Şekil 6.9. incelendiğinde RNN mimarisinin çok başarılı olmadığı görülmüştür. Önceden eğitilmiş kelime vektörlerinin de diğer modellerde olduğu kadar önemli etkiler yapamadığı grafikten anlaşılmaktadır. Ancak yine az bir oranda da olsa CBOW algoritmasının başarımı iyileştirdiği söylenebilir.

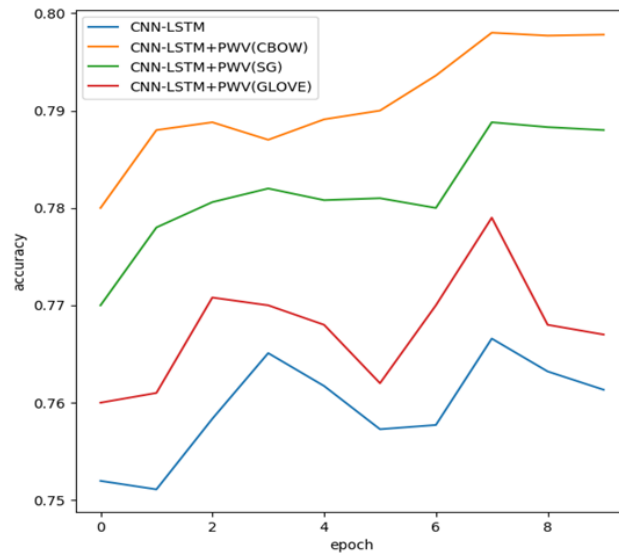
#### 6.1.2.5. CNN-LSTM

Çalışmada son olarak daha önce kullanılan iki derin sinir ağı modeli olan CNN ve LSTM yapılarının birlikte kullanıldığı bir mimari oluşturulmuştur. Bu model önceden eğitilmiş kelime vektörlerinin girdi olarak ağı verildikten sonra konvolüsyon ve ortaklama (max-pooling) adımları uygulanmıştır. Bu işlemlerin tamamlanmasından sonra LSTM katmanı eklenmiş ayrıca aşırı öğrenmeyi önlemek için dropout katmanı yerleştirilmiştir. Son katman ise çıktı katmanı olduğundan 10 nöron bulunmaktadır. Modelin mimarisi Şekil 6.10.'da görülmektedir.



Şekil 6.10. Uygulanan CNN-LSTM modelinin yapısı

Test sonuçları incelendiğinde CNN-LSTM olarak adlandırılan model önceden eğitilmiş kelime vektörlerinin kullanılmasıyla model başarımında önemli bir gelişmenin olduğu görülmektedir. Skip-gram modeli ilk etapta daha başarılı sonuçlar gösterse de ilerleyen turlarda (epoch) CBOW modeli daha başarılı olmuştur. Sonuçlar Şekil 6.11.’de verilmiştir.

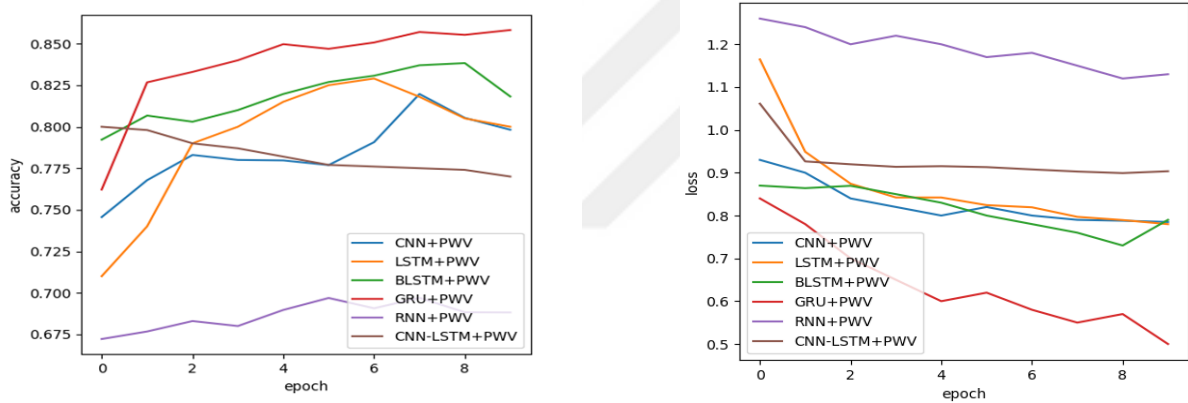


Şekil 6.11. CNN-LSTM modelinin doğruluk grafiği

### 6.1.3. Uygulamada Elde Edilen Bulgular

Önerilen yöntem ile önceden eğitilmiş kelime vektörlerinin model başarımlarına etkisi incelenmeye çalışılmış ve bunun için kelime temsil yöntemlerinden olan CBOW, Skip-Gram, Glove yöntemleri kullanılarak kelime vektörleri oluşturulmuştur. Bununla birlikte Dataset-2 üzerinde neredeyse tüm derin sinir ağları eğitilmiş ve test sonuçları şekil 6.12’de görüldüğü gibi doğruluk ve kayıp fonksiyon grafikleri dikkate alınarak değerlendirilmiştir. Deneysel çalışmaların tümünde optimizier olarak *adam optimizier*, kayıp fonksiyonu olarak *categorical\_crossentropy* kullanılmış batch\_size olarak 64 değeri tercih edilmiştir.

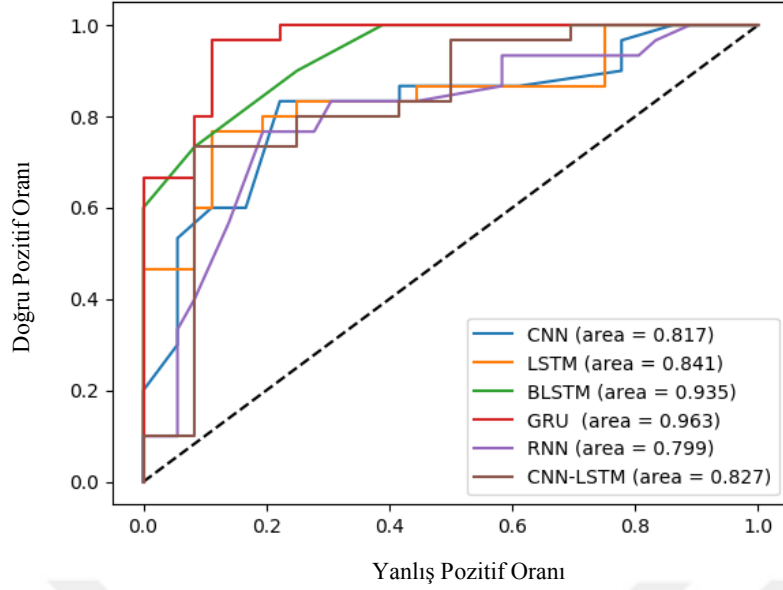
Sonuçlar incelendiğinde en başarılı performansı gösteren model GRU modeli olurken, BLSTM modeli başarımlarına göre sıralamada ikinci sırada yer almış sonrasında ise birbirlerine yakın performanslar gösteren LSTM ve CNN-LSTM modelleri gelmiştir. Dataset-2 müşterilerden toplanmış gerçek veriler olduğu için veri kirliliği, yazım hataları, yanlış etiketleme gibi durumlarında söz konusu olduğu görülmüştür.



Şekil 6.12. Modellerin doğruluk ve kayıp fonksiyon grafikleri

Deneysel çalışmalarda RNN tabanlı (GRU, LSTM ve RNN) modellerde katman sayısının artırılmasının model başarımına etkisinin olumlu olmadığı aksine bir noktadan sonra olumsuz olduğu görülmüştür. Bu çalışmada genelde RNN mimarilerde 3 katman yapısı kullanılmıştır. Katman sayısını artırmak yerine nöron sayısını artırmanın daha faydalı olduğu deneysel çalışmalarda görülmüştür. Ayrıca Bidirectional (iki yönlü) yapılar (BLSTM ve BRNN) klasik RNN ve LSTM yapısına göre daha başarılı olmuşlardır. Sonuçlar Şekil 6.12.’de modellerin doğruluk ve kayıp fonksiyon grafikleri görülmektedir.

Çalışmada kullanılan tüm modellerin ROC eğrisi grafiği Şekil 6.13.’te görülmektedir [145,146]. Önceden eğitilmiş kelime vektörleri (PWV) ile birlikte kullanılan GRU modeli tüm modeller içerisinde en başarılı performansı gösterirken, RNN modeli en kötü performansı göstermiştir.



Şekil 6.13. Tüm modellerin PWV yöntemi ile ROC eğrisi grafiği

Önceden eğitilmiş kelime vektörlerinin performansları incelendiğinde ise Word2Vec modeli Glove modeline göre daha başarılı olduğu açıkça görülmüştür. Tüm derin sinir ağı modellerinin başarımları incelendiğinde hepsinde Word2Vec yöntemi ile eğitilmiş vektörler ile eğitilen ağlar daha başarılı olmuştur. Word2Vec yöntemi içerisinde de CBOW algoritması bazı modellerde sonuçlar yakında olsa Skip-Gram algoritmasına göre daha iyi performans göstermiştir. Bu nedenle tabloda CBOW algoritması ile önceden eğitilmiş kelime vektörlerinin (PWV- Pre Trained Word Vectors) doğruluk (accuracy) değerleri dikkate alınmıştır.

Çizelge 6.1.'de modellerin tur (epoch) sayıları, önceden eğitilmiş kelime vektörleri kullanılmadan elde edilmiş test sonuçları ve önceden eğitilmiş kelime vektörleri kullanıldıktan sonraki test sonuçları gösterilirken model başarımının ne kadar iyileştirildiği bilgisi verilmiştir.

Çizelge 6.1. Tüm modeller için PWV ile elde edilen sonuçlar

| Model    | Doğruluk Değeri<br>( PWV olmadan) | Doğruluk Değeri<br>(PWV ile) | Performans<br>Artırımı | Eğitim<br>Süresi (sn) |
|----------|-----------------------------------|------------------------------|------------------------|-----------------------|
| GRU      | 78.85%                            | 85.82%                       | <b>6.97%</b>           | 32,580                |
| BLSTM    | 78.55%                            | 83.80%                       | <b>5.25%</b>           | 36,400                |
| LSTM     | 77.27%                            | 82.79%                       | <b>5.52%</b>           | 24,890                |
| CNN-LSTM | 76.10%                            | 79.80%                       | <b>3.70%</b>           | 26,100                |
| CNN      | 75.69%                            | 77.03%                       | <b>3.34%</b>           | 7,810                 |
| RNN      | 68.66%                            | 71.10%                       | <b>2.44%</b>           | 13,790                |

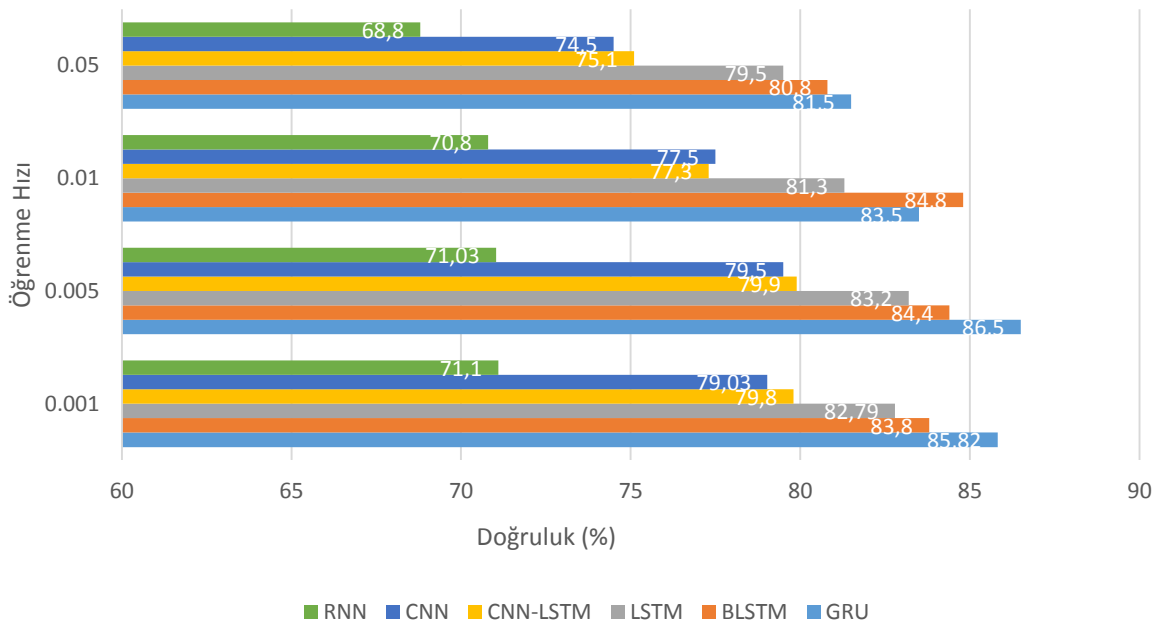
Buna göre en başarılı performansı gösteren GRU modeli önceden eğitilmiş kelime vektörleri ile birlikte kullanılarak başarımları değeri en fazla geliştirilen model olmuştur. Performans gelişimi sütunu incelendiğinde ciddi oranlarda model performanslarının iyileştirildiği görülmektedir. Ancak çalışma sonunda açıkça görülmüştür ki önceden eğitilmiş kelime vektörleri kullanılarak eğitilen ağlar daha başarılı olmuş ve başarımları değeri artırılmıştır.

Son olarak Çizelge 6.1.'de modellerin eğitim süreleri verilmiştir. Bu bölüm incelendiğinde GRU ve BLSTM modelleri en yüksek başarımları değerine sahip modeller olmalarına rağmen eğitim süreleri dikkate alındığında diğer modellerden daha yavaş bir eğitim süreci geçirdikleri görülmektedir. BLSTM yapısal olarak, LSTM modelinin iki yönlü olarak uygulanan bir türü olduğundan tabloda da görüldüğü üzere eğitim süresi daha uzun olmuştur. En hızlı eğitim işlemi ise CNN modeli üzerinde yapılmıştır, bunu RNN modeli izlemiştir.

Çizelge 6.2. PWV ile sınıflandırma sonuçları.

| Model    | Doğruluk | Özgünlük | Hassasiyet | F-Score |
|----------|----------|----------|------------|---------|
| GRU      | .85      | .84      | .80        | .81     |
| BLSTM    | .83      | .84      | .74        | .78     |
| LSTM     | .82      | .83      | .74        | .78     |
| CNN-LSTM | .79      | .71      | .80        | .75     |
| CNN      | .77      | .68      | .80        | .73     |
| RNN      | .71      | .61      | .75        | .67     |

Uygulamada kullanılan her bir model için PWV ile elde edilen sınıflandırma başarımları Çizelge 6.2.'de görülmektedir. En iyi sonuçlar, %85 doğruluk, %84 özgünlük ve %80 hassasiyet skorları ile GRU modeli ile elde edilmiştir.



Şekil 6.14. Farklı öğrenme hızı değerleri için doğruluk değerleri

Deneyisel çalışmalar sırasında doğruluk değerini etkileyen en önemli parametrenin öğrenme hızı parametresi olduğu görülmüştür [147]. Şekil 6.15.'te görüldüğü gibi her bir derin sinir ağı modeli için farklı öğrenme hızı parametreleri test edilmiş ve parametrenin optimum değerde ayarlanmasının modelin performansını önemli oranda etkilediği anlaşılmıştır. Deneyler sırasında bu uygulama için en uygun öğrenme hızı değerinin 0.005 olduğu görülmüştür.

#### 6.1.4. Uygulamanın Sonuçları

Derin öğrenme, son dönemde oldukça popüler olmuş önemli bir araştırma alanıdır. Derin sinir ağlarına dayalı olan derin öğrenme konusunda literatürde birçok farklı yöntem önerilmiştir ve bunların bazıları özellikle bazı araştırma alanlarında oldukça başarılı sonuçlar elde edilmesini sağlamıştır. Doğal dil işleme alanı da söz konusu önemli araştırma alanlarından biridir.

Önerilen bu yöntemde, Uygulama 3 ile üretilmiş kelime vektörleri daha sonra transfer öğrenimi ile transfer edilmiş ve Dataset-2 üzerinde metin sınıflandırma işlemi uygulanmıştır. Bu yöntemle, mevcut doğruluk değeri derin öğrenme modelleri üzerinde yaklaşık %5 - %7 aralığında artırılmıştır. Yapılan çalışmalar sonucunda derin sinir ağı mimarileri birbirleriyle kıyaslanmış ve en başarılı derin sinir ağı yapısı GRU modeli olmuştur. Önceden eğitilmiş kelime vektörlerinin derin sinir ağı modellerine etkileri incelendiğinde ise önceden eğitilmiş kelime vektörleri kullanılarak model başarımları geliştirildiği tespit edilmiştir. Mevcut model performanslarının gelişim oranları dikkate alındığında ise en başarılı kelime temsil yönteminin Word2Vec modeline ait CBOW algoritması olduğu görülmüştür.

Sonuç olarak, bu çalışma ile Türkçe doğal dil işleme derin sinir ağları üzerinde önceden eğitilmiş kelime vektörlerinin etkisi irdelenmiş ve sonuçları ayrıntılı olarak analiz edilmiştir. Bu çalışma ile elde edilen tespit ve bulgular bundan sonraki Türkçe metin işleme ve doğal dil işleme üzerinde derin öğrenme çalışmalarına ışık tutması buna ek olarak önerilen yöntemin doğruluk geliştirme çalışmalarına katkı sağlanması beklenmektedir.

## 6.2. Önerilen Yöntem 2: Sözlük Yöntemi

Bu bölümde, tez çalışmasında önerilen bir diğer özgün yöntem olan sözlük yöntemi tanıtılmıştır. Geliştirilen bu yöntemin ihtiyacının nasıl ortaya çıktığı, geliştirme sırasında yararlanılan yöntem ve teknikler anlatılmış ve uygulama sonucunda elde edilen bulgular paylaşılmıştır.

### 6.2.1. Problemin Tanımı

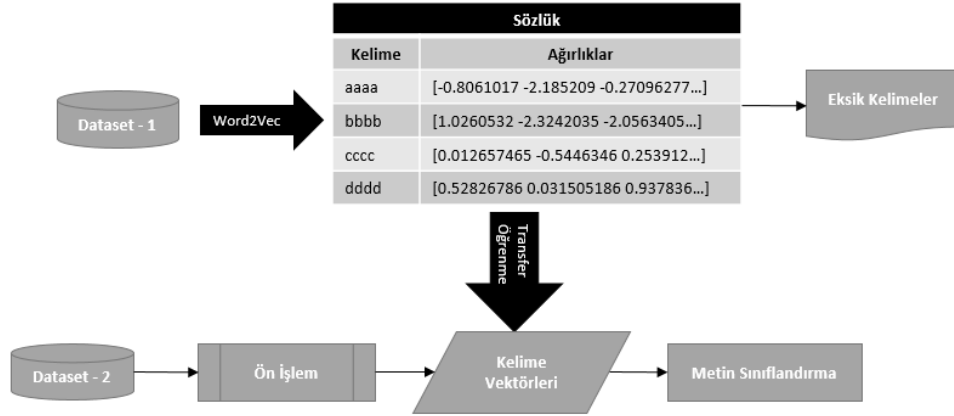
Tez çalışmasında, sınıflandırma işlemi aşamasında kullanılan Dataset-2 çağrı merkezi çalışanları tarafından tutulan kayıtlar ve kullanıcıların internet sayfasını kullanarak gönderdikleri mesajlar ile oluşturulmuş bir veri setidir. Üretilen veriseti gerçek bir veriseti olduğu için birçok yazım hatası bulunmaktadır. Ancak bu hataların düzeltilmesi amacıyla

kullanılan Türkçe için bir yazım denetleyicisi hali hazırda bulunmamaktadır. Dolayısıyla bu durum, uygulamanın başarımını düşürmektedir. Bu problemi çözebilmek amacıyla, üretilen kelime vektörleri kullanılarak yazım hatalarını gidermeye çalışan Türkçe için yeni bir yöntem önerilmiştir. Önerilen bu yöntem *sözlük yöntemi* olarak adlandırılmıştır. Daha sonra çok sınıflı Türkçe bir veri seti olan Dataset-2 üzerinde metin sınıflandırma işlemi uygulanmıştır. Bu veri seti içerisindeki token kelimelerin vektörel değerleri sözlükten transfer öğrenme ile aktarılmıştır. Ancak sözlükte bulunmayan kelimelerin hatalı kelimeler olduğu düşünülerek bir derin sinir ağı mimarisi olan LSTM yöntemi ile bu kelimelerin yerine doğru veya yakın anlamlı kelimeler tahmin edilmeye çalışılmıştır. Bu işlemin ardından metin sınıflandırma uygulamasının doğruluk oranında %8.68 değerinde bir iyileştirme olduğu görülmüştür.

### 6.2.2. Sözlük Yöntemi

Bu yöntemde, Dataset-1 üzerinde Uygulama 3 ile üretilmiş kelime vektörleri kullanılmıştır. Dataset-1 yaklaşık 80 gb boyutunda oldukça büyük bir derlem olduğu için hemen hemen tüm Türkçe kelimeleri kapsamaktadır. Böylece üretilen kelime vektörlerinden Türkçe tüm kelimeleri içeren bir sözlük oluşturulmuştur.

Bu amaçla, çalışmada şöyle bir yöntem önerilmektedir. Sınıflandırma işleminin yapılacağı Dataset-2 üzerinde kelimeler token haline dönüştürüldükten sonra kelimelerin vektör biçimindeki sayısal değerleri sözlükten çekilerek transfer öğrenme yöntemi ile aktarılmıştır. Bu aşamadaki kritik nokta vektörler içindeki değerlerin ilk başta sıfır olarak atanmasıdır. Dolayısıyla sözlük içerisinde yer alan kelimelerin, değerleri çekilerek sıfır olarak tanımlanan vektörler ile değiştirilmiştir ancak sözlük içerisinde yer almayan kelimeler ise sıfır olarak kalmaya devam etmiştir. Yani bir başka deyişle kelimelerin vektörlere dönüştürülme işlemi tamamlandığında sıfırlardan oluşan vektörler sözlükte yer almayan kelimeler olduğu anlamına gelmektedir. Bu durum, oluşturulan sözlük oldukça büyük bir sözlük olduğu ve hemen hemen Türkçede yer alan tüm kelimelerin bulunduğu için bu sözlükte yer almayan kelimelerin hatalı ve yanlış yazılmış kelimeler olduğu şeklinde yorumlanmaktadır. Şekil 6.15.'te yöntemin yapısı gösterilmiştir.



Şekil 6.15. Sözlük yönteminin çalışma yapısı

Çizelge 6.3. Sözlük yöntemi kullanılmadan elde edilen sonuçlar

|                                                              |         |
|--------------------------------------------------------------|---------|
| Sözlükteki Kelime Vektörü Sayısı                             | 847,766 |
| Dataset-2 İçindeki Kelime Sayısı                             | 197,288 |
| Sözlükten Alınan Kelime Vektörü Sayısı                       | 142,047 |
| Sözlükten Alınamayan Kelime Vektörü Sayısı (Eksik Kelimeler) | 55,241  |
| Sözlükten Kelime Vektörü Okunm Oranı                         | 72%     |

Çizelge 6.3. incelendiğinde 197.288 kelimenin sözlükten çekildiği ancak 142.047 adet kelimenin ise sözlükten okunamadığı görülmektedir. Bu durumda dataset içindeki unique kelimelerin %28'inin sözlükten alınamadığını göstermektedir. Dolayısıyla bu kelimeler yanlış yazılmış kelimelerdir.

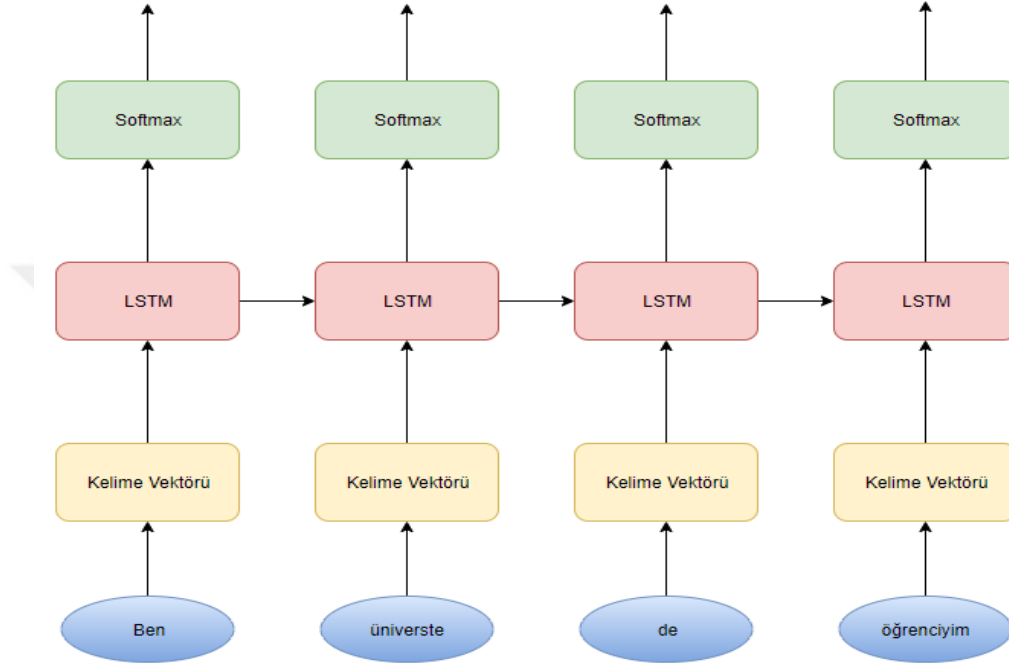
Örnek olarak “Ben üniversite de öğrenciyim.” cümlesi incelenmiştir. Ancak bu cümlede “üniversite” kelimesi “üniverste” şeklinde yanlış yazılmıştır. Cümle token haline dönüştürülüp her bir kelimenin vektörel karşılığı transfer öğrenme yöntemi ile sözlükten çekilmektedir. Dolayısıyla ilk başta sıfır olarak tanımlanmış değerler daha sonra sözlükten çekilmiştir.



Şekil 6.16. Transfer öğrenme ile sözlük kullanımı

Şekil 6.16. incelendiğinde “üniverste” kelimesi yanlış yazılmış bir kelime olduğu için sözlükte bulunamamış ve sözlükten bulunan diğer kelimelerin karşılıkları sözlükten transfer

edilirken “üniverste” kelimesi vektörü sıfır olarak kalmıştır. Bu işlemin veri seti içerisinde yer alan tüm cümleler için uygulandığı düşünülürse kelimelerin vektörlere dönüştürülme işleminin ardından sıfır olarak kalan vektörler ayrıştırılarak yanlış yazılmış kelimeler elde edilmiştir. Daha sonra bu kelimelerin yerine doğru veya doğru kelimeye en yakın kelimeler LSTM yöntemi ile tahmin edilmeye çalışılmıştır.



Şekil 6.17. LSTM ile kelime tahmini

Önerilen yöntemde kullanılan LSTM modelinin yapısı Şekil 6.17.’de gösterilmiştir. Geliştirilen modelde 256 nöron kullanılmıştır. LSTM katmanından sonra aşırı öğrenmeyi engelleyebilmek amacıyla dropout yerleştirilmiş ve değeri 0.2 olarak tercih edilmiştir. Çıkış değerini hesaplamak için ise softmax fonksiyonu kullanılmıştır.

| model.similar_word("üniverste")                                                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [('üniversite', 0.6658480167388916),<br>('fakülte', 0.621586263179779),<br>('okul', 0.6204925179481506),<br>('konservatuar', 0.6177264451980591),<br>('ilahiyat', 0.6013895869255066)] |

Şekil 6.18. Benzer kelimelerin listesi

Şekil 6.18. incelendiğinde yanlış yazılmış “üniverste” kelimesi tespit edildikten sonra LSTM yöntemi ile bu kelimenin yerine gelebilecek kelimeler tahmin edilmeye çalışılmıştır. Bu adımda ki amaç doğru kelimenin veya doğru kelimeye en yakın anlamlı kelimelerden birinin bu yanlış yazılmış kelime ile değiştirilmesidir. Şekilde “üniverste” kelimesine alternatif olarak tahmin edilmiş kelimeler görülmektedir. Önerilen yöntem ile bu kelimenin doğru yazımı olan üniversite kelimesi ilk sırada tahmin edilmiştir. Daha sonra gelen kelimelerinde bu kelime ile benzer anlamda olan kelimeler olduğu görülmüştür. Yani model bu durumda başarılı bir performans göstermiştir.

Sözlük metodu olarak adlandırılan bu yöntem tüm veri setine (Dataset-2) uygulandığında veri seti içerisinde yer alan ancak yazım yanlışından dolayı sözlükten çekilemeyen 55.241 kelime için uygulanmıştır. Eksik kelimelerin yerine doğru veya yakın anlamlı kelimeler tahmin edilmeye çalışılmıştır. Sonuçlar Çizelge 6.4.’te verilmiştir.

Çizelge 6.4. Sözlük yöntemi kullanılarak elde edilen sonuçlar

|                                               |               |
|-----------------------------------------------|---------------|
| Sözlük Yöntemi Kullanılmadan Doğruluk Değeri  | % 77.12       |
| Sözlük Yöntemi ile Değiştirilen Kelime Sayısı | 55241         |
| Sözlük Yöntemi ile Doğruluk Değeri            | % 85.80       |
| <b>Performans Artırımı</b>                    | <b>% 8.68</b> |

Çizelge 6.4. incelendiğinde sözlük yöntemi kullanılmadan sadece LSTM yöntemi ile sınıflandırılma yapılmış % 77.12 doğruluk oranı elde edilmiştir. Daha sonra sözlük yöntemi kullanılarak 55241 kelime tahmin edilmiş bu kelimelerinde vektörel karşılıklarının alınmasının ardından doğruluk değeri % 85.80 olarak hesaplanmıştır. Böylece % 8.68 gibi önemli bir oranda modelin performansı geliştirilmiştir.

### 6.2.3. Uygulamanın Sonuçları

Tez çalışması kapsamında yeni bir yöntem olarak önerilen sözlük yöntemi adı verilmiş bir metot sunulmuştur. Gerçek ortamlardan toplanan veriler genellikle kirlidir. Bu kirlilik metin işleme çalışmaları için genellikle yazım hatalarıdır. Bu durum, çalışmaların doğruluk oranını önemli ölçüde azaltan bir problemdir. Bu soruna bir çözüm üretmek için geliştirilen sözlük yöntemiyle, birçok yazım hatası içeren kirlili bir gerçek veri seti üzerinde bir metin sınıflandırma uygulaması yapılmıştır. Bu uygulama sonucunda, doğruluk oranı % 8.66 artmıştır.

Günümüzde, internet üzerinden birçok veri sağlandığı ve bu verilerin çoğunun yazım hataları içeren kirli veriler olduğu göz önüne alınırsa, bu yöntem ile veri kalitesinin artırılarak metin işleme çalışmalarına önemli bir katkı sağlayacağı düşünülmektedir.



## 7. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, büyük veri teknolojileri ile derin öğrenme mimarileri kullanılarak metin işleme çalışmaları yapılmış ve sonuçlar detaylı bir şekilde analiz edilmiştir. İki yeni Türkçe veri seti ve kelime vektörleri üretilmiş ayrıca doğruluk değerini geliştirmek için iki özgün yöntem önerilerek literatüre ve araştırmacılara katkı sağlanması amaçlanmıştır.

Metin işleme çalışmaları incelendiğinde en çok üzerinde çalışılan dilin İngilizce olduğu bunun yanı sıra Çince, Almanca, İspanyolca gibi diğer dillerde de akademik çalışmaların olduğu görülmüştür. Türkçe çalışmaların sayısı ise son yıllarda bu alanda çalışmaya başlayan araştırmacılar sayesinde artmaya başlasa da hala çalışmaların yeterli sayıda olmadığı tespit edilmiştir. Bu durumun temel nedeninin, üzerinde çalışılabilecek kapsamlı ve büyük ölçekli Türkçe veri seti eksikliğinin olduğu fark edilmiştir. Bu problemin çözümüne katkı sunmak amacıyla Dataset-1 ve Dataset-2 adında iki yeni Türkçe veri seti üretilmiştir.

Başta Word2Vec olmak üzere sinir ağı temelli kelime temsil yöntemlerinin ortaya çıkmasıyla bu alanda büyük gelişmeler yaşanmıştır. Bu yöntemler ile eğitilen kelime vektörleri metin işleme çalışmalarına önemli katkılar sağlamıştır. Ancak yine Türkçe için üretilmiş büyük ölçekli denilebilecek Türkçe kelime vektörlerinin olmadığı görülmüştür. Bu nedenle ilk veri seti Dataset-1 bu vektörlerin eğitilmesi amacıyla üretilmiştir. Dataset-1, yaklaşık 80 gb boyutunda, Türkçe Wikipedia makalelerinden oluşturulmuş en büyük Türkçe derlemidir.

İnsanların öğrenme stili düşünülerek, kelime vektörlerinin üretildikleri derlemin sadece belli bir konudan ve konseptten oluşmuş içeriklerden olmaması gerektiği bunun yerine derlemin birçok başlık içeren genel bir yapıda olmasıyla öğrenmenin daha iyi gerçekleşeceği öngörülmüştür. Bu sayede kelimeler arasındaki yakınlık ve uzaklıkların daha iyi bir şekilde ortaya çıktığı anlaşılmıştır. Elde edilen sonuçlarında bu durumu destekler nitelikte olduğu görülmüştür.

Kelime vektörleri oluşturulması için geliştirilen model üzerinde çeşitli parametrelerle ayarlamalar yapmak mümkündür ve sonuçları etkilemektedir. Ancak kelime vektörlerinin eğitilmesinde ki en önemli hususun, kullanılan derlemin büyüklüğü ve çeşitliliği olduğu çalışmalar sonucunda ortaya çıkmıştır.

Yapılan deneyler, üzerinde çalışılan verilerin büyüklüğü göz önüne alındığında klasik yöntemler ile verilerin işlenmesinin özellikle de kelime vektörlerinin eğitilmesi gibi ağır işlem yükü içeren işlemlerin yapılmasının neredeyse imkânsız olduğunu ortaya çıkarmıştır. Bu nedenle tez çalışmasında büyük veri teknolojilerinin kullanılmasına karar verilmiştir. DeneySEL

alışmalar ile Apache Spark teknolojisi tercih edilmiş böylece günler belki de haftalar sürebilecek eğitim süreci daha hızlı ve daha verimli bir şekilde yapılmıştır.

Metinlerin sınıflandırılması aşamasında ise yine üzerinde çalışılan veri setlerinin büyüklüğü dikkate alınarak derin öğrenme yaklaşımları kullanılmıştır. Derin öğrenme, özellikle grafik kartları başta olmak üzere donanımlarda meydana gelen gelişmeler ile birlikte son yıllardaki çalışmalarda oldukça tercih edilen ve günümüzde yaşanan yapay zekâ alanındaki gelişmelerin temelinde yer alan önemli bir yaklaşımdır. Özellikle büyük veri üzerinde, birçok çalışma alanındaki problemlere başarılı çözümler getirmiştir. Bu nedenle metinlerin sınıflandırılmasında çeşitli derin öğrenme mimarileri kullanılmış, modellerin elde ettiği başarımlar ve parametre seçimlerinin bu başarımlara etkileri incelenmiştir.

Metin sınıflandırma için, üretilen ikinci Türkçe veri seti olan çok sınıflı (10 sınıf) Dataset-2 kullanılmıştır. Bu aşamada, çeşitli derin öğrenme mimarileri ile birlikte önceden eğitilmiş kelime vektörleri olarak Dataset-1 üzerinde eğitilmiş kelime vektörleri ile çalışılmıştır. Metin sınıflandırma problemi için, önceden eğitilmiş kelime vektörleri, transfer öğrenme tekniği ile kullanılmış ve bu yöntemle, derin öğrenme mimarileri ile elde edilen başarımlar hemen hemen tüm modeller için %5-%7 arasında geliştirilmiştir. En başarılı model ise doğruluk değeri % 78.85 ‘ten % 85.82’ye çıkarılarak, % 6.97 oranında gelişme sağlanan GRU modeli olmuştur. Böylece önceden eğitilmiş kelime vektörlerinin sınıflandırma başarımına etkisinin ne kadar önemli olduğu görülmüştür. Ayrıca derin öğrenme modelleri geliştirilirken deneyler sırasında modelin katman ve nöron sayısının seçimi, öğrenme hızı gibi hiper-parametrelerin seçiminin modellerin başarımlarını önemli derecede etkilediği anlaşılmıştır.

Deneyssel çalışmalarda görülmüştür ki iyi eğitilmiş kelime vektörleri ile birlikte en başarılı sınıflandırma modelleri uygulansa da sınıflandırma aşamasında kullanılan veri seti gürültüler içeren kirli bir veri seti olduğu takdirde başarımlar belli bir noktaya kadar geliştirilebilmektedir. Bu çalışmada, sınıflandırma aşamasında kullanılan Dataset-2’de gerçek ortamda üretildiği için birçok yazım hatası içeren kirli bir veri setidir. Bunun da sınıflandırma başarımını düşüren olumsuz bir durum olduğu fark edilmiştir. Ancak literatürde Türkçe dili için geliştirilmiş bir yazım denetleyicisi bulunmamaktadır. Bu nedenle tezin önemli katkılarından biride, yanlış yazılmış kelimelerin tespit edilerek doğru kelimeler ile değiştirilmesine dayanan ve sözlük yöntemi adı verilmiş yeni bir yöntemin literatüre kazandırılmasıdır. Bu yöntemin uygulanması ile birlikte önce Dataset-2 üzerinde yaklaşık 55 bin hatalı yazılmış kelime tespit edilmiş daha sonra LSTM modeli kullanılarak bu kelimeler, yerlerine önerilen kelimeler ile

değiştirilmiştir. Uygulama sonunda mevcut doğruluk değeri önerilen bu yöntem ile %8.68 oranında geliştirilmiştir. Böylece veri kalitesinin artırılmasının da başarımı etkileyen bir diğer önemli unsur olduğu anlaşılmıştır.



## 8. KAYNAKLAR

- [1] K. Doğan and S. Arslantekin, *Büyük Veri: Önemi, Yapısı ve Günümüzdeki Durum. Ankara Üniversitesi Dil ve Tarih-Coğrafya Fakültesi Dergisi*, (2016) 56(1).
- [2] Wu, X., Zhu, X., Wu, G. Q., & Ding, W. *Data mining with big data. IEEE Transactions On Knowledge And Data Engineering*, (2013) 26(1), 97-107.
- [3] A. De Mauro, M. Greco and M. Grimaldi, What is big data? A consensual definition and a review of key research topics. In AIP conference proceedings (2015, February) (Vol. 1644, No. 1, pp. 97-104). AIP.
- [4] Anonymous. (2019). <https://visual.ly/community/infographic/technology/big-data-explosion> (on-line access on 15 Sep, 2019)
- [5] Anonymous. (2019). <https://www.forbes.com/sites/louiscolombus/2017/05/13/ibm-predicts-demand-for-data-scientists-will-soar-28-by-2020/#726b70297e3b> (on-line access on 10 Sep, 2019)
- [6] S. E. Seker, *Metin Madenciliği (Text Mining). YBS Ansiklopedi*, 2(3), 30-32 (2015).
- [7] M. F. Amasyali, H. Tasköprü and K. Çaliskan,). Words, Meanings, Characters in Sentiment Analysis. In 2018 Innovations in Intelligent Systems and Applications Conference (2018, October) (ASYU) (pp. 1-6) IEEE.
- [8] S. Yıldırım and T. Yıldız, *A Comparison of Different Approaches to Document Representation in Turkish Language. Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, (2018) 22(2), 569-576.
- [9] B. Ay Karakuş, M. Talo, İ. R. Hallaç and G. Aydın, *Evaluating deep learning models for sentiment classification. Concurrency and Computation: Practice and Experience*, (2018) 30(21).
- [10] G. Şahin, Turkish document classification based on Word2Vec and SVM classifier. In 2017 25th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4), (2017). IEEE.
- [11] D. Ayata, M. Saraçlar and A. Özgür, Turkish tweet sentiment analysis with word embedding and machine learning. In 2017 25th Signal Processing and Communications Applications Conference (SIU) (2017) (pp. 1-4). IEEE.
- [12] H. Polat, H and M. Körpe, *TBMM Genel Kurul Tutanaklarından Yakın Anlamalı Kavramların Çıkarılması. Bilişim Teknolojileri Dergisi*, (2018) 11(3), 235-244.
- [13] Ç. Acı and A. Çırak, *Türkçe Haber Metinlerinin Konvolüsyonel Sinir Ağları ve Word2Vec Kullanılarak Sınıflandırılması. Bilişim Teknolojileri Dergisi*, (2019) 12(3), 219-228.
- [14] B. Erşahin, Ö. Aktaş, D. Kilinc and M. Erşahin, *A hybrid sentiment analysis method for Turkish. Turkish Journal of Electrical Engineering & Computer Sciences*, (2019) 27(3), 1780-1793.
- [15] M. F. Amasyali and B. Diri, Automatic Turkish text categorization in terms of author, genre and gender. In International Conference on Application of Natural Language to Information Systems, (2006) (pp. 221-226). Springer, Berlin, Heidelberg.
- [16] D. Kılınç, A. Özçift, F. Bozyigit, P. Yıldırım, F. Yücalar and E. Borandag, “TTC-3600: A new benchmark dataset for Turkish text categorization”. *Journal of Information Science*, 43(2), 174-185, (2015).
- [17] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proceedings of the IEEE, (1998), 86(11) 2278-2324.
- [18] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning. International Conference on Machine Learning (ICML), (2008), 160-167.

- [19] Y. Kim, Convolutional neural networks for sentence classification, In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), (2014), (pp. 1746-1751).
- [20] X. Zhang, J. Zhao and Y. Lecun, *Character-level convolutional networks for text classification*. **Advances in Neural Information Processing Systems**, (2015) 649-657.
- [21] S. Lai, L. Xu, K. Liu and J. Zhao, Recurrent convolutional neural networks for text classification. AAAI Conference on Artificial Intelligence, (2015), 2267-2273.
- [22] R. J. Williams and D. Zipser, *A learning algorithm for continually running fully recurrent neural networks*. **Neural Computation**, (1989) 1(2), 270-280.
- [23] D. Ravi, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo and G. Z. Yang, *Deep learning for health informatics*. **IEEE journal of Biomedical and Health Informatics**, (2017) 21(1), 4-21.
- [24] S. Hochreiter and J. Schmidhuber, *Long short-term memory*. **Neural Computation**, (1997) 9(8) 1735-1780.
- [25] Y. Cho and L. K. Saul, *Kernel methods for deep learning*. **Advances in Neural Information Processing Systems**, (2009) 342-350.
- [26] J. Y. Lee and F. Deroncourt, Sequential short-text classification with recurrent and convolutional neural networks, In Proceedings of NAACL-HLT, (2016), (pp. 515-520).
- [27] N. Kalchbrenner, E. Grefenstette and P. Blunsom, A Convolutional Neural Network for Modelling Sentences, In Proceedings of ACL, (2014).
- [28] A. Severyn, A and A. Moschitti, Twitter Sentiment Analysis with Deep Convolutional Neural Networks In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, (2015), (pp. 959-962).
- [29] A. Hassan and A. Mahmood, Deep Learning Approach for Sentiment Analysis of Short Texts, 3rd International Conference on Control, Automation and Robotics (ICCAR), (2017), 705-710.
- [30] S. Yang, and Z. Xia, A convolutional neural network method for Chinese document sentiment analyzing. 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, (2016), 308-312.
- [31] Q. Huang, Q. X. Zheng and Z. Dong, Deep Sentiment Representation Based on CNN and LSTM, International Conference on Green Informatics (ICGI), (2016), 30-33.
- [32] P. Vateekul, and T. Koomsubha, A study of sentiment analysis using deep learning techniques on Thai Twitter data, 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, (2016), 1-6.
- [33] Q. Vo, H. Nguyen, B. Le and M. Nguyen, Multi-channel LSTM-CNN model for Vietnamese sentiment analysis, 9th International Conference on Knowledge and Systems Engineering (KSE), Hue, (2017), 24-29.
- [34] Z. Su, H. Xu, D. Zhang and Y. Xu, Chinese Sentiment classification using a neural network tool—Word2Vec. Paper presented at the 2014 International Multisensor Fusion and Information Integration for Intelligent Systems Conference. Beijing, China. (2014).
- [35] M. Hughes, I. Li, S. Kotoulas and T. Suzumura, *Medical text classification using convolutional neural networks*. In Randell, R., Cornet, R., McCowan, C., Peek, N., & Scott, P. (Eds.), **Informatics for Health: Connected Citizen-Led Wellness and Population Health** (pp. 246-250) (2017).
- [36] S. Sohngir, D. Wang, A. Pomeranets and T. M. Khoshgoftaar, *Big data: Deep learning for financial sentiment analysis*. **Journal of Big Data**, (2014) (vol. 5, no. 1, p. 3).

- [37] A. G. Shoro and T. R. Soomro, “*Big data analysis: Ap Spark perspective*”, Double Blind Peer Reviewed **International Research Journal**, USA, 0975-4350 (2015).
- [38] S. E. Seker, *Büyük Veri ve Büyük Veri Yaşam Döngüleri*, **YBS Ansiklopedi**, (2015) 2(3) 10-17.
- [39] S. Lohr, *The age of big data*. **New York Times**, 11(2012).
- [40] E. Aktan, *Büyük veri: Uygulama alanları, analitiği ve güvenlik boyutu*. **Bilgi Yönetimi**, (2015) 1(1), 1-22.
- [41] F. Ünal, *Büyük Veri ve Semantik*, Abaküs Yayınevi, (2015).
- [42] Y. Demchenko, P. Membrey, P. Grosso and C. de Laat, Addressing Big Data Issues in Scientific Data Infrastructure. First International Symposium on Big Data and Data Analytics in Collaboration (BDDAC 2013). Part of The 2013 Int. Conf. on Collaboration Technologies and Systems (CTS 2013), (2013).
- [43] G. Yavuz, S. Aytekin and M. Akçay, *Apache Hadoop Ve Dağıtık Sistemler Üzerindeki Rolü*. **Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü Dergisi**, (2012) (027) 43-54.
- [44] G. Z. Özen, M. Tekerek and R. Sultanov, *Hadoop framework implementation and performance analysis on a cloud*. **Turkish Journal of Electrical Engineering & Computer Sciences**, (2017) 25(2) 705-716.
- [45] Muhammed Emre Çolak. *MapReduce ile metasezgisel optimizasyon/Metaheuristic optimization with MapReduce*. Yüksek Lisans Tezi, Fırat Üniversitesi Türkiye, 2015.
- [46] Anonymous. (2019). <https://spark.apache.org/docs/latest/ml-classification-regression.html>, (on-line access on 11 Oct, 2019)
- [47] İ. Ü. Oğul, C. Özcan and Ö. Hakdağlı, Fast text classification with Naive Bayes method on Apache Spark. In 2017 25th Signal Processing and Communications Applications Conference (SIU) (2017) (pp. 1-4). IEEE.
- [48] X. Meng, X. J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu and D. Xin, *Mllib: Machine learning in apache spark*. **The Journal of Machine Learning Research**, (2016) 17(1) 1235-1241.
- [49] Barış Akgün. *Apache spark tabanlı destek vektör makineleri ile akan büyük veri sınıflandırma*. Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Türkiye, 2016.
- [50] İ. Ü. Oğul, C. Özcan and Ö. Hakdağlı, Spark Destek Vektör Makinesi ile Metin Sınıflandırma, 1. Ulusal Bulut Bilişim ve Büyük Veri Sempozyumu B3S’17, (2017).
- [51] B. Karabay and M. Ulaş, Büyük Veri İşlemede Yaygın Kullanılan Araçların Karşılaştırılması, 8th International Advanced Tehnologies Symposium, (2017).
- [52] T. Atanasova, M. Kasheva, S. Sulova and J. Vasilev, Analysis of the possible application of Data Mining, Text Mining and Web Mining in business intelligent systems. In The 33rd International Convention MIPRO, (2010) (pp. 1294-1297). IEEE.
- [53] E. Adalı, E. *Doğal Dil İşleme*. **Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi**, (2012) 5(2).
- [54] B. Diri, *Doğal Dil İşleme (DDİ) Natural Language Processing (NLP)*.
- [55] J. S. Street, W. S. Clark, K. S. Gannon, J. L. Cummings, F. Bymaster, R. Tamura and G. Tollefson, *Olanzapine treatment of psychotic and behavioral symptoms in patients with Alzheimer disease in nursing care facilities: a double-blind, randomized, placebo-controlled trial*. **Archives of General Psychiatry**, (2000) 57(10) 968-976.
- [56] V. Srividhya and R. Anitha, *Evaluating preprocessing techniques in text categorization*. **International journal of computer science and application**, (2010) 47(11), 49-51.

- [57] R. Duwairi and M. El-Orfali, *A study of the effects of preprocessing strategies on sentiment analysis for Arabic text*. **Journal of Information Science**, (2014) 40(4) 501-513.
- [58] M. Y. Pak and S. Gunal, *The impact of text representation and preprocessing on author identification*. **Anadolu Üniversitesi Bilim Ve Teknoloji Dergisi A-Uygulamalı Bilimler ve Mühendislik**, (2017) 18(1) 218-224.
- [59] E. Kiliç, N. Ateş, A. Karakaya and D. Ö. Şahin, Two new feature extraction methods for text classification: TESDF and SADP. In 2015 23rd Signal Processing and Communications Applications Conference (SIU), (2015), (pp. 475-478). IEEE.
- [60] S. Weiss, N. Indurkha, T. Zhang and F. Damerau, *Text mining: predictive methods for analyzing unstructured information*. **Springer Science & Business Media**, (2010).
- [61] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah and D. C. L. Ngo, *Text mining for market prediction: A systematic review*. **Expert Systems with Applications**, (2014) 41(16) 7653-7670.
- [62] M. U. Salur and İ. Aydın, Türkçe Metin Sınıflandırmada Metin Önışlemlerin Evrişimsel Yapay Sinir Ağı Sınıflandırma Başarımı Üzerindeki Etkisi, International Conference on Artificial Intelligence and Data Processing (IDAP), (pp. 1-4) (2018)
- [63] S. C. Yücebaş and R. Tintin, Gövde-Türk: A Turkish stemming method. In 2017 International Conference on Computer Science and Engineering (UBMK), (2017) (pp. 343-347). IEEE.
- [64] T. Schnabel, I. Labutov, D. Mimno and T. Joachims, Evaluation methods for unsupervised word embeddings. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, (2015) (pp. 298-307).
- [65] M. U. Sen and H. Erdogan, Learning word representations for Turkish. In Signal Processing and Communications Applications Conference (SIU), 2014 22nd (2014) (pp. 1742-1745). IEEE.
- [66] F. Gözükarar and S. A. Özel, *Türkçe ve İngilizce Yorumların Duygu Analizinde Doküman Vektörü Hesaplama Yöntemleri için Bir Deneyisel İnceleme*. **Çukurova Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi**, (2016) 31(2) 464-482.
- [67] O. Güngör and E. Yıldız, Linguistic features in Turkish word representations. In Signal Processing and Communications Applications Conference (SIU), (2017) (pp. 1-4). IEEE.
- [68] G. Şahin, Turkish document classification based on Word2Vec and SVM classifier. In Signal Processing and Communications Applications Conference (SIU), (2017) (pp. 1-4). IEEE.
- [69] T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient estimation of word representations in vectorspace”, arXiv:1301.3781, (2013).
- [70] Q. Le and T. Mikolov T, Distributed representations of sentences and documents. 31th International Conference on Machine Learning, (2014).
- [71] T. Mikolov, I. K. Sutskever, G. Corrado, J. Dean, Distributed Representations of Words and Phrases and their Compositionality, AAAI Spring Symposium AI Technologies for Homeland Security 200591-98, vol. cs.CL, pp. 3111–3119, (2013).
- [72] H. C. Wu, R. W. P. Luk, K. F. Wong and K. L. Kwok, *Interpreting tf-idf term weights as making relevance decisions*. **ACM Transactions on Information Systems (TOIS)**, (2008) 26(3) 13.
- [73] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov, *Bag of tricks for efficient text classification*. arXiv preprint arXiv:1607.01759 (2016).
- [74] Ö. Çoban, B. Özyer and G. T. Özyer, Sentiment analysis for Turkish Twitter feeds. In 2015 23rd Signal Processing and Communications Applications Conference (SIU), (2015) (pp. 2388-2391). IEEE.

- [75] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu and B. Qin, Learning sentiment-specific word embedding for twitter sentiment classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, (Volume 1: Long Papers) (2014) (pp. 1555-1565).
- [76] T. Mikolov, W. T. Yih and G. Zweig, Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (2013) (pp. 746-751).
- [77] L. K. Şenel, V. Yücesoy, A. Koç and T. Çukur, Interpretability analysis for Turkish word embeddings. In 2018 26th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4) (2018). IEEE.
- [78] M. Aydoğan and A. Karcı, *Kelime Temsil Yöntemleri ile Kelime Benzerliklerinin İncelenmesi. Çukurova Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*, (2019) 34(2), 181-196.
- [79] J. Pennington, R. Socher and C. Manning, Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), (2014) (pp. 1532-1543).
- [80] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng and C. D. Manning, Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the conference on empirical methods in natural language processing, (2011) (pp. 151-161).
- [81] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, (2016).
- [82] B. Diri, *Makine Öğrenmesine Giriş (Machine Learning–ML)*, (2012).
- [83] F. Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review*, (1958) 65(6), 386.
- [84] D. Ravi, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo and G. Z. Yang, Deep learning for health informatics. *IEEE journal of biomedical and health informatics*, (2016) 21(1), 4-21.
- [85] I. Goodfellow, Y. Bengio ve A. Courville, “Deep Learning”, MIT Press, Cambridge, 2016.
- [86] F. Chollet, “Deep Learning with Python”, Manning Publications, New York, 2017.
- [87] J. Y. Lee and F. Deroncourt, *Sequential short-text classification with recurrent and convolutional neural networks*, (2016) arXiv preprint arXiv:1603.03827.
- [88] M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks. In European conference on computer vision, (2014) (pp. 818-833).
- [89] Y. Kim, *Convolutional neural networks for sentence classification*, **arXiv preprint** arXiv:1408.5882 (2014).
- [90] M. M. Lopez, J. Kalita, *Deep Learning applied to NLP*, **arXiv preprint** arXiv:1703.03091 (2017).
- [91] A. Severyn, A. Moschitti, *Modeling Relational Information in Question-Answer Pairs with Convolutional Neural Networks*, **arXiv preprint arXiv:1604.01178** (2016).
- [92] C. N. Dos Santos and M. Gatti, Deep convolutional neural networks for sentiment analysis of short texts. International Conference on Computational Linguistics (COLING), (2014) 69-78.
- [93] S. Wermter, G. Arevian and C. Panchev, *Recurrent neural network learning for text routing*, In 1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470) (Vol. 2, pp. 898-903) (1999). IET.
- [94] A. Conneau, H. Schwenk, L. Barrault and Y. Lecun, *Very deep convolutional networks for natural language processing. arXiv preprint* arXiv:1606.01781, (2016).

- [95] N. T. Vu, H. Adel, P. Gupta and H. Schutze, Combining Recurrent and Convolutional Neural Networks for Relation Classification, In Proceedings of NAACL-HLT, (2016) (pp. 534-539).
- [96] R. Messina and J. Louradour, Segmentation-free handwritten Chinese text recognition with LSTM-RNN. In 2015 13th International Conference on Document Analysis and Recognition (ICDAR), (2015) (pp. 171-175). IEEE.
- [97] M. Tan, C. D. Santos, B. Xiang and B. Zhou, *LSTM-based deep learning models for non-factoid answer selection*, **arXiv preprint** arXiv:1511.04108, (2015).
- [98] K. S. Tai, R. Socher and C. D. Manning, Improved semantic representations from tree-structured long short-term memory networks, In: IN PROC. ACL. (2015).
- [99] M. Sundermeyer, R. Schlüter and H. Ney, LSTM neural networks for language modeling. In Thirteenth annual conference of the international speech communication association, (2012).
- [100] Y. Wang, M. Huang and L. Zhao, Attention-based LSTM for aspect-level sentiment classification. In Proceedings of the 2016 conference on empirical methods in natural language processing, (2016) (pp. 606-615).
- [101] C. Zhou, C. Sun, Z. Liu and F. Lau, *A C-LSTM neural network for text classification*, **arXiv Preprint**. arXiv1511.08630 (2015).
- [102] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, In NIPS 2014 Workshop on Deep Learning, (2014).
- [103] D. Tang, B. Qin and T. Liu, Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of EMNLP, (2015) 1422–1432.
- [104] Z. Wu and S. King, Investigating gated recurrent networks for speech synthesis, In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2016) (pp. 5140-5144). IEEE.
- [105] M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, (1997) 45(11) 2673-2681.
- [106] M. Sundermeyer, T. Alkhoul, J. Wuebker and H. Ney, Translation modeling with bidirectional recurrent neural networks, In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014) (pp. 14-25).
- [107] Y. Fan, Y. Qian, F. L. Xie and F. K. Soong, TTS synthesis with bidirectional LSTM based recurrent neural networks. In Fifteenth Annual Conference of the International Speech Communication Association, (2014).
- [108] A. Graves, N. Jaitly and A. R. Mohamed, Hybrid speech recognition with deep bidirectional LSTM. In 2013 IEEE workshop on automatic speech recognition and understanding (pp. 273-278) (2013). IEEE.
- [109] B. Plank, A. Søgaard and Y. Goldberg, *Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss*, **arXiv preprint** arXiv:1604.05529 (2016).
- [110] Z. Huang, W. Xu and K. Yu, *Bidirectional LSTM-CRF models for sequence tagging*, **arXiv preprint**, arXiv:1508.01991 (2015).
- [111] B. Ay Karakuş, *Text Analysis with Deep Learning and Big Data Approaches*, PhD Thesis, Firat University Turkey, 2018.
- [112] Z. Kurt, *Deep learning applications on biological data*, Master's Thesis, Kadir Has University Turkey, 2018.
- [113] M. A. Kızrak and B. Bolat, *Derin Öğrenme ile Kalabalık Analizi Üzerine Detaylı Bir Araştırma*. **Bilişim Teknolojileri Dergisi**, 11(3), 263-286 (2018).

- [114] E. Buber and O. K. Sahingoz, Makine Öğrenmesi sistemi ile görüntü İşleme ve en uygun parametrelerin ayarlanması [Turkish-only]. In 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), (2017) (pp. 1-5). IEEE.
- [115] M. Coşkun, Ö. Yıldırım, A. Uçar and Y. Demir, *An overview of popular deep learning methods*. **European Journal of Technique**, (2017) 7(2), 165-176.
- [116] F. Agostinelli, M. Hoffman, P. Sadowski, P. and P. Baldi, *Learning activation functions to improve deep neural networks*, **arXiv preprint** arXiv:1412.6830 (2014).
- [117] P. Ramachandran, B. Zoph and V. Le, *Searching for activation functions*, **arXiv preprint** arXiv:1710.05941 (2017).
- [118] Y. Cho and L. K. Saul, Kernel methods for deep learning. In Advances in neural information processing systems, (2009) (pp. 342-350).
- [119] Anonymous. (2019). <https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4> (on-line access on 21 Oct, 2019).
- [120] G. Ser and C. T. Bati, *Derin Sinir Ağları ile En İyi Modelin Belirlenmesi: Mantar Verileri Üzerine Keras Uygulaması*, **Yüzüncü Yıl Üniversitesi Tarım Bilimleri Dergisi**, (2019) 29(3) 406-417.
- [121] H. M. Fayek, M. Lech and L. Cavedon, *Evaluating deep learning architectures for Speech Emotion Recognition*. **Neural Networks**, (2017) 92, 60-68.
- [122] Ö. İnik and E. Ülker, *Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri*, **Gaziosmanpaşa Bilimsel Araştırma Dergisi**, (2017) 6(3) 85-104.
- [123] T. Xiao, H. Li, W. Ouyang and X. Wang, Learning deep feature representations with domain guided dropout for person re-identification. In Proceedings of the IEEE conference on computer vision and pattern recognition, (2016) (pp. 1249-1258).
- [124] J. Ba and B. Frey, Adaptive dropout for training deep neural networks. In Advances in Neural Information Processing Systems, (2013) (pp. 3084-3092).
- [125] J. M. Alvarez and M. Salzmann, Learning the number of neurons in deep networks. In Advances in Neural Information Processing Systems, (2016) (pp. 2270-2278).
- [126] A. G. Schwing and R. Urtasun, *Fully connected deep structured networks*. **arXiv preprint** arXiv:1503.02351 (2015).
- [127] K. Janocha and W. M. Czarnecki, *On loss functions for deep neural networks in classification*, **arXiv preprint** arXiv:1702.05659 (2017).
- [128] M. Jiang, Y. Liang, X. Feng, X. Fan, Z. Pei, Y. Xue and R. Guan, (2018). *Text classification based on deep belief network and softmax regression*. **Neural Computing and Applications**, 29(1), 61-70.
- [129] J. Howard and S. Ruder, Universal language model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 328-339) (2018).
- [130] J. Wang, Z. Wang, D. Zhang and J. Yan, Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In IJCAI (2017) (pp. 2915-2921).
- [131] M. Aydoğan and A. Karci, *Improving the accuracy using pre-trained word embedding on deep neural networks for Turkish text classification*. **Physica A: Statistical Mechanics and its Applications**, 123288. (2019).
- [132] Y. Qi, D. S. Sachan, M. Felix, S. J. Padmanabhan and G. Neubig, (2018). *When and why are pre-trained word embeddings useful for neural machine translation*. **arXiv preprint** arXiv:1804.06323.

- [133] T. Mikolov, E. Grave, P. Bojanowski, C. Puhersch and A. Joulin, Advances in pre-training distributed word representations. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (2018).
- [134] E. Grave, P. Bojanowski, P. Gupta, A. Joulin and T. Mikolov, *Learning word vectors for 157 languages*. **arXiv preprint** arXiv:1802.06893 (2018).
- [135] S. M. Rezaeinia, A. Ghodsi and R. Rahmani, *Improving the accuracy of pre-trained word embeddings for sentiment analysis*, **arXiv preprint** arXiv:1711.08609 (2017).
- [136] Anonymous. (2019). [https://www.chip.com.tr/forum/spam-nedir-zararlari-nelerdir-nasil-korunulur\\_t32830.html](https://www.chip.com.tr/forum/spam-nedir-zararlari-nelerdir-nasil-korunulur_t32830.html) (online Access on 10 Oct,2019)
- [137] A. Baltas, A. Kanavos and A. K. Tsakalidis, An apache spark implementation for sentiment analysis on twitter data. In International Workshop of Algorithmic Aspects of Cloud Computing (pp. 15-25) (2016).
- [138] O. Doğan, *Ücretsiz Veri Madenciliği Araçları ve Türkiye’de Bilinirlikleri Üzerine Bir Araştırma*, **Ege Stratejik Araştırmalar Dergisi**, Cilt 8 Sayı1 (2017).
- [139] D. Torunoğlu, E. Çakirman, M. C. Ganiz, S. Akyokuş and M. Z. Gürbüz, Analysis of preprocessing methods on classification of Turkish texts. In 2011 International Symposium on Innovations in Intelligent Systems and Applications (pp. 112-117) (2011). IEEE
- [140] Anonymous. (2019). <https://github.com/ahmetaa/zemberek-nlp> (on-line access on 20 Sep, 2019).
- [141] M. F. Amasyalı, M. Çetin and C. Akbulut, *Metinlerin Anlamsal Uzaydaki Temsil Yöntemlerinin Sınıflandırma Performansına Etkileri*, **Sigma**, 5, 8-14 (2013).
- [142] X. Rong, *Word2Vec Parameter Learning Explained*, **arXiv preprint** arXiv:1411.2738 (2014).
- [143] Anonymous. (2019). [https://tr.wikipedia.org/wiki/Fausto\\_Zonaro](https://tr.wikipedia.org/wiki/Fausto_Zonaro), (online Access, 1 Sep 2019)
- [144] M. Aydoğan and A. Karci, Turkish Text Classification with Machine Learning and Transfer Learning. In 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), (2019) (pp. 1-6). IEEE.
- [145] A. P. Bradley, *The use of the area under the ROC curve in the evaluation of machine learning algorithms*. **Pattern Recognition**, (1997) 30 1145–1159.
- [146] D. J. Hand and R. J. Till, *A simple generalisation of the area under the ROC curve for multiple class classification problems*. **Machine Learning**, (2001) 45 171–186.
- [147] J. Schmidhuber, *Deep learning in neural networks: An overview*. **Neural networks**, (2015) 61, 85-117.

## ÖZGEÇMİŞ

**Ad Soyad** : Murat AYDOĞAN

**Doğum Yeri ve Tarihi** : Elazığ – 18.9.1986

**Adres** : Bingöl Üniversitesi Genç Meslek Yüksekokulu Genç/BİNGÖL

**E-Posta** : maydogan@bingol.edu.tr

**Lisans** : Fırat Üniversitesi Bilgisayar Öğretmenliği (2011)

**Yüksek Lisans** : Fırat Üniversitesi Fen Bilimleri Enstitüsü Elektronik-Bilgisayar Eğitimi ABD (2014)

**Mesleki Deneyimler** : Kafkas Üniversitesi - Uzman (2013-2015)  
Bingöl Üniversitesi - Öğretim Görevlisi (2015-devam ediyor)

### TEZDEN TÜRETİLEN YAYINLAR

AYDOĞAN, M., & KARCI, A. (2018, September). Spam Mail Detection using Naive Bayes method with Apache Spark. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)* (pp. 1-6). IEEE.

AYDOĞAN, M., & KARCI, A. Kelime Temsil Yöntemleri ile Kelime Benzerliklerinin İncelenmesi. *Çukurova Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*, 34(2), 181-196.

AYDOĞAN, M., & KARCI, A. (2019, September). Turkish Text Classification with Machine Learning and Transfer Learning. In *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)* (pp. 1-6). IEEE.

AYDOĞAN, M., & KARCI, A. (2019). Improving the accuracy using pre-trained word embeddings on deep neural networks for Turkish text classification. *Physica A: Statistical Mechanics and its Applications*.