DERİN ÖĞRENME VE BÜYÜK VERİ YAKLAŞIMLARI İLE METİN ANALİZİ

Yük. Müh. Betül AY KARAKUŞ

Doktora Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Galip AYDIN

HAZİRAN–2018

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME VE BÜYÜK VERİ YAKLAŞIMLARI İLE METİN ANALİZİ

DOKTORA TEZİ

Yük. Müh. Betül AY KARAKUŞ

(132129201)

Tezin Enstitüye Verildiği Tarih : 15 Mayıs 2018
Tezin Savunulduğu Tarih : 7 Haziran 2018

Tez Danışmanı : **Doç.Dr. Galip AYDIN (F.Ü)**
Diğer Jüri Üyeleri : **Prof.Dr. Mehmet KAYA (F.Ü.)**
Prof.Dr. İbrahim TÜRKOĞLU (F.Ü.)
Prof.Dr. Ali KARCI (İ.Ü)
Dr.Öğr.Üyesi Muhammed TALO (M.Ü)

HAZİRAN-2018

ÖNSÖZ

Akademik hayatım boyunca, değerli vaktini bana harcayıp bilgi ve tecrübesi ile her türlü imkânı sunan, yanında çalışmaktan onur duyduğum, insani ve ahlaki değerleri ile de örnek edindiğim değerli danışmanım Doç. Dr. Galip AYDIN'a, tecrübelerinden yararlanırken göstermiş olduğu hoş görü ve sabrından dolayı çok teşekkür ederim.

Tez süresince teknik çalışmalarıma verdiği destek ile bilgi ve tecrübemin artmasına katkı sağlayan; proje yürütücülüğünü yaptığım 2140243 numaralı proje için Türkiye Bilimsel ve Teknolojik Araştırma Kurumu'na ve araştırmacı olarak çalıştığım Değirmen isimli proje için Savunma Sanayii Müsteşarlığı'na teşekkürlerimi borç bilirim.

Tez çalışmalarım süresince moral, destek ve katkılarından dolayı Fırat Üniversitesi Büyük Veri ve Yapay Zeka Laboratuvarı ekip arkadaşlarıma teşekkürlerimi sunarım.

Eğitim hayatım boyunca maddi ve manevi desteğini esirgemeyen, ilk hocam ve sevgili babam Osman Ay'a, desteğini hep yanımda hissettiğim sevgili anneme ve kardeşlerime, tezimin hazırlanma süresince göstermiş olduğu sabır ve desteklerinden dolayı sevgili eşime teşekkürlerimi sunarım.

Tezimi varlığı ile hayatımda en büyük destekçim olan biricik kızım Zeynep Farah Karakuş'a ithaf ederim.

Betül AY KARAKUŞ
ELAZIĞ – 2018

İÇİNDEKİLER

ÖZET

Büyük veri analitiği ve derin öğrenme, gelişen dijital dünyada veri biliminin son yıllarda odaklandığı iki önemli araştırma ve çalışma alanıdır. Büyük miktarda ve farklı çeşitlikteki metin verilerini geleneksel yazılım araçları ve teknolojileri kullanarak analiz etmek ve yönetmek zor bir problemdir.

Bu tez çalışmasında büyük veri teknolojileri ve derin öğrenme mimarileri detaylı bir şekilde analiz edilmiş olup dört temel uygulama sunularak akademik katkı sunulması hedeflenmiştir. İlk olarak çağrı merkezleri için bulut tabanlı dağıtık performans analizi ve değerlendirme sistemi geliştirilmiştir. Bu sistemin, iç ve dış çağrı kayıtlarını dağıtık bir şekilde işleyen bulut tabanlı bir performans ölçüm sistemi sunarak müşteri memnuniyeti, satış ve pazarlama, hizmet kalitesi ve performans yönetiminde yüksek bir performans ile önemli bir katkı sağlaması hedeflenmiştir. İkinci uygulamada büyük veri teknolojileri kullanarak Türk dili için dağıtık okunabilirlik analiz sistemi geliştirilmiştir. Türkiye'de eğitim kurumları tarafından kullanılan herhangi bir okunabilirlik uygulaması yoktur ve bu ihtiyaçtan dolayı Türkçe okuma kitaplarını kısa bir sürede analiz edecek okunabilirlik sistemi geliştirilmiştir. Üçüncü uygulamada, farklı mimariler, yöntemler, katmanlar ve hiper parametre optimizasyonları ile oluşturulan derin öğrenme modelleri ile duygu analizi ve haberler veri setinde çok kategorili metin sınıflandırma çalışmaları gerçekleştirilmiştir. Dördüncü uygulamada ise dil bağımsız metin sınıflandırma problemlerinde kullanılabilecek yeni bir Ortalama Doküman Vektörü (ADE) yöntemi sunulmuştur. Önerilen yöntem Türkçe ve İngilizce film yorumlarında duygu sınıflandırması için test edilmiş ve başarılı bir performans göstermiştir.

Türkçe metin sınıflandırma çalışmalarında kullanılabilecek büyük ölçekli bir kıyaslama veri seti yoktur. Bu tez çalışmasının diğer temel katkısı ise bu ihtiyacı karşılamaya yönelik yaklaşık 1 milyon benzersiz kelime içeren Türkçe haberler veri setinin ve 150 bin adet etiketli Türkçe film yorumları veri setinin oluşturulması ve akademik kullanıma açık olarak sunulmasıdır.

Anahtar Kelimeler: Büyük Veri, Derin Öğrenme, Metin Sınıflandırması, Ortalama Doküman Vektörleri, Duygu Analizi, Okunabilirlik Analizi, Çağrı Merkezi, Film Yorumları Sınıflandırma, Haber Sınıflandırma

SUMMARY

Text Analysis with Deep Learning and Big Data Approaches

Big data analytics and deep learning are two significant areas of research and study that data science has focused on in the developing digital world over the last few years. Analyzing and managing large amounts of text data using a variety of traditional software tools and technologies is a difficult problem.

In this thesis, big data technologies and deep learning architectures have been analyzed in detail and four basic applications have been proposed as academic contributions. First, a cloud based distributed performance analysis and evaluation system was developed for call centers. The proposed system aims to provide significant contribution in terms of customer satisfaction, sales and marketing, high quality of service and performance management by offering a cloud based performance measurement system that handles both internal and external call records in a distributed manner. Second, a distributed readability analysis system for the Turkish language was developed using big data technologies. There is no readability application used by educational institutions in Turkey and due to this need, a readability system has been developed to analyze Turkish reading books in a short time. Third, using various deep learning models which are created with different architectures, methods, layers and hyper parameters sentiment analysis and multi-category text classification on news datasets studies are performed. Lastly, a novel Average Document Embedding (ADE) approach is presented which can be used for multi-category language independent text classification. The proposed method has been tested for sentiment classification in Turkish and English movie reviews and has performed well.

There is no large scale benchmark dataset that can be used in Turkish text classification studies. The other main contribution of this thesis is that the Turkish news data set containing about 1 million unique words to meet this need and the creation of the 150,000 labeled Turkish movie reviews dataset, which is made available for academic use.

Key Words: Big Data, Deep Learning, Text Classification, Average Document Embedding, Sentiment Analysis, Readability Analysis, Call Center, Movie Reviews, News Classification

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

SEMBOLLER LİSTESİ

C	: Sınıf numarası
D	: Doküman
M	: Eğitim setindeki toplam doküman sayısı
T	: Test setindeki toplam doküman sayısı
$\boldsymbol{ADE}$	: Ortalama doküman temsili
$\boldsymbol{V}$	: Doküman vektörü
$\boldsymbol{w}$	: Kelime
$\boldsymbol{c}$	: Pencere sayısı
C	: Küme sayısı
N	: Kümelemeden sorumlu verilen bir mapper' ın nokta sayısı
RD	: Disk için okuma süresi
WD	: Disk için yazma süresi
M	: Sistemdeki mapper sayısı
W	: Ağırlık değeri
X	: Veri girdisi
ŷ	: Tahmini çıktı
$\boldsymbol{\sigma(z)}$	: Sigmoid fonksiyon
$\boldsymbol{E}$	: Hata fonksiyonu
$\boldsymbol{\nabla E}$	: Hata fonksiyonunun gradyan değeri
$\boldsymbol{\delta_k}$	: k çıktı birimi için hata değeri
$\boldsymbol{h}$	: Gizli katman
$\boldsymbol{\Delta w_{ij}}$	: Ağırlık güncellemesi
B	: Boyut
A	: Ağ
$\boldsymbol{S_t}$	: t zaman adımındaki durum
$\boldsymbol{O_t}$	: Ağ çıktısı
$\boldsymbol{i_t}$	: İhmal faktörü
$\boldsymbol{b_n}$	: Bias değeri
$\boldsymbol{f_t}$	: Unutma faktörü
$\boldsymbol{E^W}$	: Kelime vektörü
G	: Aktivasyon fonksiyonları
$\boldsymbol{h_F}^{(t)}$	: İleri yayılım katmanının gizli durumu
$\boldsymbol{h_B}^{(t)}$	: Geri yayılım katmanının gizli durumu

KISALTMALAR LİSTESİ

RDDs : Esnek Dağıtık Veri Setleri (Resilient Distributed Datasets)
HDFS : Hadoop Dağıtık Dosya Distemi (Hadoop Distributed File System)
CDH : Cloudera Açık Kaynak Dağıtımı
MR : MapReduce
TF-IDF : Term Frequency-Inverse Document Frequency
LDA : Saklı Dirichlet Ataması (Latent Dirichlet Allocation)
NLP : Doğal Dil İşleme (Natural Language Processing)
CNN : Konvolüsyonel Sinir Ağı (Convolutional Neural Network)
LSTM : Uzun Kısa Süreli Bellek Ağı (Long Short-Term Memory)
RNN : Tekrarlayan Sinir Ağı (Recurrent Neural Network)
BiLSTM : İki yönlü LSTM (Bidirectioanal LSTM)
DBOW : Dağıtık Kelime Torbaları (Distributed Bag of Words)
CBOW : Devamlı Kelime Torbaları (Continuous Bag of Words)
DM : Dağıtık Bellek (Distributed Memory)
ADE : Ortalama Doküman Temsili (Averaged Document Embeddings)
PV : Paragraf Vektörü
PWE : Önceden Eğitilmiş Kelime Temsili (Pre-trained Word Embeddings)
ReLU : Doğrultulmuş Lineer Birim (Rectified Linear Unit)
SGD : Olasılıksal Gradyan Azaltma (Stochastic Gradient Descent)
GPU : Grafiksel İşleme Birimi (Graphics Processing Unit)
GRU : Kapılı Tekrarlı Birim (Gated Recurrent Unit)
PReLU : Parametrik ReLU
DH : Doğrulama hatası
EH : Eğitim hatası
AMI : Amazon Makine Görüntüsü (Amazon Machine Images)
TF : TensorFlow
UPA : Uygulama Programı Arayüzü

1. GİRİŞ

Büyük veri analitiği ve derin öğrenme, gelişen dijital dünyada veri biliminin son yıllarda odaklandığı iki önemli araştırma alanıdır [1,2]. Geleneksel yazılım araçları ve teknolojileri kullanarak analiz edilmesi ve yönetilmesi zor olan ya da mümkün olmayan muazzam büyüklük ve çeşitlilikteki dijital veriler büyük veri olarak adlandırılmaktadır. Büyük veri, pazarlama, dolandırıcılık tespiti, siber güvenlik, ulusal istihbarat, sağlık çalışmaları ve eğitim hizmetleri gibi potansiyel pek çok alanda yaşanan problemlerle ilgili yararlı bilgi içeren çok büyük miktarda veri edinmiş kamu ve özel şirketlerin dikkatini çekmiştir. Büyük veri analitiğinin temel amacı ise, daha etkin iş kararları için şirketlere ve kurumlara yardımcı olmaktır. Google, Microsoft ve IBM gibi şirketler iş analizi ve kararı için geleneksel iş zekâsı ile kullanılamayan çok geniş miktardaki veriyi büyük veri teknolojilerini kullanarak analiz etmektedir. Derin öğrenme, görüntüler, sesler ve metinler gibi verileri anlamlandırmaya yardımcı olan soyutlama ve gösterimin çoklu seviyelerini öğrenmeyi hedefleyen makine öğrenmesi algoritmalarının bir sınıfıdır. 2006 yılından beri geliştirilen derin öğrenme algoritmaları, hiyerarşik mimaride doğrusal olmayan bilginin işlenmesi ile veri gösterimleri olarak yüksek seviyeli ve karmaşık soyutlamaları çıkarmaktadır [3]. Hiyerarşik mimari oluşturmak için düşük seviyeli özelliklerden yüksek seviyeli özellikler türetilmiştir. Bir derin öğrenme algoritması, veriyi daha soyut açılardan gösteren yüksek seviyeli özellikler ile çoklu gösterim seviyelerini bulan öğrenme sürecinin özel bir gösterim çeşididir [4]. Derin öğrenme, NLP, bilgisayar görmesi ve konuşma tanıma gibi büyük veri problemlerini çözmek için akademik ve endüstriyel alanda büyük ilgi görmüştür. Derin öğrenmenin büyük veri analitiği için önemli bir araç olmasının nedeni, çok büyük miktarda gözetimsiz veriyi öğrenebilmesi ve analiz edebilmesidir.

Metin madenciliği olarak da adlandırılan metin analitiği, iş zekâsı, veri analizi, araştırma ve inceleme çalışmaları için metinlerden yararlı ve yeni bilgi çıkarımıdır. Metinlerin sınıflandırılması, dokümanların kümelenmesi, belge özetleme, müşteri ilişkileri yönetimi, web madenciliği, bilgi çıkarımı ve bilgiye erişimi içeren pek çok uygulamada kullanılmaktadır [5–8]. Son yıllarda, derin öğrenme, görüntüler ve videolar üzerinde başarılı sonuçlar vermiş olmasına rağmen metinler üzerinde istenilen başarıyı elde edememiştir [9]. Bir dizi kelimelerden oluşan metinlerdeki büyük sorun, derin öğrenme sistemlerini

kullanmak için bir dilde sayısız kelimelerin olmasıdır. Küresel Dil Monitörünün (Global Language Monitor) 2014 yılında yaptığı tahmine göre İngilizce dilinde yaklaşık bir milyon kelime bulunmaktadır ve bu yapay sinir ağlarının giriş katmanında bir milyon nöronun olmasını gerektirmektedir [10]. LeCun ve arkadaşları [11,12], bu probleme çözüm olarak metni kelimelerden oluşan bir kelime dizisi olarak görmek yerine, karakter dizisi olarak görmüştür. Böylece sayısız kelime yerine muhtemel birkaç düzine karakter ile metin sınıflandırma için derin öğrenme sistemlerini mümkün kılmıştır. Her bir karakteri ikili bir vektörde temsil edilen karakter dizisi geliştirdikleri Konvolüsyonel Sinir Ağları üzerinde uygulanabilen bir görüntü dizisine benzemektedir.

Makine öğrenmesi tekniklerinin gelişmesi ile büyük verilerin işlenmesi ve analizinde daha karmaşık modellerin oluşturulması mümkün hale gelmiştir. Metin analizinde geliştirilen bu modeller n-gram, kelime torbaları (bag of words) ve kelime frekansları (TF-IDF) gibi yaygın yöntemlerde kullanılan basit modellere göre üstünlük sağlamıştır. Son günlerde geliştirilen en başarılı model ise kelimelerin dağıtık gösterimleridir. Saklı Anlam Analizi (Latent Semantic Analysis), Saklı Dirichlet Ataması (Latent Dirichlet Allocation) ya da dağıtık gösterimlerde kelimeler gerçek sayılardan oluşan bir vektör ile temsil edilmektedir. Kelimelerin dağıtık gösterimleri [13], çeşitli yapay sinir ağları tabanlı dil modellerinden elde edilmektedir ve n-gram modellere göre üstünlük sağlamaktadır [14]. Mikolov ve arkadaşları [15], "Sinir Dil Modelleri (Neural Language Models)" olarak adlandırılan bir yöntemler sınıfı ile dağıtık kelime gösterimleri için yeni bir mimari sunmuştur. Geliştirdikleri skip-gram modellerini içeren *word2vec* yazılımı ile basit bir logaritmik doğrusal sınıflandırma ağı oluşturmuştur [16]. Skip-gram model ile çok büyük miktarda yapılandırılmamış metin verisinden yüksek doğrulukta kelime vektörlerinin gösterimlerini öğrenmek için etkili bir yöntemdir.

1.1 Büyük Veri ile Metin Analizinde Derin Öğrenme Uygulamaları

Makine öğrenmesi çalışmalarında yeni bir araştırma alanı olan "Derin Öğrenme (Deep Learning)", beynin derin öğrenmesinden esinlenerek geliştirilmiştir. Son zamanlara kadar, çoğu geleneksel makine öğrenmesi ve sinyal işleme yöntemleri bir ya da iki gizli katman ile doğrusal olmayan veri transferini sağlayan derin olmayan (shallow) mimariler kullanmaktaydı [17]. Derin olmayan mimari örneklerinden bazıları doğrusal veya doğrusal

olmayan dinamik sistemler, destek vektör makineleri, lojistik regresyon, kernel (çekirdek) regresyonu, çok katmanlı perceptron ve Gauss karışımı modelidir. Bu mimariler konuşma, ses, görüntü ve dil tanıma gibi problemlerin çözümünde pek çok zorluk yaşayarak derin mimarilere ihtiyaç duymuştur. Seviyelerine göre derin ve derin olmayan öğrenme mimarileri Şekil 1.1.'de sunulmuştur.

Şekil 1. 1. Seviyelerine göre öğrenme algoritmaları

Derin mimari modellerinin, Destek Vektör Makineleri ve tek gizli katmanlı sinir ağlarından öğrenmesi zor problemler üzerinde daha iyi sonuçlar verdiği deneysel sonuçlardan gözlemlenmiştir [15]. Fakat tüm seviyelerde eğitilebilir ağırlıkları barındıran birden çok gizli katmanla derin mimarilerin parametrelerini öğrenmek zordur. Ayrıca, çok katmanlı sinir ağları için verimli hesaplama modeli olarak bilinen geri yayılım, derin bir mimari model altında çalışırken yüksek hesapsal maliyet, zayıf yerel optimaller ve yetersiz işaretli veri konusunda sıkıntı yaşamaktadır [18]. Derin öğrenmenin bu zorluklarını aşmak için Hinton ve Salakhutdinov [19], çoklu gizli katmanlar ile Derin Fikir Ağları (Deep Belief Networks) oluşturmuştur. Yüksek boyutlu verileri, çoklu bir sinir ağı ile eğiterek düşük boyutlu kodlara çevirmiştir ve yeterince hızlı bilgisayarlar, büyük veri setleri ve iyi bir çözüme yakın başlangıç ağırlıkları ile geri yayılımlı öğrenme algoritmalarını iyileştirmiştir.

Doküman özetleme ve doküman konusu (topic) belirleme doğal dil işlemede bilinen en zor problemlerden birisidir. Literatürde, Özyinelemeli Sinir Ağ, Konvolüsyonel Sinir Ağ modeli ve diğer derin ağ modelleri kullanarak otomatik doküman özetleme, cümle çıkarımı ve konu çıkarımı gerçekleştiren pek çok çalışma yapılmıştır [20–22]. Yapılan çalışmalar göstermiştir ki, derin öğrenme modelleri, işaretli eğitim verisi kullanan Destek Vektör

Makineleri, Destek Vektör Regresyonu ve graf tabanlı diğer modellere [23,24] göre avantaj sağlamaktadır. Çünkü gerçek dünya uygulamalarında veri kaynaklarının çoğu işaretsizdir ve bu nedenle işaretli bilgi her zaman yetersiz olacaktır. İşaretli eğitim verileri üzerinde yapılan doküman özetleme ve konu çıkarma uygulamaları bu nedenle zaman ve maliyet açısından dezavantaj oluşturmaktadır.

Derin öğrenme ile metin analizi için son zamanlarda geliştirilen en başarılı model ise kelimelerin dağıtık gösterimidir. Kelimelerin dağıtık gösterimleri, çeşitli yapay sinir ağları tabanlı dil modellerinden elde edilmektedir ve n-gram modellere göre üstünlük sağlamaktadır. Mikolov [25], Sinir Dil Modelleri olarak adlandırılan bir yöntemler sınıfı ile kelimelerin bir öğrenme tablosu ile sayısal değerlere dönüştürüldüğü dağıtık kelime gösterimleri için yeni bir mimari sunmuştur. Geliştirdikleri *word2vec* yazılımı ile basit bir logaritmik doğrusal sınıflandırma ağı oluşturmuştur. Pek çok doğal dil işleme uygulamalarında, kelimeler sıklıklar TF-IDF puanları ile gösterilmektedir. Bu puanlar bir dokümandaki bir kelimenin önemi hakkında bilgi verirken, kelimenin semantik anlamı hakkında bilgi vermez. Sinir ağ modellerinin bir sınıfı olarak adlandırılan Word2Vec, semantik bilgiyi çıkarmak amacıyla belirli bir doküman, paragraf ya da cümle içindeki her kelime için bir vektör oluşturmaktadır. Bu vektörlerin kullanımının faydaları iki yolla ifade edilebilir:

1. Kelime vektörleri arasında kosinüs benzerlik hesaplanarak iki kelime arasındaki semantik benzerlik ölçülebilir.
2. Doküman sınıflandırma, metinden bilgi çıkarımı ve duygusal analiz gibi çeşitli gözetimli Doğal Dil İşleme (Natural Language Processing-NLP) uygulamaları için bu kelime vektörleri kullanılabilir.

Derin öğrenme, görüntüler, sesler ve metinler gibi verileri anlamlandırmaya yardımcı olan soyutlama ve gösterimin çoklu seviyelerini öğrenmeyi hedefleyen makine öğrenmesi algoritmalarının bir sınıfıdır. Literatürde geçen temel derin öğrenme terimleri Tablo 1.1'de sunulmuştur.

Derin öğrenme ile çıkarılan gösterimler, karar verme, anlamsal indeksleme, bilgiye erişim için pratik bir bilgi kaynağı olarak düşünülebilir ve karmaşık veri daha yüksek soyutlamalarda gösterildiğinde büyük veri analizi için basit doğrusal modelleme teknikleri

olarak düşünülebilir. Düşük seviyeli özelliklerden, yüksek seviyeli özellikler türeterek verileri sınıflandırmayı ve tanımayı amaçlayan derin öğrenme algoritmaları, geleneksel makine öğrenme yöntemlerinin aksine, muazzam büyüklükteki veriler üzerinde problemleri çözmede ve bu verilerin analizinde büyük bir avantaj sağlamaktadır.

Tablo 1. 1. Bazı temel derin öğrenme terminolojileri [26]

Derin Öğrenme	Büyük verilerin analizi, sınıflandırılması ve gözetimli-gözetimsiz öğrenilmesinde kullanılan hiyerarşik mimarilerde çok katmanlı bilgi işleme için gerçekleştirilen makine öğrenmesi yöntemlerinin bir sınıfıdır.
Derin Fikir Ağı	Çok katmanlı stokastik, gizli değişkenlerden oluşan Olasılıklı Üretken Modeller'dir.
Boltzmann Makinesi	Stokastik kararlar veren sinirlere simetrik bağlantılı bir ağdır.
Kısıtlanmış Boltzmann Makinesi	Görünür birim ve gizli birim katmanlarından oluşan ve sinirlerin birbirine tamamen bağlı olmadığı özel bir Boltzmann makinesidir.
Derin Boltzmann Makinesi	Sadece komşu katmanların bağlantılı olduğu (aynı katmanda görülebilir birimler ve gizli birimler kendi aralarında bağlantılı değildir), derin birimlerin derin bir katman şeklinde düzenlendiği özel bir Boltzmann makinesidir.
Derin Sinir Ağı	Birbirine tamamen bağlantılı çok sayıda gizli katmandan oluşan çok katmanlı sinir ağ modelidir.
Derin Otomatik Kodlayıcı	Öğrenmeyi düzenlemek için genellikle bozuk eğitim verilerini kullanarak ya da önceden eğitilmiş derin fikir ağlarını veri girişi alarak çıkış üreten derin sinir ağlarıdır.
Dağıtık Gösterim	Sinir ağı dil modellerinde gözlemlenen verinin bir gösterimidir.

1.2 Tezin Yapısı

Bu tez çalışmasının ikinci bölümünde, büyük veri kavramı tartışılmış, ilgili teknolojiler ve yaklaşımlar detaylı olarak incelenmiştir. Makine öğrenmesi kavramı ve büyük veri analizi için kullanılan dağıtık makine öğrenmesi algoritmaları tartışılmış ve karşılaştırılmıştır. Üçüncü bölümde, büyük veri teknolojileri kullanılarak geliştirilen Dağıtık Okunabilirlik Sistemi ve Çağrı Merkezleri için Dağıtık Performans ve Analiz Sistemlerinin detayları verilmiştir. Dördüncü bölümde, yapay sinir ağları ve derin öğrenme kavramları, derin sinir ağı mimarileri ve derin öğrenme zaman çizelgeleri sunulmuştur. Beşinci bölümde, tez çalışmasındaki modellerin geliştirme ortamları ve platformları, model hiper parametre

ayarlamaları ve seçimleri, veri setleri ön işleme adımları ve modelin aşırı öğrenme problemlerinin çözümü için gerekli kontroller ve yaklaşımlar anlatılmıştır. Altıncı bölümde, farklı mimariler ve parametre seçimleri ile oluşturulan derin öğrenme modellerinin eğitim ve test sonuçları tartışılmıştır. Yedinci bölümde, önerilen Ortalama Doküman Vektörleri açıklanmış, eğitim ve test sonuçları, başarı performansları literatürdeki çalışmalar ile karşılaştırmalı olarak tartışılmış ve sunulmuştur. Yapılan tüm çalışmaların katkıları ilgili bölümlerde ve sonuç bölümünde ortaya koyulmuştur.

2. BÜYÜK VERİ VE DAĞITIK MAKİNE ÖĞRENMESİ

2.1 Özet

Geleneksel veri depolama ve analiz sistemleri ile depolanamayacak veya işlenemeyecek büyüklük veya çeşitlilikteki verilere büyük veri adı verilmektedir. Bu kavram aynı zamanda büyük veri analizi için kullanılan algoritma ve teknolojileri de ifade etmektedir.

Büyük veriyi işlemek için pek çok dağıtık çözümler sunulmuştur. Bu çözümlerden en yaygını MapReduce platformudur. Büyük veri işlemek için popüler bir paradigma olan MapReduce, Hadoop gibi pek çok açık kaynak makine öğrenmesi platformlarında uygulanmaktadır. Hadoop MapReduce platformunun avantajları esnek ve ölçeklenebilir olmasıdır. Veri depolama sisteminden ya da programlama dilinden bağımsiz, yüksek derecede ölçeklenebilir, basit ve hata toleransı (fault tolerance) yüksek olan bir uygulama çatısı sunar. Fakat, makine öğrenmesi ve online işleme için karmaşık yinelemeli algoritmaları desteklemez. Karmaşık ve yinelemeli algoritmaları desteklemek için Mahout, Spark, Scikit-learn ve GraphLab gibi makine öğrenmesi platformları geliştirilmiştir.

Veri boyutu katlanarak büyüdükçe, büyük veriyi etkili bir şekilde analiz eden algoritmaları seçmek daha önemli bir hale gelmektedir. Bu noktada, çoklu bilgisayarlar üzerinde dağıtık analiz algoritmalarını uygulamak önemli bir performans maliyeti sağlar.

Günümüzde dağıtık makine öğrenmesi popüler bir alan haline gelmiştir [27]. Tek bir makine üzerinde çalışan geleneksel makine öğrenmesi genellikle küçük ve kontrollü veri setleri kullanmaktadır. Döküman sınıflandırma, ağ trafiği, biyoinformatik, bilgisayarlı görme ve diğer büyük veri alanlarında toplanan gerçek veriler ise daha geniş ve gürültülüdür. Öğrenme algoritmalarının eğitim veri boyutu arttıkça sınıflandırma ve kümeleme algoritmalarından elde edilen doğruluk oranı büyük ölçüde artar. Fakat tek bir makine küçük veri kümelerini işleyebilir ve yüksek hesaplama gücü, depolama kapasitesi ve ağ trafiği gerektiren daha geniş veri kümeleri için yüksek performans sunamaz. Bu problemi çözmek için, ölçeklenebilir makine öğrenmesi başarılı sayısız teknikler ile önemli bir alan olmuştur ve dağıtık makine öğrenmesi gibi ölçeklenebilir makine öğrenmesine talep son yıllarda artmıştır.

Tezin bu bölümünde, öncelikle büyük veri kavramı tartışılmış, ilgili teknolojiler ve yaklaşımlar detaylı olarak incelenmiştir. Sonraki kısımda makine öğrenmesi kavramı ve

büyük veri analizi için kullanılan dağıtık makine öğrenmesi algoritmaları kütüphanelerini içeren Spark, Mahout, Scikit-learn ve GraphLab gibi platformlar tartışılmış ve karşılaştırılmıştır. Literatürde var olan çalışmalar detaylı bir şekilde incelenerek seçilen dağıtık makine ögrenmesi platformalarının avantajları ve dezavantajları açıkça ortaya konulmuş ve yapılan araştırmaların büyük verileri işleme çalışmalarına genel bir bakış açısı sunması hedeflenmiştir. Son kısımda ise tez kapsamında gerçekleştirilen iki ana büyük veri çalışması verilmiştir. Bunlar Dağıtık Okunabilirlik Analizi Sistemi ve Çağrı Merkezleri İçin Dağıtık Performans Değerlendirme Sistemidir.

2.2. Giriş

World Wide Web'in (WWW veya web) gelişimi ile birlikte veri boyutları benzeri görülmemiş bir şekilde artmaktadır. Web tıklama geçmişi, mobil cihazlar üzerinden online etkileşimler gibi internet üzerinden ve Twitter, Facebook gibi sosyal ağlardaki güncellemeler veya Youtube videoları gibi sosyal medya sitelerinden üretilen veri miktarı IBM'e göre günlük yaklaşık olarak 2.5 kentilyondur. Örneğin, Twitter Aralık 2017 itibariyle aylık 330 milyon aktif kullanıcıya ve günlük ortalama 400 milyon tweet sayısına ulaşmıştır. Veri hacmindeki büyümenin katlanarak artması, veri analizini zorlaştırmakta ve araştırmacıları karmaşık analiz tekniklerine yönlendirmektedir.

Geleneksel veri tabanı sistemleri ve yazılım teknikleri kullanarak işlenmesi zor olan böylesi geniş veri kümeleri "büyük veri" olarak adlandırılmaktadır. Büyük veri kavramı aynı zamanda bu verilerin depolanması, sorgulanması ve analizi için geliştirilen teknolojileri de ifade etmektedir.

Endüstri analisti Daug Laney [28], bugün de geçerliliğini hala koruyan büyük verinin tanımını 3V (Volume, Velocity, Variety) ile açıklamaktadır. Volume (katlanarak büyüyen veri hacmi), velocity (veri oluşum oranındaki hız) ve variety (veri kaynaklarının çeşitliliği) büyük verinin karşılaştığı en temel sorunlardır.

Yüzlerce, hatta binlerce bilgisayara dağıtılan ve çoğunluğu yapısal olmayan (sosyal medya verileri, email, sensör bilgileri vb) çok büyük miktardaki veriler, dağıtık analiz gerektirdiği için geleneksel makine öğrenmesi yöntemleri büyük verinin analizinde yetersiz kalmaktadır. Bu nedenle, reklamcılık, pazarlama, e-ticaret, müşteri ilişkileri yönetimi, güvenlik gibi pek çok endüstriyel ve akademik uygulama alanlarına sahip ciddi strateji ve

çözümler gerektiren büyük verinin işlenmesi için Hadoop/MapReduce, Mahout, Spark, GraphLab, Scikit-learn, Storm gibi açık kaynaklı proje gerçekleştirilmiştir. Çeşitli sunucular üzerinde dağıtık bulunan verilerin bulunduğu disklere tek bir dosya sistemi üzerindeymiş gibi erişebilme imkanı sunularak çeşitli makine ögrenmesi tekniklerini geliştiren açık kaynaklı bu projeler zaman ve maliyet açısından avantaj sağlayarak büyük verinin karşılaştığı sorunlara çözüm önermiştir.

2.3 Büyük Veri

Büyük veri çeşitlilik, hacim, hız, doğruluk ve değer gibi 5V (variety, volume, velocity, veracity, value) olarak adlandırılan çeşitli sorunlar nedeniyle pek çok zorluğu beraberinde getirmektedir. Büyük verinin 5V ile tanımı Tablo 2.1'de açıklanmıştır.

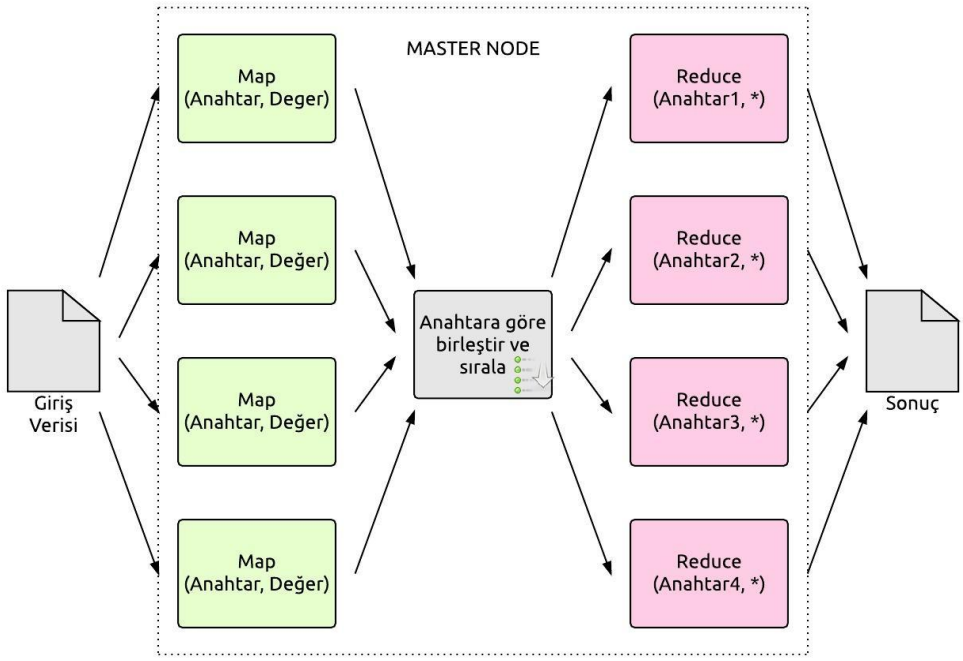
Tablo 2. 1. 5V ile büyük veri

Büyük Veri Özellikleri	**Tanımı**
Variety(Çeşitlilik)	Sosyal medya yorumları, videolar, fotoğraflar ve mesajlar gibi farklı şekillerde yapılandırılmamış veriler
Volume (Hacim)	Özellikle internet teknolojilerinin gelişimi ile katlanarak büyüyen veriler
Velocity (Hız)	Verinin nasıl hızla üretildiği ve nasıl hızla analiz edilmesi gerekliliği
Veracity (Doğruluk)	Önemli kararlar için veri güvenirliğinden emin olma
Value (Değer)	Veri analitiği için kullanılmak üzere farklı kurumlar tarafından depolanan verinin değeri

Mevcut veri tabanı yönetim sistemlerinin ve yazılım gereçlerinin işleyemediği büyük ve karmaşık veri kümeleri olarak adlandırılan büyük veri, günümüzde yapısal olmayan verileri barındıran veri tipi problemlerine çözüm aramaktadır. İnternet ortamında sayısı katlanarak artan verinin işlenme hızının da aynı oranda artması gerekmektedir. Veri transferi sırasında güvenlik seviyelerinin maksimun düzeyde tutulup, veri gizliliğinden emin olunması beklenmektedir. En önemli özellik ise verinin kullanıldığı kurum için bir artı değer yaratıyor olması gerektiğidir.

2.3.1. MapReduce Paradigması

Büyük veriyi işlemek için 5V ile açıklanan temel sorunlar, kurumlar tarafından toplanan tüm verilerden en önemli olanları filtreleme ve böylesi geniş miktarda verileri etkili bir şekilde analiz eden algoritmaları tasarlamaktır. Büyük veriyi işlemek için pek çok dağıtık çözümler sunulmuştur. Bu çözümlerden en yaygını, Google [29] tarafından geliştirilen MapReduce yaklaşımıdır. MapReduce yaklaşımı Şekil 2.1'de gösterildiği gibi çalışmaktadır.



Şekil 2. 1. MapReduce yaklaşımına genel bir bakış

Bir master node tarafından kontrol edilen MapReduce görevleri, Map ve Reduce olarak ikiye ayrılır. Map fonksiyonu giriş verisini anahtar-değer gruplarına böler ve her map görevinin çıktısı onların anahtar değerlerine göre sınıflandırılır. Reduce fonksiyonu ise değerleri birleştirerek sonuç verileri elde eder.

Klasik bir uygulama olarak MapReduce'ün WordCount (kelime sayma) örneğini düşünelim. Bu uygulama, kendisine verilen bir metin dosyası içerisinde hangi kelimenin ne sıklıkla geçtiğini bulmaktadır. Verilen metin dosyaları içerisinde kelimelerin sayılması, basit bir yazılım uygulaması ile çözümlenebilmektedir. Fakat kelimeleri saymamız istenen metin

dosyasının terabayt ya da petabayt boyutlarında olduğu durumda basit bir yazılım ile uygulamanın gerçekleştirilmesi mümkün değildir. Buna karşılık Hadoop kümesi üzerinde çalışan bir MapReduce uygulaması ile bu sorun basitçe çözülmektedir.

MapReduce paradigmasının en temel bileşeni görev (job) ismini almaktadır. Görev, programı map ve reduce fonksiyonlarına ayırarak işlemi iki aşamada tamamlamayı hedeflemektedir. WordCount örneğine geri dönecek olursak, map aşamasında her bir dosyadaki kelimelerin sayılma işlemi gerçekleşmektedir. Reduce aşaması ise, map aşaması sonucunda çıkan değerleri birleştirerek bir kelimenin verilen tüm metin dosyalarında ya da dosya kümelerinde ne sıklıkla geçtiğini sayarak sonuç üretmektedir. WordCount uygulamasının sözde kodu aşağıda verilmiştir.

```
Map(String anahtar, String değer)
//anahtar: metin dosyasının ismi
//değer: dosya içerikleri
for each kelime w in değer:
   EmitIntermediate (w, "1");

Reduce(String anahtar, String değer)
//key: bir kelime
//değerler: sayıların listesi
int sonuc -0;
for each v in degerler;
   sonuc += ParseInt(v);
Emit(AsString (sonuc));
```

Farklı makinelere dağıtılan map görevleri, diğer map görevleri ile iletişim halinde değildir; aynı durum reduce görevleri için de geçerlidir. Verinin kopyaları ayrı makinelerde tutulduğu için, bir makinenin donanımsal ya da yazılımsal herhangi bir nedenle çökmesi durumunda görevler devam etmektedir. Ayrıca map işlemleri diğer map işlemlerini beklemek zorunda değildir, bir map görevinin arkasından reduce görevi başlayabilir yani sıralı bir şekilde çalışmak zorunda değildir. Böylelikle MapReduce veri depolama sisteminden ya da programlama dilinden bağımsız, yüksek derecede ölçeklenebilir, basit ve hata toleransı (fault tolerance) yüksek olan bir uygulama çatısı sunar.

MapReduce, büyük veri işlemek için popüler bir platformdur ve ölçeklenebilir, güvenilir bir depolama için dağıtık dosya sistemini (distributed file system-DFS) kullanmaktadır. Bir MapReduce görevi dağıtık dosya sisteminden giriş verisini okur ve çıkış verisini dağıtık

dosya sistemine yazar. Dağıtık dosya sisteminde büyük bir dosya, küme (cluster) içerisinde dağıtık bulunan çoklu bloklara bölünür. Her dosya bloğu hata toleransı (sistemin düşmesi durumu) için farklı düğümlerde saklanan verilerin kopyalarını tutmaktadır.

Büyük ölçekli veri analizi için MapReduce tabanlı sistemlerin gittikçe yaygınlaşmasının pek çok sebebi vardır. Bu sebeplerden bir kaçı aşağıda sıralanmıştır:

- *MapReduce arayüzü basit fakat etkileyicidir.* MapReduce sadece map ve reduce fonksiyonlarını kullansa da, SQL sorgulama, veri madenciliği, makine öğrenmesi, graf işlemeyi içeren sayısız veri analitik görevleri MapReduce ile yerine getirilebilir.
- *MapReduce esnektir (flexible).* MapReduce depolama sistemlerinden bağımsız tasarlanmıştır. Analiz edilecek tüm veriyi bir yerden başka bir yere taşımak yerine verilerin olduğu sistem üzerinde küçük analizler gerçekleştirilmektedir ve yapılandırılmış ya da yapılandırılmamış farklı çeşitlerdeki tüm veriler analiz edilebilmektedir.
- *MapReduce ölçeklenebilirdir (scalable).* Paylaşılan bir kümede binlerce düğüm üzerinde MapReduce çalışabilirken, sistemde bir hata oluştuğu zaman sadece hatanın olduğu düğümdeki görevleri tekrar çalıştırarak hata toleransını sağlayabilmektedir.

MapReduce paradigması, farklı lokasyonlarda bulunan büyük ölçekli veri kümeleri üzerinde veri işleme problemlerini çözmek için ortaya çıkmıştır. Büyük veri işlemek için popüler bir paradigma olan MapReduce, Hadoop [30] gibi pek çok açık kaynak projelerinde uygulanmıştır.

Hadoop, iki temel bileşenden oluşmaktadır: Yarn ve HDFS. Yarn (Yet Another Resource Negotiator), bir Hadoop kümesi üzerinde çalışan uygulamaları saklamakta ve CPU, hafıza yönetimi sağlamaktadır. Hadoop'un ilk jenerasyonu yalnızca MapReduce uygulamalarını çalıştırabilirken Yarn, Hadoop'un yanısıra Spark gibi diğer uygulama platformlarının çalışmasına da olanak sağlamaktadır.

HDFS (Hadoop Distributed File System) [31], veri depolama için bir Hadoop kümesinde bütün düğümlere verileri dağıtan bir dosya sistemidir. Yüksek hız ile büyük miktardaki veriye erişim sağlayabilen bu dağıtık dosya sistemi olarak, bir çok makinedeki dosya

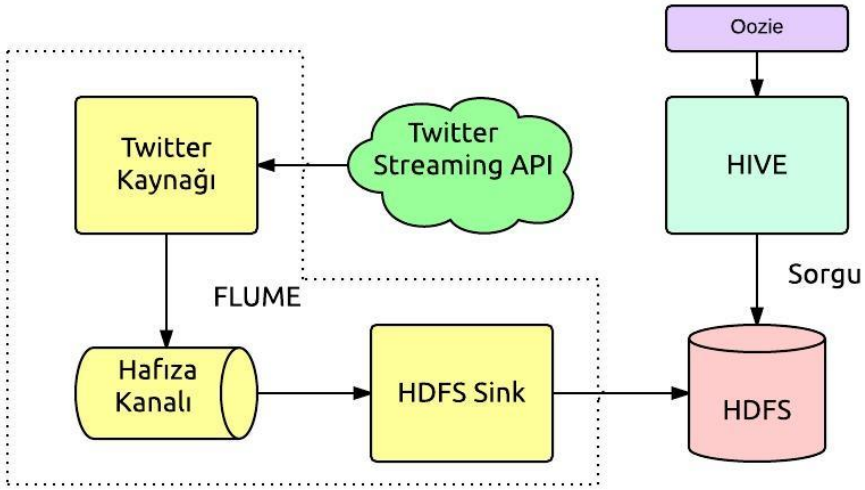
sistemlerini birbirine bağlayarak tek bir dosya sistemi gibi kullanılabilmelerine olanak sağlamaktadır.

Hadoop'un Google'un geliştirdiği MapReduce'dan en büyük farkı açık kaynaklı olmasıdır. Böylece kullanıcıya düşük bir maliyet ile Google kadar hızlı çalışabilen bir arama motoru gerçekleştirebilme olanağı tanımaktadır. Ayrıca kullanıcıya, Facebook, Twitter ve diğer sosyal medya konuşmalarından duygu, davranış ve içerik analizlerini gerçekleştirebilme imkanı sunmaktadır.

Örneğin Twitter mekanizmasının nasıl çalıştığını düşünelim. Ayşe adlı bir kullanıcı pek çok kişiyi takip etmektedir ve aynı şekilde Ayşe de pek çok kişi tarafından takip edilmektedir. Ayşe bir tweet attığında, yani bir düşüncesini yazdığında tüm takipçileri tarafından bu tweet görülebilmektedir. Aynı zamanda Ayşe başka kullanıcıların tweet'lerini de retweet yapabilmektedir. Retweet, bir e-postayı iletmek gibidir. Ayşe, Fatmadan bir tweet görürse, onu retweet eder ve Ayşe'nin takipçileri Fatma'nın tweet'ini görür. Twitter, tüm tweet'ler için retweet sayısını tuttuğu için, kullancı tweet'lerini analiz ederek istediğimiz kullanıcıya ulaşabiliriz. Ayrıca en çok retweet alan kullanıcı kim olduğunu bularak istenilen alanda en popüler kişiyi tespit edebiliriz. SQL sorguları bu soruların cevabı için kullanılabilir. Retweet'ler azalan şekilde sıralandığında, en çok retweet'i alan kişiye bakabiliriz. Fakat Twitter Streaming API tweet'leri karmaşık gelebilen bir JSON (JavaScript Object Notation) formatında verdiği için geleneksel ilişkisel veri tabanı yönetim sistemleri bu işlem için yetersiz kalmaktadır. Hadoop ekosisteminde, Hive [32] projesi Hadoop dağıtık dosya sisteminde bulunan verileri sorgulamak için kullanılabilen bir sorgu arayüzü sağlamaktadır ve HiveQL [32] adı verilen SQL'e benzer bir dil kullanarak veriyi sorgulamaktır. Şekil 2.2'de twitter verilerinin Hadoop ile nasıl analiz edilebileceğini gösteren bir diyagram sunulmuştur.

Apache Flume [33], Hadoop'un Cloudera açık kaynak dağıtımını (CDH) kullanarak HDFS (Hadoop dağıtık dosya sistemi) içerisine veriyi getirmenin bir yoludur. Flume büyük veriyi etkili bir şekilde toplayan ve götüren dağıtık, güvenilir ve erişilebilir bir servistir. Twitter örneğinde, Twitter Streaming API'den veri toplamak ve Hadoop dağıtık dosya sistemine göndermek için Flume kullanılmıştır. Flume bu işlemi gerçekleştirmek için 3 temel yapıdan oluşur (kaynak, kanal ve sink). HDFS Sink, bir kanal üzerinden gelen tweet'leri dağıtık dosya sistemine yazmak için kullanılır. Hadoop dağıtık dosya sistemine veriler yüklendikten sonra, ilk adım Hive üzerinde harici bir tablo yaratarak verileri

sorgulamaktır. Harici tablo kullanmak, bir veri kümesinin dağıtık dosya sisteminde tamamlandığı yerden veriyi başka bir dosya sistemine taşımaksızın tabloyu sorgulama imkanı sağlar. Ölçeklenebilirliği sağlamak için, daha fazla veri yükledikçe tabloyu bölmemiz gerekmektedir. Bölümlenmiş bir tablo, büyük veri kümelerini analiz ederken daha iyi sonuç elde etmek için sorgulama sırasında okunmuş dosyaları ayrı tutmamıza olanak sağlar. Fakat Twitter API, tweet akışına devam eder ve Flume sürekli yeni dosyalar oluşturur. Böylece yeni veri geldikçe tabloya bölümleri ekleyerek bu periyodik işlem otomatik bir şekilde devam etmektedir. Apache Oozie [34] bu sorunu çözmek için kullanılabilen bir iş akışı (workflow) koordinasyon sistemidir. Oozie iki temel bölümden oluşmaktadır: MapReduce, Pig [35], Hive ve benzeri farklı Hadoop işlerinden oluşan iş akışını çalıştıran ve saklayan bir iş akışı motoru (workflow engine) ve önceden tanımlanmış programlar üzerinde iş akışı işlerini çalıştıran bir koordinatör motoru (coordinator engine).



Şekil 2. 2. Apache Hadoop ile Twitter verilerinin analiz edilmesi

Büyük veriyi depolayabilen dağıtık dosya sisteminden ve kullanıcıdan yalnızca map-reduce görevlerini yazması istenen basit bir programlama modeli olan MapReduce platformundan oluşan Hadoop'un temel avantajları ölçeklenebilirlik, maliyet etkinliği, esneklik ve hata toleransı olarak sıralanabilir. Bu avantajların kısaca açıklaması şöyledir:

- Verinin hangi formatta olduğundan ve hangi uygulama altında çalıştığından bağımsız bir şekilde yeni veri düğümleri eklenebilmektedir (ölçeklenebilirlik).

- Hadoop, sunucular arasında etkili bir paralel hesaplama gücü sağlayarak daha fazla verinin depolanmasına ve maliyet açısından daha büyük kazanç elde edilmesine olanak tanımaktadır (maliyet etkinliği).
- Farklı dağıtık kaynaklardan gelen yapısal veya yapısal olmayan tüm veri tipleri Hadoop ile işlenebilmekte ve analiz edilebilmektedir (esneklik).
- Donanımsal ya da yazılımsal bir hata sebebiyle bir veri düğümü düştüğü zaman sistem veri işlemeye devam etmektedir (hata toleransı).

Büyük veriyi hızlı bir şekilde işleyebilen, yapısındaki HDFS sayesinde veri okuma performansını arttıran, sistem hatalarının üstesinden gelebilen ve birbirinden bağımsız makineler üzerinden görevler gerçekleştirebilen avantajlarının yanında Hadoop MapReduce platformunun dezavantajları aşağıda verilmişir:

- Bir mapper fonksiyonundan sonra reducer fonksiyonu çalıştırmak zorunda kalınan bir model ile sınırlıdır. Dolayısıyla ihtiyaç duyulduğunda çok fazla mapper ve reducer ile büyük veriyi işlemek zorunda kalmaktadır.
- Sunucu kümeleri üzerinde hata ayıklama (debug), yazılım dağıtımı ve loglama gibi işlemleri gerçekleştirmek ve yönetmek zordur.
- Sık kullanılan karmaşık iteratif (yinelemeli) algoritmaları desteklemez. Iteratif algoritmalarda, bir değerin hesaplanması önceki hesaplanan değerlere bağlı olarak değişir ve map-reduce görevleri birbirinden bağımsız bir şekilde çalıştığı için MapReduce kullanılamaz.
- Bir MapReduce platformunda makine öğrenmesi algoritmaları uygulandığında, her iterasyon sonucu diske yazılır. Diskten okuma ve yazma gibi iteratif işlemler ise sistemi yavaşlatmaktadır.

İterative ve karmaşık algoritmaları desteklemek için Haloop [36], Twister [37] ve Mahout [38] gibi birçok alternatif MapReduce tabanlı platformlar tasarlanmıştır. Haloop ve Twister, MapReduce paradigmasını kullanarak etkili bir şekilde iteratif hesaplamalar geliştirmek ve iteratif MapReduce işlerini optimize etmek için tasarlanmıştır.

2.4. Makine Öğrenmesi

Büyük miktarda verinin manuel olarak işlenmesi ve analizinin gerçekleştirilmesi, geleneksel veri tabanı sistemleri ile mümkün değildir. Şimdiki çalışmalar, geçmiş gözlemlere dayanarak doğru tahminleri yapmak için otomatik öğrenmenin nasıl gerçekleşeceğine odaklanmaktadır. Makine öğrenmesi, geniş veri kümelerinin işlenmesi ve analizinin gerçekleşmesi için gerekli algoritmaları, yöntemleri, geliştirme süreçlerini inceleyen bir bilgisayar bilimidir. Büyük miktardaki verilerin, makineler tarafından çözümlenmesi için önerilen yaklaşımlar veriye uygun bir model bulmaya çalışır ve bu modele göre verilerin analizini gerçekleştirilir. Günümüzde makine öğrenmesi bilgisayarlı görme, NLP, konuşma tanıma, tıbbi tanı, borsa çözümlemesi, arama motorları, spam yakalama gibi çok geniş bir uygulama alanına sahip olduğu için çok sayıda çalışmanın ilgi noktası durumundadır.

Makine öğrenmesinin temel amacı geçmiş deneyimleri kullanarak gelecek kararları vermektir [39]. Bilgisayar bilimcisi Tom Mitchell [40], makine öğrenmesini şu şekilde tanımlamaktadır:

"Eğer bir bilgisayar programı, bir T görevini, P performansında yaparak, deneyiminde E artış sağlıyorsa, o bilgisayar programı, T görevinde, P performansıyla, E deneyimlerinden öğreniyordur denilir".

Makine öğrenmesinin basit bir örneği döküman filtrelemedir. Spor ya da ekonomi konularını içeren binlerce metin belgeleri gozlemlenerek, döküman filtreleme yeni metin belgelerini sınıflandırmayı öğrenir. Bir T görevinde, dökümanlar ekonomi ve spor konularına göre ayrılarak sınıflandırılır. Bir P performansında bilgisayar programı, sınıflandırılmış dökümanları bu görevde çalıştırır ve E deneyimlerinde sınıflandırılmış dökümanların doğruluk yüzdelerini hesaplayarak performansını değerlendirir.

Makine öğrenmesi gözetimli (supervised) ve gözetimsiz (unsupervised) öğrenme olarak kabaca ikiye ayrılabilir. Gözetimli öğrenmede, bir program işaretlenmiş bir girdi için eğitim verilerini kullanarak çıktıyı tahmin eder. Gözetimli öğrenmede önceden belirlenmiş bir eğitim verisi üzerinde çeşitli öğrenme algoritmaları uygulanarak sınıf eşleştirmesi yapılır ve daha sonradan değişik veri setleri üzerinde bu eşleşmenin başarı oranı hesaplanır. En yaygın gözetimli öğrenme yaklaşımları sınıflandırma ve regresyondur. Gözetimsiz öğrenmede, bir program işaretlenmiş verilerden öğrenme ve eğitim verilerini kullanma yerine verideki bazı

örüntüleri bulmaya çalışır. Uygulanan öğrenme algoritması bütün verileri girdi olarak alır ve veri üzerinde örüntü saptamaya çalışır. En yaygın gözetimsiz öğrenme yaklaşımı, kümeleme olarak adlandırılan veri setlerindeki benzer grupları bulmaktır. Sınıflandırmanın aksine kümeleme, eğitim verilerinden öğrenmeden ziyade gözleme dayalı bir öğrenme tekniğidir. Yarı-gözetimli öğrenme ise işaretli ve işaretsiz verileri kullanılarak verilerin birleşimi olan her iki gözetimli ve gözetimsiz öğrenmeyi kullanmaktadır.

2.5. Dağıtık Makine Öğrenmesi

Günümüze dağıtık makine öğrenmesi popüler bir alan haline gelmiştir [27]. Tek bir makine üzerinde çalışan geleneksel makine öğrenmesi genellikle küçük ve kontrollü veri setleri kullanmaktadır. Döküman sınıflandırma, ağ trafiği, biyoinformatik, bilgisayarlı görme ve diğer büyük veri alanlarında toplanan gerçek veriler ise daha geniş ve gürültülüdür. Öğrenme algoritmalarının eğitim veri boyutu arttıkça sınıflandırma ve kümeleme algoritmalarından elde edilen doğruluk oranı büyük ölçüde artar. Fakat, tek bir makine küçük veri kümeleri ile çalışır ve yüksek hesaplama gücü, depolama kapasitesi ve ağ trafiği gerektiren daha geniş veri kümeleri için yüksek performans sunamaz. Bu problemi çözmek için, büyük-ölçekli makine öğrenmesi (large-scale machine learning) son yıllarda başarılı sayısız teknikler ile önemli bir alan olmuştur [41–47].

Büyük verilerin işlenmesi ve analizinde var olan makine öğrenmesi algoritmalarının uygulanması oldukça zordur ve bu algoritmaların tek bir makine üzerinde uygulaması günlerce hatta haftalarca sürmesi zaman ve maliyet açısından dezavantaj oluşturmaktadır. Ek olarak [48]'de görüleceği gibi tek bir merkezi işlem birimi üzerinde dağıtık bilgi işleme aşağıdaki nedenlerden dolayı etkisiz bir yöntemdir ve internet üzerinde arama motorları, e-mail, sosyal ağlar ve diğer servislerin yaygınlaşması ile ilgi odağı haline gelen veri koruma (data privacy) hususunda veri güvenliğini risk altına sokmaktadır.

- Tek bir merkezi veri setinin depolama maliyeti, veri setinin daha küçük parçalara bölünerek elde edilen küçük veri setlerinin depolama maliyetinin toplamından daha büyüktür. Merkezi bir depolama sisteminin çok yüksek bir kapasite alanı gerektirdiği tartışılmazdır. Yeryüzü ve uzay teleskobundan astronomi bilim ve özellikle görüntü verilerinin elde edildiğini düşünelim. Böylesi bir veritabanı

boyutu yüksek hızda artarak eksabaytlara (10^{18} bayt) ulaşmaktadır. Gezegenin tüm teleskop verilerinin tutulması, muazzam yükseklikte bir maliyet gerektiren çok büyük veri ambarı gerektirmektedir.

- Merkezi bir veritabanının işlenmesi için gerekli hesaplama maliyeti, verilerin daha küçük parçalar halinde analiz edilerek hesaplanması maliyetlerinin toplamından daha büyüktür. Ayrıca öğrenme görevi, paralel bir şekilde çalışabilen dağıtık birim olarak parçalar halinde pek çok alt görevlere bölünebilir. Dağıtık bir veri analizi yaklaşımı, mevcut kaynaklardan daha iyi yararlanılmasını sağlamaktadır. Merkezi bir yaklaşım çok uzun bir zaman aldığından dolayı, dağıtık bir şekilde veriyi analiz etmek başarılı bir iş stratejisi geliştirmek için en iyi yoldur.
- Ağ üzerinden çok büyük boyutlara sahip verilerin transferi çok fazla zaman alır ve finansal maliyet gerektirir. Küçük boyutlu veriler bile sınırlı bant genişliğine sahip kablosuz ağ ortamlarında problemler oluşturabilir. Sık sık güncellenen veritabanları düşünüldüğünde bu durum, uygulamanın uzun zaman alması ile makinenin kullanışlığına engel olacaktır.
- Tıbbi ve finansal kayıtlar gibi özel verilerin korunması son derece önemlidir. Böylesi verilerin tutulduğu merkezi bir veritabanına yapılacak olası bir saldırı sistemi ve veri güvenliğini riske sokacaktır.

Tek bir merkezi sistemde, farklı lokasyonlardan toplanan dağıtık büyük verinin işlenmesi çok büyük hesaplama gücü ve depolama maliyeti gerektirmektedir. Diğer bir açıdan, veri setleri daha küçük veri setlerine bölünüp çeşitli sunuculara dağıtıldığında, küçük veri setlerinin analiz maliyetlerinin toplamı merkezi bir veri setinin depolama ve hesaplama maliyetinden daha az olacaktır. Büyük dağıtık bilginin işlenmesi ölçeklenebilir makine öğrenmesi sorunlarını da birlikte getirir. Bu sorun, dağıtık makine öğrenmesi gibi ölçeklenebilir makine öğrenmesi çözümlerine talebi arttırmaktadır. Dağıtık makine öğrenmesi, dağıtık veriden etkili bir şekilde yararlı bilgi çıkarımını hedeflemektedir.

Veri boyutu katlanarak büyüdükçe, büyük veriyi etkili bir şekilde analiz eden algoritmaları seçmek daha önemli bir hale gelmektedir. Bu noktada, çoklu bilgisayarlar üzerinde dağıtık analiz algoritmalarını uygulamak önemli bir performans maliyeti sağlar [49]. Dağıtık veri ortamındaki heterojenlik, çapraz platform, tutarlılık, esneklik, karmaşıklık,

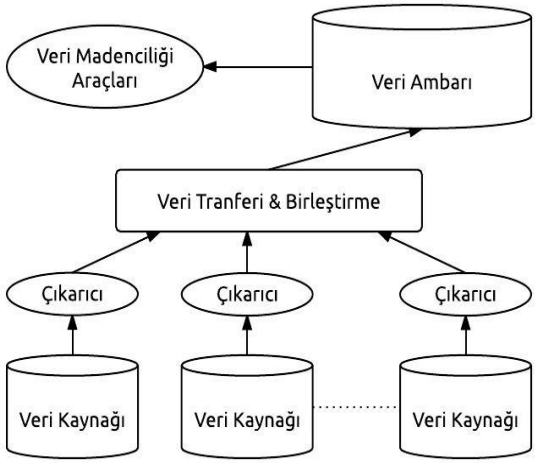
gizlilik sorunu ve diğer kısıtlamalar potansiyel pek çok soruna neden olmaktadır. Bu sorunları çözmek için, dağıtık veri madenciliği teknikleri geliştirilmiştir. [50]'de dağıtık veri madenciliğinin temel sorunları tartışılmış ve çözüm olarak hiyerarşik bir dağıtık veri madenciliği yapısı önerilmiştir. Büyük veri ile veri madenciliği sorunları [51]'de veri, model ve sistem seviyesinde detaylı bir şekilde analiz edilmiştir:

- Veri seviyesinde, farklı lokasyonlarda depolanan büyük veri sıklıkla heterojen, belirsiz ve eksik veri içerir. Bu yüzden, güvenli veri koruma ve bilgi paylaşım protokolleri geliştirmek büyük bir sorundur.
- Model seviyesinde, farklı veri kaynaklarından global modeller üretmek anahtar sorundur. Bu durum, dağıtık veri kaynakları arasında model ilişkileri için tasarlanan algoritmaların dikkatlice analiz edilmesini gerektirmektedir.
- Sistem seviyesinde, büyük veri madenciliği dayanıklılık (robustness) ve ölçeklenebilirliği sağlamak amacıyla veri kaynakları ve modeller arasında bazı önemli ilişkileri dikkate almasını gerektirir.

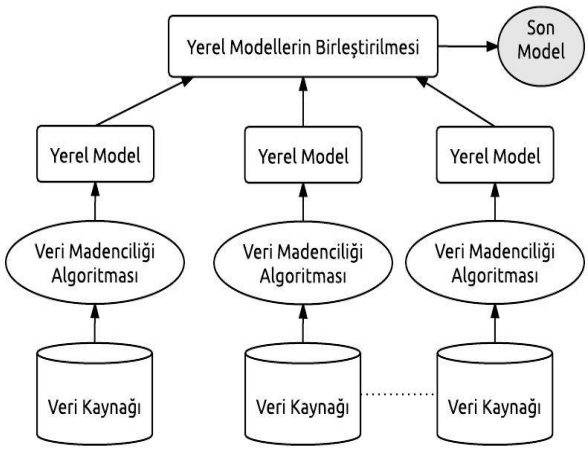
Büyük veri ile ilgili diğer sorunlar [45]'da sunulmuştur. Nong Ye [52], Şekil 2.3'de gösterilen dağıtık veri madenciliği platformunun geleneksel veri madenciliği mimarisine göre aşağıdaki avantajlarına dikkat çekmiştir:

- *Veri ambarı tabanlı bir mimaride madencilik*, her bir veri kaynağından çıkarılan veriler bir sonraki merkezi veri madenciliği uygulaması için veri ambarına transfer edilmektedir. Bu merkezi yaklaşım, dağıtık kaynakların kullanım eksikliği, gizlilik sorunlari ve uzun tepki süresi nedeniyle veri madenciliği algoritmalarını çoğu dağıtık veri madenciliği uygulaması için kullanışsız hale getirir.
- *Dağıtık bir veri madenciliği platformunda,* seçilen veri madenciliği uygulaması her bir veri kaynağına uygulanır ve elde edilen yerel modeller toplanarak son model elde edilir. Dağıtık veri madenciliğinin amacı, hesaplama, depolama ve iletişim yetenekleri gibi özelliklere göre dağıtık veri kaynakları üzerinde veri madenciliği algoritmalarını gerçekleştirmektir. Özellikle uygulama daha geniş sayıda veri kaynağı gerektirdiğinde bu yaklaşım zaman ile değişen veriyi analiz etmek için daha ölçeklenebilir ve pratiktir.

Guo ve Sutiwaraphum [53], öğrenme doğruluğu ve çalışma zamanlarını baz alarak iki öğrenme yaklaşımı ile dağıtık veri madenciliğinin avantajlarını tartışmıştır. İlk olarak, dağıtık veri setleri için farklı dağıtık makine öğrenmesi algoritmalarını kullanmak, özellikle geniş boyutlu bir alanda yüksek doğruluk oranına erişme olasılığını arttırır. İkincisi, bağımsız bir model çıkarmak amacıyla her işlem verinin farklı bir parçasında çalışır. Böylece dağıtık makine öğrenmesi, büyük ölçekli makine öğrenmesi sorunlarına bir çözüm olarak çalışma zamanı ve hafıza sınırlamaları gibi kısıtlamaların üstesinden gelmektedir.



(a)



(b)

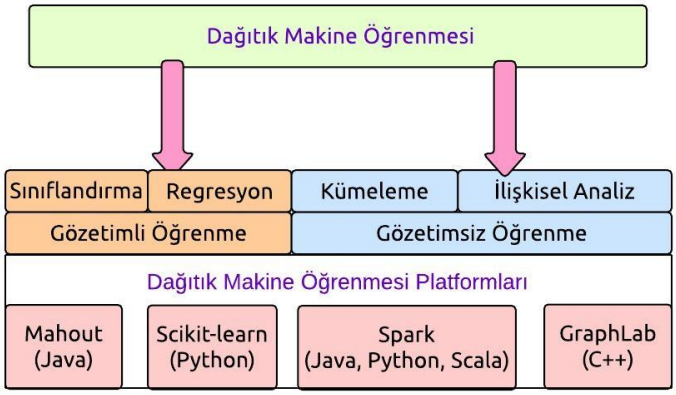
Şekil 2. 3. Veri ambarı tabanlı bir mimaride madencilik (a), dağıtık veri madenciliği platformu (b) [17]

Geleneksel makine öğrenmesi algoritmaları merkezi bir veritabanında depolanan bir veri setini kümeler ya da sınıflandırır. Fakat sınırlı hafıza nedeniyle bütün dağıtık verinin merkezi bir veritabanına transfer edilmesi mümkün değildir [54]. Çoğu dağıtık makine öğrenmesi aşağıdaki temel prensip üzerine çalışmaktadır:

- Her bir veri kaynağına belirlenen merkezi bir algoritma uygulanır.
- Her alt veri kaynağı merkezi algoritma sonuçlarını global veri kaynağına gönderir.
- Bütün yerel modeller toplanarak global bir model elde edilir.
- Ana veri kaynağı global modeli her bir alt veri kaynağına gönderir ve her bir veri kaynağı global modele göre algoritma sonuçlarını günceller.

Zhang ve Bao [55], dağıtık veri kaynaklarında konumlanan dağıtık veriyi işlemek için yeni bir dağıtık veri madenciliği platformu geliştirmiş ve makine öğrenmesi algoritmaları, veri ambarı, yüksek performans ve hesaplama gücü içeren veri madenciliğini etkili bir şekilde uygulamıştır. Önerilen dağıtık veri madenciliği mimarisi, dağıtık veri kaynaklarındaki bütün yerel verileri analiz eden ve yöneten veri ambarından global bir bilgi çıkarımını hedeflemektedir ve yapılandırılmamış verileri dönüştürmek için veri dönüşüm hizmetinin yanı sıra veri madenciliği kaynakları arasında iş gönderme hizmeti ve ağ yönetimi sağlamaktadır.

2.6. Dağıtık Makine Öğrenmesi Platformları



Şekil 2. 4. Dağıtık makine öğrenmesi platformlarına genel bir bakış

Dağıtık makine öğrenmesi platformlarının geliştirilmesinin bilinen ilk nedeni, verinin tek bir diskte saklanamayacak kadar büyük olmasıdır. Şekil 2.4'de dağıtık makine öğrenmesi kavramına genel bir bakış açısı sunulmuştur.

2.6.1. Apache Mahout

Mahout, Apache Yazılım Lisansı altında lisanslanmış dağıtık bir makine öğrenmesi platformudur. Mahout [38], sınıflandırmadan ortak filtreleme (collaborative filtering) ve kümelemeye kadar pek çok algoritmayı çok sayıda makine üzerinde paralel bir şekilde çalıştırabilmektedir. Bu algoritmaları gerçekleştiren kullanıcı dostu RapidMiner ve Weka gibi pek çok makine öğrenmesi platformları vardır. Fakat, bu geleneksel platformlar genellikle tek bir makien üzerinde çalıştığından hafıza yetersizliği nedeniyle büyük veriyi işleyememektedir. Mahout platformu büyük veri kümelerini işlemek ve analiz etmek için tasarlanmıştır. Mahout'un amacı, böl-birleştir paradigması olarak adlandırılan MapReduce ile çalışan Hadoop sistemi üzerinde kullanılan ölçeklenebilir makine öğrenmesi kütüphanelerini barındıran bir platform sunmaktır. Bu şekilde geniş veri kümelerini işlemek için tüm görev, alt görevlere bölünür ve hesaplama işlemleri tamamlanan alt kümeler tekrardan birleştirilir.

Mahout, Hadoop kümesi üzerinde çalışarak büyük veriyi işlemek için yaygın olarak kullanılan makine öğrenmesi kütüphanelerinin bir kümesidir. Mahout kütüphanesi aynı zamanda, Hadoop sistemi haricinde MapReduce paradigmasını kullanan diğer işleme sistemlerinde de kullanılabilir.

Ericson ve Pallickara [56], Hadoop ve Granules [57] olmak üzere iki farklı işleme sistemi üzerinde Mahout ile 2 sınıflandırma algoritması ve 4 kümeleme algoritmasının performanslarını değerlendirmiş ve karşılaştırmıştır. Aşağıda performansı değerlendirilen kümeleme ve sınıflandırma algoritmalarının ayrıntılı çalışmaları ve sonuçları verilmiştir:

2.6.1.1. K-means Kümeleme

Mahout farklı kümeleme algortimalarını desteklemektedir. Başlangıç olarak bu çalışmada k-means kümeleme kullanılmıştır. K-means kümeleme algoritmasında, kullanıcı bütün verileri gruplamak için yeterli kaç küme gerektiğini (k) tahmin eder.

Hadoop uygulamasında :

Mapper, her iterasyonda diskten gelen mevcut kümeleri yüklemekten sorumludur. Kümeler belleğe yüklenince, atanan noktaları okur ve noktanın ait olduğu kümeyi tanımlar. Bir nokta, kümeye atandıktan sonra, dosyaya yazılır:

```
Hadoop mapper işleminin toplam maliyeti (overall cost): CRD + NCRDWD
C : Küme sayısı
N : Kümelemeden sorumlu verilen bir mapperın nokta sayısı
RD, WD : Disk için okuma yazma süreleri (Veriyi okuma yazma süresi
aynı zamanda
çalışma zamanıdır)
Hadoop reducer işleminin toplam maliyeti(overall cost) : CRD + MNRDWD
M: Sistemdeki mapper sayısı
```

Granules uygulamasında ise şu şekildedir:

```
Granules mapper çalışma zamanı: CRS + NC + CWS
RS: Bir soket üzerinden veri aktarımının okuma maliyeti
WS: Bir soket için veri yazma maliyeti
Granules reducer çalışma zamanı: MCRS +CWS
```

Hesaplamaların çoğu mapperlarda harcandığı için, reducer toplam maliyette küçük bir rol oynamaktadır.

Her iki hadoop ve granules uygulamasında, 100 iterasyon için 88 küme üzerinde 20 k-means iterasyonu çalıştırılmıştır. Çalışma zamanlarındaki standart sapma Hadoop'ta yaklaşık 2 dakika, Granules'de 8 dakikadır. Sonuçlardan açıkça görülmektedir ki, granules uygulaması işlemi tamamlamak için gerekli disk erişim miktarı azaltılarak hadoop uygulamasına göre daha iyidir. Her iki uygulama da HDFS üzerinde çalışırken, Granules Hadoop versiyona göre daha iyi olsa da Hadoop erişim zamanını hızlandırmak için lokal verilerden yararlanabilir

2.6.1.2. Bulanık k-means kümeleme

Bulanık k-means, k-means e benzer bir prensibe göre çalışır: Kullanıcı başlangıç bir k-means küme seçer ve kümelerine atanan noktaların merkezlerini bulur. Bulanık k-means birden fazla kümeye ait bir noktaya izin vererek ekstra bir serbestlik derecesi verir. K-means ile makaleleri guruplamak ve kapsayıcı konuları bulmak mümkündür fakat birden çok konuyu kapsayan veri noktalarını işleyemez. Örneğin, bir makalede ortadoğudaki petrol

fiyatlarının konu alındığını düşünelim. K-means, ortadoğu hakkındaki makaleleri kümeleyebilir ya da hammade ücretlerinin olduğu makaleleri kümeleyebilir ancak her ikisini de barındıran makaleleri kümeleyemez. Bulanık k-means ise makaleleri her iki konu ile de ilişkilendirebilir. İlişkilendirilen konulari gösterir ve her konu ile ilişkili makalenin ilişki derecelerini de gösterebilir. Bulanık k-means, kabaca k-means ile aynı çalışır fakat tek bir kümeye ait her noktanın yerine, her nokta her kümeye ait bir olasılık atanır. Bu adımdan sonra reducer, kümeye ait en yüksek olasılıklı noktalara göre merkezi kümeyi ayarlar.

Hadoop uygulamasında:

```
Hadoop mapper işleminin toplam çalışma zamanı: RDC + RDNCWD
C : Küme sayısı
N : Kümelemeden sorumlu verilen bir mapperın nokta sayısı
RD, WD : Disk için okuma yazma süreleri (Veriyi okuma yazma süresi
aynı zamanda
çalışma zamanıdır)
Hadoop reducer işleminin toplam çalışma zamanı: MNCRD + CWD
M: Sistemdeki mapper sayısı
```

Granules uygulamasında:

```
Granules mapper çalışma zamanı: RS C + NC + CWS
RS: Bir soket üzerinden veri aktarımının okuma maliyeti
WS: Bir soket için veri yazma maliyeti
Granules reducer çalışma zamanı: MCRS + CWS
```

Granules uygulaması N faktörü ile daha hızlı bir reducer çalışma zamanına sahip olmasına rağmen, toplam çalışma sureleri Hadoop ile benzerdir. Bulanık k-means, noktaların birden fazla kümeye ait olabilmesine imkan sağladığı için calışma süresi daha uzun olmaktadır. Granules uygulaması, Hadoop uygulamasından yaklaşık 12 dakika daha erken sürede islemi bitirmektedir.

2.6.1.3. Dirichlet kümeleme

Dirichlet kümeleme k-means den oldukça farklıdır. Dirichlet kümeleme gerekli gördüğü halde küme ekleyebilir ve kaldırabilir ayrıca, farklı şekilde modelleri destekleyebilir. Bulanık k-means ve k-means kümelemenin her ikisinde de merkezi bir nokta etrafında normal dağılımlar mevcuttur. Ancak farklı bir model ile eşleşen kümelerin olduğu bir

dağıtımı işleyemez. Mahout şu anda Gaussian kümeleme, normal model ve basit normal dağıtım gibi modelleri desteklemektedir. Aynı zamanda daha fazla model tanımlanabilir. Dirichlet kümelemenin çalışma süresi, k-means ve bulanık k-meanse göre daha uzun sürebilir. Daha hızlı çalışan k-means algoritmalarına vermek üzere uygun bir k'nın belirlenmesine yardımcı olduğunda iyi bir başlangıç küme algoritmasıdır.

Hadoop uygulamasında:

```
Hadoop mapper işleminin toplam çalışma zamanı: DRD + RDNDWD
D: Model sayısi sayısı
N : Kümelemeden sorumlu verilen bir mapperın nokta sayısı
RD, WD : Disk için okuma yazma süreleri
Hadoop reducer işleminin toplam çalışma zamanı: RDD + RDMND + WDD
M: Sistemdeki mapper sayısı
```

Granules uygulamasında:

```
Granules mapper çalışma zamanı: RS D + ND + DWS
RS: Bir soket üzerinden veri aktarımının okuma maliyeti
WS: Bir soket için veri yazma maliyeti
Granules reducer çalışma zamanı: RS MD+DWS
```

Granules mapper noktaları okumaya ihtiyaç duymaz, okuma işlemini tamamen kaldırır. Her biri 40 iterasyon için çalışan, 20 Dirichlet kümeleme iterasyonu çalıştırılmıştır. Dirichlet kümeleme çalıştıran Granules, Hadooptan 5 kat daha hızlı çalışmıştır. Dirichlet kümeleme, bulanık k-means gibi çok fazla veri noktası işlemeyi gerektirmez. Sonuçlar açıkça göstermektedir ki Hadoop işlem süresinin çoğunu her adımda dosya yüklemek için harcamaktadır.

2.6.1.4. Latent Dirichlet Allocation

LDA, Dirichlet kümelemeye benzer bir kümeleme methodudur. LDA k-means gibi veri kümesindeki konu başlığı (topic) sayısını gösteren bir k değişkenine ihtiyaç duyar. Böylelikle belirli bir olasılıkla karışık konu başlıklarının olduğu tüm dökümanları tanımlayarak topic (konu) içerisindeki kelimeleri kümeler. LDA sınıflandırıcı ayrı başlıkları ayırt eder ve uygun konu başlıklarının içine her bir dökümanı kümeler. Algoritma her dökümandaki her kelimeyi okur ve her kelimenin ait olduğu muhtemel bir başlık konusu

belirler. Her konu başlığına ait dökümandaki kelimelerin sayısına göre dökümanın genel konusu tespit edilebilir.

Hadoop uygulamasında:

```
Hadoop mapper işleminin toplam çalışma zamanı: RDT|P| + RDN|P|TWD
T: Topic sayısı
N : Kümeleme için düğümler
RD, WD : Disk için okuma yazma süreleri (Veriyi okuma yazma süresi
aynı zamanda
çalışma zamanıdır)
|P|: N' deki her noktanın boyutu
Hadoop reducer işleminin toplam çalışma zamanı: NMT|P|RD + T|P|WD
M: Sistemdeki mapper sayısı
```

Granules uygulamasında:

```
Granules mapper çalışma zamanı: T|P|RS + N|P|T + T|P|WS
RS: Bir soket üzerinden veri aktarımının okuma maliyeti
WS: Bir soket için veri yazma maliyeti
Granules reducer çalışma zamanı: MT|P|RS + T|P|WS
```

Testlerde LDA 40 iterasyon ile 10 topic kümelenmiştir. Hadoop ve Granules versiyonların her ikisinde de 40 iterasyon çalıştırılmıştır. Ortalama olarak Granules, Hadoop'dan 4 dakika daha erken sürede bitirmiştir.

2.6.1.5. Naive Bayes ve Complementary Bayes

Naive Bayes ve Complementary Bayes ortalama çalışma süreleri Hadoop ve Granules'de aynıdır, bu yüzden başlangıçta uygulama tabanlı bir işlem kazancı önemsiz görülmektedir. Ancak Granules uygulamasında standart sapmada bir artış gözlemlenmiştir. Complementary Bayes'de daha fazla olsa da çok varyasyonlu olarak her iki metotta da 5 kez gözlemlenmiştir. Mahout'da sınıflandırma, 3 dakikadan daha az bir sürede genellikle hızlıdır. Bunun nedeni, Granules uygulamasında hata toleransını (Fault-tolerant) uygulamak için çalışılmamıştır. Bu durum Granules yaklaşımında küçük bir dezavantaj yaratırken, yeniden başlatılmış bir işlemin toplam eğitim (training) hızında gözle görülür bir iyileşme vereceği net değildir. Sonuçlar göstermiştir ki, işlemlerin kümeleme ve sınıflandırmasında Granules kullanmak

daha uygundur. Granules çoklu hesaplamaları yürütebildiği için, iterasyon gerektiren kümeleme algoritmaları için çok iyidir.

Esteves ve Rong [58], geniş bir veri kümesi kullanarak Mahout ile desteklenen k-means kümeleme algoritmasının performansını değerlendirmiştir. Önerilen algoritmada k-means üç adımda gerçekleştirilmiştir:

1. Başlangıç adımı:
    a. HDFS bloklarına veri kümelerinin bölünmesi, segmentasyonu
    b. Verinin kopyalanması (replication) ve diğer makinelere aktarımı
    c. blokların sayısına ve küme konfigürasyonuna göre gerekli görevlerin yapılması ve dağıtılması
2. Map adımı:
    a. Örnekler ve merkezleri arasındaki uzaklığın hesaplanması,
    b. En yakın merkez ile örneklerin eşleşmesi ve belirli kümelere atanması,
    c. Her map görevinin bir veri bloğu ile işlenmesi gerçekleştirilir.
3. Reduce adımı:
    a. Kümedeki bütün noktaların ortalama koordinatları kullanılarak merkez noktanın tekrar hesaplanması gerçekleştirilir.
    b. İlişkili noktaların yeni merkez konumunu bulmak için ortalaması alınır.
    c. Merkezi konfigürasyon mapper'a geri dönüş yapar ve merkeze (centroid) yakınlaştığında döngü sona erer.

Amazon EC2 sanal makineleri üzerinde çalışan test sonuçları göstermiştir ki, düğüm sayısı arttıkça Mahout'un çalışma ve kümeleme zamanı azalır ve dosya boyutu 66 MB'dan 1,1 GB'a arttırılırsa performans kazancı %6' dan %351' e ulaşır.

Diğer bir çalışmada [59], Mahout platformu kullanılarak k-means ve mean shift kümeleme algoritmalarının performansları karşılaştırılmıştır. Test sonuçlarına göre, dosya boyutu %50'nin üzerinde olduğunda k-means kümeleme algoritması mean shift algoritmasından daha iyi bir performans elde etmiştir.

Son yıllarda, çevrimiçi dökümanların artışı ile birlikte içerik benzerliklerine göre sayısız dökümanın organize edilmesi son derece önemli olmuştur. Literatürde döküman kümeleme ile ilgili pek çok yaklaşım sunulmuştur [38,60–64]. Döküman kümeleme ile ilgili detaylı bir araştırma [65]'de incelenebilir. Doğru kümeleme için çok fazla eğitim örneği gerekmektedir. Aynı zamanda, internet tabanlı çok büyük sayıda dökümanın işlenmesi de yüksek boyutlu bir hesaplama uzayı gerektirir. Örneğin İngiliz Wikipedia 5 TB' lık bir döküman kümesidir ve bilgi kaynağı arttıkça, döküman sınıflandırma zorlaşmaktadır. [66]'daki çalışmada, Mahout platformu üzerinde gürültülü gerçek ve büyük bir veri kümesini kümelemek için kullanılan k-means ve bulanık k-means algoritmalarının kıyaslanması gerçekleştirilmiştir. Literatürdeki [67,68] çalışmaların aksine test sonuçları, k-means algoritmasının bulanık k-means algoritmasından daha iyi çalıştığını göstermiştir. Dolayısıyla, Mahout'un büyük veri işleme ve analizi için kullanımı kolay ve etkili bir platform olduğu görülmüştür.

2.6.2. Scikit-learn

Python dilinde yazılmış ve BSD lisansı altında lisanslanmış Scikit-learn, açık kaynaklı makine öğrenmesi kütüphaneleri sunan bir çatıdır ve aynı zamanda popüler bir akademik araştırma alanıdır. İyi dökümante edilmiş, kullanımı kolay ve fonksiyonel API sağlayan Scikit-learn [69], sınıflandırma, regresyon, boyutsal azaltma ve kümeleme gibi pek çok algoritmayı gerçekleştirebilmektedir [70]. Scikit-learn'de uygulanan destek vektör makineleri, naive bayes, k-means, temel bileşenler analizi gibi pek çok algoritma büyük veri kümelerinde hızlı ve ölçeklenebilir bir performans göstermiştir.

Büyük ölçekli gözetimli öğrenme, doğru tahminler gerçekleştirmek için geniş miktarda eğitim verisi gerektirmektedir. Önerilen bir öğrenme yöntemi, önceden belirlenmiş bir eğitim seti kullanılarak test edilmekte ve daha sonra sisteme verilen giriş verisini tahmin etmek için geliştirilmektedir. Manuel olarak ya da yarı-otomatik toplanan pek çok eğitim seti yüksek maliyete yol açmaktadır. Ayrıca, aynı eğitim setinde öğrenme parametrelerini oluşturma ve test etme doğru olmayan sonuçlar getirmektedir. Bu problemi çözmek için, veri kümesi, doğrulama kümesi (validation set) ve eğitim kümesi (training set) olarak ikiye bölünür. Fakat bu işlemler, modeli öğrenmek için kullanılan örnek sayısını azaltır ve doğrulama ve eğitim kümesinden rastgele seçimlere bağlı doğru olmayan sonuçlar üretilir.

Scikit-learn bu problemi çözmek için çapraz-doğrulama (cross-validation) adı verilen etkili bir çözüm sunmuştur. Eğitim ve doğrulama performansının değerlendirilmesi yetersiz kaldığında, çapraz-doğrulama aynı veri kümesi üzerinde eğitim ve doğrulama için kullanılabilmektedir. Scikit-learn aynı zamanda, Boston ev-fiyatları veriseti, iris veriseti, rakam veriseti ve benzeri veri kümelerine sahiptir [71]. Scikit-learn, popüler dağıtık makine öğrenmesi algoritmalarının uygulaması için NumPy ve SciPy, Python modüllerini içerir ve bu modüllerin işlevselliğini kullanarak gözetimli ve gözetimsiz öğrenme algoritmalarını gerçekleştirmek için etkili bir araç seti sağlar [72].

[73]'de, genel veri madenciliği uygulamaları için Scikit-learn araçları ile diğer yazılım araçlarının karşılaştırılması sunulmuştur. Önerilen çalışma sonuçları göstermiştir ki, Scikit-learn bütün algoritma uygulamaları için iyi yazılmış bir dökümantasyon sağlar, fakat komut satırı arayüzü ile kullanımı için ileri düzey programlama bilgisi gerektirmesi diğer yazılım araçlarına göre dezavantajdır.

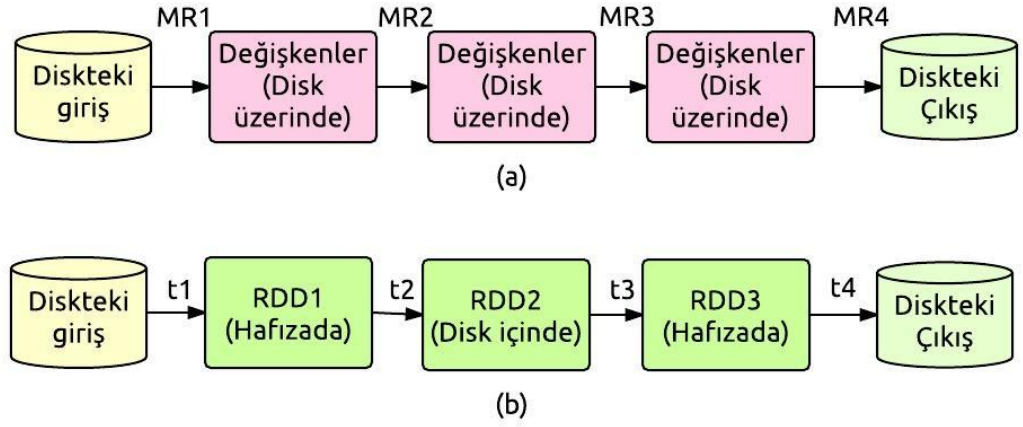
2.6.3. Spark

Spark [74], Mahout gibi Apache Yazılım Lisansı altında lisanslanan ve Berkeley AMPLab'da geliştirilen hafıza tabanlı dağıtık makine öğrenmesi platformudur. Hafızada yerleşmiş büyük veriyi analiz etmek için hızlı ve esnek bir iteratif hesaplama platformudur. Ayrıca, Hadoop'a benzer şekilde dağıtık veri ile çalışmak için Java, Pyton and Scala dillerinde yazılmış yüksek seviyeli API'ler sağlamaktadır ve Apache Hadoop için sunulmuş tek hafıza işleme çözümüdür [75]. Spark, 4 küme modunda çalışmayı destekler; Standalone, Amazon EC2, Apache Mesos ve Hadoop Yarn. Standalone modda, Spark kümesi test amaçlı lokal modunda tek bir sunucu üzerinde çalıştırılabilir.

- Standalone deploy mod; Spark bir deploy script kümesini kullanarak tek makinadan oluşan sanal bir küme üzerinde çalıştırılabilir. Ek olarak, bütün Spark işlemleri Standalone lokal modda aynı Java Sanal Makinesi (JVM) işleminde çalıştırılabilir.
- Amazon EC2; Spark kümeleri Amazon EC2 üzerinde başlatılabilir, yönetilebilir ve kapatılabilir ve kullanıcının kümesi üzerinde Spark, Shark ve HDFS kurulumu otomatik olarak gerçekleştirilir.

- Apache Mesos; Spark ve diğer platformlar arasında dinamik olarak kaynak paylaşımı sağlar. Kolay ve etkili açık kaynak küme platformudur.
- Hadoop Yarn: Uygulama master'ında çalışmak için Spark sürücülerine izin verir ve genellikle Hadoop 2 olarak adlandırılır.

Makine öğrenmesi algoritmaları, Mahout gibi MapReduce paradigmasını kullanan dağıtık makine öğrenmesi platformlarında uygulandığında, her iterasyon sürecinde diskten okuma ve yazma gerçekleştirilir. Diğer yandan Spark kullanıldığında, her iterasyon işlemi hafızada tutulur Mahout ve Spark üzerinde makine öğrenmesi algoritmalarının kullanımı Şekil 2.5'de gösterilmiştir.



Şekil 2. 5. (a) Mahout üzerinde algoritmaların çalışması, (b) Spark üzerinde algoritmaların çalışması

Hafızadan veri işleme, diskten veri işlemeden daha hızlı olacağından dolayı, Spark Mahout/Hadoop ile kıyaslandığında önemli bir performans kazancı sağlamaktadır. Spark, büyük kümelerde hafıza hesaplamaları için bir veri yapısı sağlayan ve Resilient Distributed Datasets (RDDs) olarak adlandırılan yeni bir hafıza katmanı sunmaktadır.

RDD'ler hata toleransı elde edebilir [76], yani donanım hataları ya da kullanıcı kodu hataları gibi nedenlerden dolayı verilen bir görev düşerse, kayıp veri kalan görevler üzerinde yeniden kazanılabilir (recovery) ve yapılandırılabilir (reconstruction). Dağıtık bir hesaplama platformu olan Spark, RDD içeren programlama kümesi için veri katmanı kullanarak var olan küme hesaplama platformlarından daha güçlü ve kullanışlı iteratif hesaplamalara sahiptir [77].

Sparkın son sürümleri ile birlikte, bir veritabanı (SharkQL [78] yerine Spark SQL[79])bir makine öğrenmesi kütüphanesi (MLLib [80]) ve bir graf motoru (GraphX [81]) gibi zengin araçlar sunulmaktadır. MLLib sınıflandırma, kümeleme, ortak filtreleme ve analiz (decomposition) içeren makine öğrenmesi algoritmalarını uygulamak için bir Spark bileşenidir. Spark'ın hızlı gelişimi nedeniyle, MLLib son zamanlarda fazlasıyla dikkat çekmiştir ve açık kaynak komitesinden yazılım geliştiricileri tarafından desteklenmiştir. Böylece, kullanıcı istediği algoritmayı etkili bir şekilde uygulamak için yeterince destek sağlayabilmektedir.

Hadoop için hızlı bir alternatif olması sebebiyle, Apache Spark ölçeklenebilir makine öğrenmesinde endüstriyel ve akademik alanlarsa dikkatleri üzerine çekmiştir [82,83]. [84] tarafından sunulan Spark ve Hadoop performans sonuçları göstermiştir ki, Spark WordCount ve Grep gibi basit programlar çalıştırıldığında Hadoop'dan daha iyi sonuçlar elde etmiştir. Benzer bir çalışmada [85], dosya boyutu çok küçük olsa bile Spark üzerinde çalışan k-means algoritmasının MapReduce' den 5 kat daha hızlı olduğu gösterilmiştir. Buna karşın, işlem süresince veri kümesi sabit bir şekilde değişirse, Spark MapReduce üzerindeki avantajını kaybetmektedir. Lawson [86], Apache Spark kullanarak optimizasyon problemlerini çözmek için Alternating Direction Method of Multipliers (ADMM) olarak adlandırılan dağıtık bir metod önermiştir. İterative algoritmalarda etkisiz olması sebebiyle MapReduce platformu yerine Spark üzerinde önerilen dağıtık metodu uygulamayı tercih eden diğer bir çalışmanın sonucu [87] göstermiştir ki, dağıtık Newton yöntemi Spark'ın sağladığı hata toleransı ile lineer destek vektör makineleri ve lojistik regresyon testlerinde başarılıdır. [88]'da k-means ve naive bayes algoritmaları kullanılarak Hadoop, Spark ve DataMPI platformlarının performans karşılaştırılması yapılmıştır. Sonuçlar göstermiştir ki, %39 ve %41 oranlar ile sırasıyla DataMPI ve Spark, Hadoop'dan daha etkili bir şekilde CPU kullanmıştır. Hastaneler gibi çeşitli kurumlardan toplanan yüksek boyutlu verilerden bilgi çıkarımı, büyük ölçekli veri kümeleri üzerinde gerçek-dünya uygulamalarında en iyi bilinen problemlerdir. [89]'deki çalışmada, Spark platformu tabanlı bir paralel apriori algoritması olan YAFIM (Yet Another Frequent Itemset Mining) sunulmuştur ve hastaneler ile diğer sağlık kurumlarından toplanan veriler üzerinde bu algoritma test edilmiştir. Önerilen paralel apriori algoritması MapReduce platformundaki uygulamasından 18 kat daha hızlı çalışmıştır. Bütün bu çalışmalar göstermiştir ki, Spark iteratif hesaplamalar için iyi bir

çözümdür ve ölçeklenebilir makine öğrenmesi algoritmaları için MapReduce tabanlı diğer dağıtık makine öğrenmesi platformları ile kıyaslandığında pek çok avantaj sağlamaktadır.

2.6.4. GraphLab

C++ dilinde yazılmış ve Apache Yazılım Lisansı altında lisanslanmış GraphLab [90], graf tabanlı dağıtık bir makine öğrenmesi platformudur. Bilgisayar ağ güvenliği, yazılım teşhisi, öneri sistemleri, hesapsal biyoloji ve sosyal ağ analizi gibi popüler pek çok gerçek-dünya uygulamasında kullanılmaktadır [91]. [92]'da GraphLab'ın, belirli hesaplama, veri bağımlılıkları ve zamanlama sorunlarını çözmek için büyük ölçekli makine öğrenmesinde güçlü ve hızlı graf paralel hesaplama sunduğu belirtilmiştir. GraphLab platformu, Round Robin, Priority ve FIFO gibi çeşitli zamanlayıcılar (schedulers) ve uygun tutarlılık seçenekleri ile her paralel çalışma için sıralı tutarlılığı (sequential consistency) garantileyen tam tutarlılık, köşe tutarlılık ve kenar tutarlılık gibi tutarlılık modellerini sağlar [90].

Büyük veri üzerinde iteratif hesaplamalar asenkron olarak daha yüksek performans sergilemektedir. Senkron hesaplamada bir işlem ilgili hesaplamalarını tamamlayana kadar diğer işlemleri beklediğinde, tüm çalışma zamanı bu gecikmeler ile birlikte artmaktadır. Bu problemin üstesinden gelmek ve pek çok iteratif makine öğrenme algoritmalarını daha kısa sürede uygulamak için, GraphLab asenkron ve graf-paralel hesaplama sunmaktadır. Spark, Mahout ve Pregel [93] (diğer bir graf-tabanlı dağıtık hesaplama platformu) gibi platformlar senkron bir iteratif hesaplama kullanırken, Graphlab asenkron hesaplamayı destekleyerek çalışma zamanında önemli bir azaltma ile diğer senkron platformlara göre avantaj kazanır. [94]'de Apache Mahout ve GraphLab platformlarında uygulanan çeşitli makine öğrenmesi algoritmaları karşılaştırılmıştır. Deneysel sonuçlar göstermiştir ki, GraphLab çalışma zamanı çok daha kısadır. Fakat, hata ölçümü dikkate alındığında Mahout, GraphLab'dan daha iyi çalışmaktadır. Dağıtık GraphLab'da hata toleransı checkpoint'ler kullanarak elde edilir. Böylelikle verilen bir görev donanımsal ya da kullanıcı hata kodundan dolayı düşerse son checkpoint'den onarılır.

2.7. Dağıtık Makine Öğrenmesi Platformlarının Karşılaştırılması

Tablo 2. 2. Dağıtık makine öğrenmesi platformlarının karşılaştırılması

	Mahout	**Scikit-Learn**	**Spark**	**GraphLab**
Kütüphane	Java	Python	Java, Python, Scala	C++
Model üzerinde iteratif hesaplama	Senkron	Senkron	Senkron	Senkron & Asenkron
Hata Toleransı	Görevler Master tarafından tekrar başlatılır.	Hata toleransına odaklanmamıştır.	Kalan görevler otomatik olarak tekrardan yapılandırılır.	Hata toleransı yoktur fakat dağıtık GraphLab' da, kayıp veri son chechpoint ile tekrardan iyileştirilir.
Avantajlar	Popüler makine öğrenmesi algoritmaları için kullanımı kolay ve etkili bir platformdur.	Büyük veri kümleri için hızlı ve ölçeklenebilirdir, çapraz-doğrulama ve standart veri kümeleri sunar.	RDD olarak adlandırılan yeni bir hafıza katmanı sunar, hızlı ve esnek iteratif hesaplama sağlar.	Uygun tutarlılık seçenekleri ile her paralel çalışma için sıralı tutarlılığı (sequential consistency) garantiler, asenkron ve graf paralel hesaplama sağlar.
Dezavantajlar	Doğru kümeleme sonuçları için iş bellek yapılandırma parametrelerini ayarlamak için dikkat gerektirir.	İstatistiklere daha az odaklanmıştır, araçlar ve kütüphaneler dağınıktır, komut satırlı arayüze sahiptir.	Metin dosyaları yavaşça kaydedilir, çoklu bilgisayarlar arasında dağıtık veriyi takip etmek zordur, Spark üzerinde Pig kullanmak kullanışsız bir yöntemdir.	Doğal grafların bölünmesi için verimsiz ve kötü performans sergiler.
Önerilen Algoritmalar	Naïve Bayes, HMM, logistic regression, random forest, k-means, fuzzy k-means, canopy, spectral clustering, latent drichlet allocation, collaborative filtering	SVM, nearest neighbors, random forest, SVR, ridge regression, k-means, spectral clustering, PCA, future selection, grid search, preprocessing	Lineer SVM ve lojistik regresyon, sınıflandırma ve regression tree, k means clustering, singular value decomposition, lineer regresyon Naive Bayes, basic statistics, feature transformations	Collaborative filtering, k-means, PageRank, latent drichlet allocation, Jacobi method for linear solver, distributed dual decomposition, Image-Stitching, graph analytics

2.8. Sonuçlar

Bu tez çalışmasında, çok büyük veriyi işlemek ve analiz etmek için kullanılan Spark, Mahout, Scikit-learn ve GrapLab açık kaynak-dağıtık makine öğrenmesi platformları karşılaştırılmış ve değerlendirilmiştir. Karşılaştırma tablosu, Tablo 2.2'de sunulmuştur. Literatürde var olan çalışmalar detaylı bir şekilde incelenerek seçilen dağıtık makine ögrenmesi platformalarının avantajları ve dezavantajları açıkça ortaya konulmuş ve yapılan araştırmaların büyük verileri işleme çalışmalarına genel bir bakış açısı sunması hedeflenmiştir.

Yapılan çalışmalar göstermiştir ki; Mahout popüler makine öğrenmesi algoritmaları için kullanımı kolay ve etkili bir platformdur, ancak her iterasyon sürecinde diskten okuma ve yazma gerçekleştirildiği için önemli performans düşüşleri görülmüştür. Hafızadan veri işleme, diskten veri işlemeden daha hızlı olacağından dolayı, Spark, Mahout/Hadoop ile kıyaslandığında önemli bir performans kazancı sağlamaktadır. Spark, büyük kümelerde hafıza hesaplamaları için bir veri yapısı sağlayan ve resilient distributed datasets (RDDs) olarak adlandırılan yeni bir hafıza katmanı sunmaktadır.

Büyük veri üzerinde iteratif hesaplamalar asenkron olarak daha yüksek performans sergilemektedir. Senkron hesaplamada bir işlem ilgili hesaplamalarını tamamlayana kadar diğer işlemleri beklediğinde, tüm çalışma zamanı bu gecikmeler ile birlikte artmaktadır. Spark, Mahout ve Scikit-learn senkron iteratif hesaplama kullanırken, Graphlab asenkron hesaplamayı destekleyerek çalışma zamanında önemli bir azaltma ile diğer senkron platformlara göre avantaj kazanır.

Scikit-learn hata toleransına odaklanmaması sebebiyle, diğer platformlara göre veri güvenliği açısından dezavantaj sağlamaktadır, çapraz-doğrulama özelliği ile algoritma sonuçlarının daha doğru performans karşılığı almayı hedeflemektedir.

GraphLab doğal grafların bölünmesi için verimsiz ve kötü performans sergilerken, uygun tutarlılık seçenekleri ile her paralel çalışma için sıralı tutarlılığı (sequential consistency) garantilemektedir.

3. DAĞITIK BÜYÜK VERİ UYGULAMALARI

3.1. Giriş

Tezin bu bölümünde 2. Bölümde anlatılan büyük veri teknolojileri kullanılarak, tez çalışması kapsamında geliştirilen iki ana büyük veri uygulaması sunulmuştur. Bunlar Dağıtık Çağrı Merkezi Analiz Sistemi ve Dağıtık Okunabilirlik Analiz Sistemidir.

Çağrı merkezleri için kalite izleme, bir müşteri hizmetleri temsilcisinin performansını ölçmek için kaydedilen çağrıları dinleme işlemi olarak tanımlanabilir. Kalite izlemenin ana sorunu, yöneticilerin tüm kayıtları dinlemek için zamana sahip olmaması ve bu nedenle kayıtlı çağrıların sadece bir kaçının rastgele seçilmesidir. Bu durum, çağrı kayıtlarının çoğu dinlenemediği için hatalı performans ölçümlerine sebep olur. Bu tez çalışmasında, kaydedilen tüm çağrıları, çeşitli kalite kriterlerini kullanarak değerlendirmek için geliştirilen, dağıtık bir çağrı izleme sistemi sunulmaktadır. Önerilen sistemde, popüler Hadoop MapReduce çerçevesini kullanarak çok sayıda çağrı kaydını analiz edilebilmekte ve kosinüs ve n-gram gibi metin benzerlik algoritmalarını kullanılmaktadır. Ayrıca argo kelime listeleri de izleme sistemine entegre edilmiştir. Önerilen çağrı izleme sisteminin performansını göstermek için deneysel çağrı kayıtları kullanılmıştır.

Okunabilirlik, bir metnin okuma zorluğu seviyesini ölçmek için kullanılan bir terimdir. Farklı diller için geliştirilmiş birçok okunabilirlik formülü ve yaklaşımı bulunmaktadır. Bu tez çalışması kapsamında okunabilirlikle kapsamlı bir ilgili literatür taraması yapılmış ve Türkçe ders kitaplarının okunabilirlik seviyelerini tespit etmek için kullanılabilecek dağıtık bir okunabilirlik analizi sistemi geliştirilmiştir. Geliştirilen sistemin yapısı, kullanılan teknolojiler, mevcut sistemlere göre avantajları ve örnek veri setleri üzerindeki performans çıktıları bu bölümde verilmiştir.

3.2. Dağıtık Çağrı Merkezi Analiz Sistemi

3.2.1. Özet Ve Motivasyon

Çağrı merkezleri için hizmet kalitesi, gerçek hizmet performansı ile müşteri beklentilerinin karşılaştırılması sonucu ortaya çıkmaktadır. Müşteri temsilcilerinin,

müşterilere verdikleri hizmet kalitesini değerlendirmek ise ürün kalitesini değerlendirmekten daha zordur. Çağrı merkezlerinde performans ölçümü, müşteri temsilcileri ile müşteriler arasında geçen kayıtlı konuşmalar rasgele dinlenerek ve görüşmelerdeki kelimeler tek tek incelenip değerlendirilerek yapılmaktadır. Piyasada var olan bu performans yönetim sistemleri, tüm görüşmeler takip edilemediği için gerçek performans karşılığını alamamakta, yönetici ya da danışmanlıklar tarafından kayıtlar dinlendiği için zaman ve maliyet açısından dezavantaj oluşturmaktadır. Buna karşılık önerilen sistem, iç ve dış çağrı kayıtlarını dağıtık bir şekilde işleyen bulut tabanlı bir performans ölçüm sistemi sunarak müşteri memnuniyeti, satış ve pazarlama, hizmet kalitesi ve performans yönetiminde yüksek bir performans ile önemli bir katkı sağlamaktadır.

Çağrı Merkezleri Derneği'nden alınan 2013 verilerine göre Türkiyede, 300 adet büyük ve orta ölçekli 1000 adet çağrı merkezi ve yaklaşık 49.000 çağrı masası bulunmaktadır. 1000 çağrı masasına sahip bir şirkette, her bir çağrı masasının saatte 20 çağrıya cevap verdiği düşünüldüğünde günlük yaklaşık 1TB ses verisi oluştuğu düşünülebilir. Mevcut sistemler, günlük 1TB veriyi işleyebilecek donanım ve yazılıma sahip değildir. Buna karşılık, bulut teknolojisi yardımı ile oluşturulan, örneğin 100 ticari bilgisayar ile bu veriler paralel olarak kısa bir sürede işlenebilir.

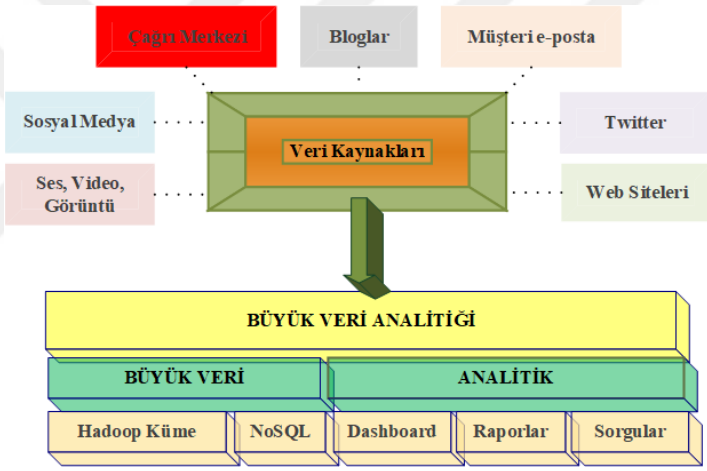
Mevcut teknolojiler ve sistemler incelendiğinde görüşme verilerinin depolama yetersizliği, durum analizi için kullanılan veri madenciliği yöntemleri, çalışan maliyetini azaltmak ve zaman verimliliğini arttırmak için otomatik bir yönetim yazılımı ihtiyacı görülmektedir. Önerilen çözüm, günlük yaklaşık 1TB verinin işlenebilmesi, yani görüşmelerin içerik analizinin yapılarak müşteri temsilcisi performansının gerçek karşılığının değerlendirilmesi için bulut alt yapısı kullanmaktadır. Dağıtık yapıdaki donanım üzerinde sanal makineler çalıştırılarak oluşturulan bulut alt yapısında, sanal makineler eldeki kaynaklar sınırında ihtiyaç duyulduğu kadar arttırılabilir ve azaltılabilir.

Mevcut sistemlerde yöneticiler tarafından aylık 5 görüşmeden az olmamak üzere, rassal olarak müşteri temsilcilerinin görüşme kayıtları dinlenerek değerlendirilmektedir. Böyle bir sistemde, performans ölçümünün gerçek karşılığını almak olanaksızdır.

Günümüzde Twitter paylaşımları, Facebook sayfaları, müşteri e-postaları, çağrı merkezi kayıtları, bloglar, forumlar ve diğer web sayfaları gibi kaynaklardan, yapılandırılmamış veya yarı yapılandırılmış formatta büyük miktarda bilgi toplanmakta ve bunlar farklı depolama sistemlerinde saklanmaktadır [95]. Metin analitiği, iş zekası, medya analizi, müşteri ilişkileri

yönetimi (CRM) ve tahmine dayalı analitik gibi çeşitli uygulamalarda kullanım için bu verilerden yararlı ve yeni bilgiler elde etmeyi amaçlamaktadır. Geleneksel makine öğrenme teknikleri, NLP, bilgi edinme ve bilgi yönetimi kullanarak metin madenciliğinin önünde, farklı dil özellikleri ve büyük veri gibi çeşitli zorluklar bulunmaktadır [96].

Büyük veri ile ilgili güçlükler genellikle verinin çeşitliliği, hacmi, üretilme hızı, doğruluğu ve değeri ile ilişkilidir. Çeşitlilik, müşteri e-postaları, sosyal medya verileri, ses ve video verileri gibi farklı biçimlerde yapılandırılmamış veriyi ifade eder. Hacim, büyük miktarda veriyi, hız, verinin ne kadar hızlı üretildiğini ve ne kadar hızlı analiz edilmesi gerektiğini, doğruluk verilerin güvenilirliğini ya da karar alabilmek için ne kadar güvenilir olduğunu, değer ise farklı kurum/kuruluşlar tarafından saklanan verilerin değerini ifade eder [97].



Şekil 3. 1. Büyük veri analitiğinin temel bileşenleri

Veri setlerinin miktarı ve boyutları katlanarak büyüdükçe, bu veriler üzerinde çalışabilecek gelişmiş analitik teknikleri daha fazla önem kazanmaya başlamıştır [2]. Büyük veri analitiği, Şekil 3.1'de gösterilen büyük veri ve analitiğin birleşimi olarak, verimlilik, gelir ve müşteri memnuniyetinin artırılması, maliyetlerin düşürülmesi ve işletmelerin kalite güvencesinin karşılanmasında büyük bir etki yaratmıştır. Her ne kadar yeni teknolojilerin bir etkisi olarak, günümüzde her zamankinden daha fazla veri olsa da, birçok kuruluş veri setlerinden değerli bilgi elde etmek için geleneksel yöntemlere göre daha iyi seçenekler aramaktadır [98]. MapReduce çerçevesi [29], büyük miktarda veriyi verimli bir şekilde analiz etmek için ana seçeneklerden biri olarak ortaya çıkmıştır. Büyük veri işleme için en

popüler paradigma olan MapReduce, Hadoop gibi açık kaynak projelerde uygulanmaktadır [30]. Hadoop, yapılandırılmış ve yapılandırılmamış verileri analiz etmek için esnek ve ölçeklenebilir bir çerçeve sunar ve Hadoop Dağıtılmış Dosya Sistemi'ni (HDFS) hatasız bir şekilde kullanarak bağımsız diskler üzerindeki verileri dağıtır.

Bu çalışmada, çağrı merkezi temsilcilerine ait çağrı kayıtlarını izleme ve analiz etme zorluklarına karşı bazı çözümler sunulmaktadır. Bu çalışmanın temel amacı, kaydedilen tüm çağrıları analiz etmek ve büyük veri analizi ve metin madenciliği teknikleri kullanarak müşteri temsilcisinin performansını ölçmektir. Özellikle aşağıdaki anahtar avantajları sunmaya çalışmaktadır:

- Müşteri çağrısı verilerini değerli bilgiye dönüştürmek
- Daha iyi kararlar verebilmek için yeni bilgiler edinmek
- Rekabet avantajı elde edebilmek ve zaman kazanmak
- Önemli oranda maliyet tasarrufu edinmek
- Karı arttırmak ve müşteri memnuniyeti elde etmek
- Tüm gelen ve giden çağrıları analiz ederek, doğru performans ölçümleri ile sonuçlanan güvenilir bir izleme sistemi sunmak.

3.2.2. İlgili Çalışmalar

Çağrı merkezleri, telekomünikasyon, finans, ulaşım, sağlık, otomotiv vb. pek çok sektöre hizmet vermektedir. Birçok çalışma, müşteri temsilcisi performansını değerlendirebilmek için çeşitli yaklaşımlar ve çözümler önermiştir. Takeuchi [99], bir müşterinin araba rezervasyonu yapma niyetinde olup olmadığını bulmak için Tetik Bölütleme Tespiti ile bir kiralık araba rezervasyon şirketince kaydedilen çağrıları analiz etmiştir. Mishne [100], metin analizi ve bilgi çıkarımı yöntemlerini kullanan bir çağrı merkezi izleme sistemi önermiştir. Sistem çağrı merkezi konuşmalarının içeriğini analiz etmek ve çağrıda belirtilen ana sorunu tespit etmek için kullanılır. [101]'da sunulan proje, Fransızca çağrı merkezi kayıtları için konuşma analiz sistemi ile desteklenen otomatik konuşma tanıma ve metin madenciliği teknolojileri kullanmaktadır. Konuşma analitiği ve metin madenciliğinde mevcut yaklaşımadan farklı olarak, [102]'de, pragmatik analiz üzerine inşa edilen etkileşimli madencilik aracı yardımıyla çağrı merkezi analitiği sunulmuştur. Araç, bankacılık

alanındaki bir yardım masası çağrı merkezinin, manuel olarak metne çevrilen ve 213 konuşmasından oluşan bir derleme uygulanmıştır. Kopparapu [103] çağrı merkezleri için sorunlu çağrıların otomatik olarak tanınmasını sağlayan bir yöntem geliştirmiştir

3.2.3. Müşteri Temsilcisi Performansı

Çağrı merkezi, insan, süreç, teknoloji ve stratejilerden oluşan müşteri ilişkileri yönetiminin önemli bir parçasıdır. Bir çağrı merkezinin hizmet kalitesi, gerçek servis performansının ve müşteri beklentilerinin karşılaştırılması ile hesaplanabilir. Müşteri hizmetleri temsilcisinin müşteriye sunduğu hizmet kalitesinin ölçülmesi, ürün kalitesini değerlendirmekten daha zordur. Çağrı merkezlerinde performans değerlendirmesi genellikle kaydedilen aramalardan rastgele seçilen aramaları dinleyerek ve ilgili konuşmada kelimeleri tek tek değerlendirerek gerçekleştirilir. Bununla birlikte, böyle bir değerlendirme, gerçek kişilerin kayıtlı çağrıları dinlemesi gerektiğinden ve zaman kısıtları sebebiyle yalnızca birkaç aramanın tam olarak değerlendirilebilmesi nedeniyle zaman ve maliyet açısından verimsiz bir yöntemdir.

Bu nedenle, çalışan maliyetlerini düşürmek ve zaman verimliliğini artırmak için otomatik performans değerlendirme sistemlerine ihtiyaç duyulduğu açıktır. Geleneksel yaklaşımlar, depolama ve içerik analizi açısından, paralel olarak yüzlerce konuşmayı işlemek gerektiğinde önemli kısıtlamalara sahiptir.

Önerilen sistem, kaydedilen tüm konuşmaları saklamak ve bunları dağıtık metin analizi yöntemleri kullanarak analiz etmek için büyük veri teknolojileri kullanmaktadır. Dağıtık depolama ve analiz sistemini çalıştırmak için gerekli olan sanal makineler, sistemin, girdi sayısına göre ölçeklenebilmesini sağlayan bir bulut mimarisi üzerinde çalıştırılmaktadır.

Çağrı merkezi temsilcisinin performansının önerilen sistemi kullanarak ölçülmesi, çağrı merkezleri için aşağıdaki önemli avantajları sunar:

- *Teknolojik yenilik:* Bulut alt yapısı ile sektöre ve hizmet sağlayıcılarına güvenilir ve ölçeklenebilir bir depolama çözümü sunulması, gerçeğe yakın zamanlı tüm görüşmelerin otomatik analizi gerçekleştirilmesi, yöneticilere erken uyarı sistemi sunulması, metrik tabanlı analiz, otomatik müşteri temsilcisi performans

puanlanması ve günlük, haftalık, aylık performans analizi ile var olan teknolojilere oranla yüksek bir performans sağlamasıdır.

- *Personel maliyetlerinin düşürülmesi:* Önerilen sistem ile çağrı merkezlerindeki görüşme kayıtları, personel yerine otomatik sistem tarafından analiz edileceği için çalışan personel sayısı ile orantılı olarak personel maliyetinin azaltılması sağlanabilir.

3.2.3.1. Performans Metrikleri

Aşağıdaki performans metrikleri, bir müşteri temsilcisinin performansını değerlendirmek için kullanılır:

- Aracıların gelen çağrılara cevap verme sürelerini ve günlük konuşma miktarlarını tahmin etme ve puanlandırma (çağrılara cevap verme hızı).
- Soruna bir çözüm getirmek için kaç çağrının yeterli olduğunu ve aynı konuda tekrar eden çağrıların olup olmadığını değerlendirmek (talepler üzerindeki hız).
- Puanlama sistemi ile görüşmelerin değerlendirilmesi (güvenilirlik).
- Gerekli olan açılış ve kapanış kelimelerinin söylenip edilmediğini değerlendirmek
- Argo kelimeler kullanılıp kullanılmadığı (garanti).
- Duygusal durumların, içerik analizi ve duygu analizi yardımıyla, öfke, kararlılık, sinirlilik, sessizlik vb şeklinde sınıflandırılması (empati ve cevap verme yeteneği).

3.2.3.2. Metod ve Teknolojiler

Önerilen sistem aşağıdaki teknolojileri kullanmaktadır:

- Sistem için geliştirilen analiz kodlarını paralel olarak çok sayıda sanal makine üzerinde çalıştırabilmek için gerekli sunucu altyapısını sağlayabilmek amacıyla, özel amaçlı bulut altyapısı oluşturulmuştur. Bunun için OpenStack açık kaynaklı buldu bilişim sistemi kullanılmıştır.
- Kaydedilen çağrılar Google Speech API kullanılarak metin dosyalarına dönüştürülür ve dönüştürülen veriler dağıtık dosya sisteminde saklanır.

- HBase (Hadoop üzerinde çalışan büyük veri için ölçeklenebilir, dağıtık NoSQL veritabanı), Hive (büyük veri üzerinde çalışan iş zekası sistemi), Mahout (ölçeklenebilir makine öğrenimi ve veri madenciliği kütüphanesi), Pig (Paralel hesaplamalar için yüksek seviyeli veri akışı dili ve uygulama kütüphanesi), ZooKeeper (dağıtık uygulamalar için yüksek seviyeli koordinasyon uygulaması) ve Kosinüs ve n-gram benzerlik algoritmaları, bulut yapısı üzerindeki büyük verileri analiz etmek için kullanılmıştır.
- JSF (Java Server Faces), PrimeFaces (CSS-JS kütüphanesi) ve MongoDB (doküman tabanlı NoSQL veritabanı) teknolojileri yardımıyla kullanıcı arayüzleri geliştirilmiştir.

3.2.4. Büyük Veri Analitiği

Veri akışı sırasında verinin gizli kalması ve en önemlisi de büyük veri, doğru analiz metotlarıyla yorumlandıktan sonra çıkan verinin anlamlı bir değer yaratması gerekir. Bu çalışmada büyük veri setleri üzerinde yüksek performansla analitik işlemleri yürütebilmeyi sağlayan Apache Hadoop kullanılmıştır.

Açık kaynaklı bir MapReduce modeli uygulaması olarak Apache Hadoop, büyük veri işleme ve depolama için kayda değer bir çözüm olarak ortaya çıkmıştır. Apache Hadoop ölçeklenebilir bir şekilde dağıtık işlemler gerçekleştirebilmek için açık kaynak kodlu yazılımlar geliştirmeyi sağlar. Apache Hadoop yazılım kütüphanesi, büyük veri setlerinin dağıtık bir şekilde işlenmesi için olanak sağlayan bir çerçevedir. Apache Hadoop binlerce petabyte veri üzerinde bağımsız bilgisayarlarla çalışmak için çeşitli araçlar sağlar. Hadoop, Google tarafından geliştirilen MapReduce ve Google File System teknolojilerinin açık kaynaklı olarak geliştirilmesi sonucunda elde edilmiştir [30]. Hadoop ekosistemi, dağıtık dosya sistemi HDFS ve veri setleri üzerinde hesaplama yapmak için kullanılan MapReduce teknolojilerini barındırır. Hadoop'un iki ana bölümü vardır: girdi ve çıktı dosyalarını depolayan Hadoop Dağıtık Dosya Sistemi (HDFS) ve büyük hacimli verileri işleyen MapReduce çerçevesi.

HDFS [31] hataya dayanıklılık sağlayan ve pahalı olmayan donanımlar üzerinde çalışacak şekilde tasarlanmış bir dağıtık dosya sistemidir. HDFS uygulama verilerine yüksek

kapasiteli erişim sağlar ve büyük veri setleri uygulamalar için uygundur. MapReduce programlarının binlerce sunucu üzerinde çalışmasına olanak sağlayan bir dağıtık dosya sistemidir. HDFS master/slave mimarisine sahiptir. Büyük veriler otomatik olarak bölümlendirilerek Hadoop kümeleri tarafından farklı düğümlerce yönetilir [30]. HDFS iki bileşenle gelir; NameNode olarak bilinen tek bir ana düğüm ve DataNodes olarak bilinen hesaplama düğümleri. DataNode HDFS üzerinde gerçek verileri depolarken, NameNode HDFS meta verilerinin depolanmasından sorumludur.

MapReduce, 2004 yılında Google tarafından bilgisayar kümeleri üzerinde büyük veri setlerinin hesaplamasını dağıtık bir şekilde yapmayı destekleyecek şekilde geliştirilmiş bir programlama modelidir**.** MapReduce çerçevesi, hata toleransı, otomatik paralelleştirme, ölçeklenebilirlik ve veri lokalizasyonu tabanlı optimizasyonları içeren birçok özellik sunmaktadır [104]. Bir ana düğüm tarafından kontrol edilen MapReduce işlemleri, Map ve Reduce adındaki iki fonksiyona ayrılır. Map fonksiyonu, giriş verilerini bir grup anahtar-değer çiftine böler ve her bir map görevinin çıktısı anahtarlarına göre sıralanır. Reduce işlemi, bu değerleri nihai sonuç olacak şekilde birleştirir [105]. MapReduce çeşitli programlama dillerinde yazılabilir.

Map Fonksiyonu: Ana düğüm daha küçük alt düğümlere bölümlendirilir ve bu düğümler işçi düğümlere dağıtılır. İşçi düğümü küçük problemi işler ve ana düğüme geri cevap gönderir. Map bir veri etki alanındaki bir çift veri tipini alır ve farklı bir etki alanında yeni bir veri çifti döndürür.

$$Map(k_1, v_1) \rightarrow List(k_2, v_2) \tag{3.1}$$

Reduce Fonksiyonu: Ana düğüm tüm alt düğümlerin cevaplarını toplar ve problemin cevabını oluşturacak şekilde onları birleştirir. Reduce fonksiyonu aynı işlemi aynı etki alanında oluşan bütün koleksiyon gruplarına uygular.

$$Reduce(k_2, List(v_2)) \rightarrow List(v_3) \tag{3.2}$$

Büyük veri, verilerin hacminin inanılmaz büyüme oranlarıyla artmakta olduğu veri kümeleri olarak kabul edilir. Böyle büyük veri kümeleri, geleneksel ilişkisel veritabanı

yönetim sistemleri tarafından etkin bir şekilde yönetilemez. Bu problemin üstesinden gelmek için, NoSQL veritabanı yönetim sistemleri, güçlü tutarlılık, yüksek kullanılabilirlik ve bölünme toleransı özelliklerine sahip olan dağıtık bir çözüm sunmaktadır [106]. NoSQL veritabanları, [107] tarafından üç popüler kategoride sınıflandırılmıştır: SimpleDB gibi anahtar-değer sistemleri, HBase ve Cassandra gibi sütun tabanlı veritabanları ve MongoDB gibi belge tabanlı depolama sistemleri. NoSQL veritabanları ilişkisel veritabanlarına göre aşağıdaki avantajlara sahiptir:

- NoSQL veritabanları, genellikle büyük verileri ilişkisel veritabanlarından daha hızlı ve daha verimli işler.
- Arama veri kayıtları, belge ve e-posta arşivleri, web günlükleri, sosyal medya etkileşim verileri ve sensör verileri için yüksek düzeyde ölçeklenebilir ve büyük hacimli veri depolama imkânı sağlarlar.
- Otomatik kurtarma, daha kolay veri dağıtımı, esnek ve basit veri modelleri, yüksek kullanılabilirlik oranları ve Hadoop / MapReduce ile entegrasyon sunarlar.
- NoSQL veritabanları açık kaynaklıdır ve ucuzdur. Büyük verileri ve ilgili işlemleri yönetmek için daha ucuz sunucular ve depolama sistemleri kullanırlar; ilişkisel veritabanları ise büyük depolama maliyetleri ve pahalı sunucular gerektirir.

Bu projede de kullanılan NoSQL veritabanı HBase, çok büyük boyutlara sahip verilere gerçek zamanlı okuma/yazma erişimi yapmak gerektiği zamanlarda kullanılır. Hbase JAVA kullanılarak yazılmıştır ve yapısı geleneksel veritabanı yapısından farklıdır. Örneğin aynı tabloda bulunan iki satır birbirinden çok farklı sütunlara sahip olabilir. Hbase ve Google BigTable veritabanlarının temeli map kavramına dayanmaktadır. Map kavramı Hash olarak düşünülebilir, yani veriler anahtar-değer çiftleri şeklinde tutulur. Hbase ve BigTable dağıtık olarak tasarlanmıştır dolayısıyla veriler birden çok makineye dağıtılarak saklanabilir. Dolayısıyla Hbase kullanılırken Dağıtık Dosya Sistemi kullanılır. Hbase, anahtar-değer çiftlerini alfabetik sıraya uygun olarak tutar ve özellikle seyrek veritabanları açısından çok verimlidir. Seyrek veritabanına örnek olarak, 1000 satırdan oluşan bir tabloda sadece bir satır için belli bir sütunda veri tutulması verilebilir. Hbase'in bir diğer güçlü yanı her veri için zaman damgası bilgisi tutmasıdır.

3.2.5. Konuşma Analitiği

Konuşma analitiği, müşteri temsilcisi ve müşteri arasındaki konuşmalardan, yeni ve faydalı bilgiler çıkarabilmek için, tüm kayıtlı çağrıları otomatik olarak analiz etme sürecini ifade eder. Genellikle üç adımda gerçekleştirilir:

- Otomatik konuşma tanıma,
- NLP,
- Metin madenciliği.

Konuşma analitiğinin ilk adımı, ses kayıtlarının metin verilerine dönüştürülmesidir. NLP, cümle sınır tespiti, cümlenin bileşenlerine ayrılması, varlık çıkarma gibi işlemler yardımıyla metin madenciliği teknikleri çalıştırmak için bir iskelet oluşturur.

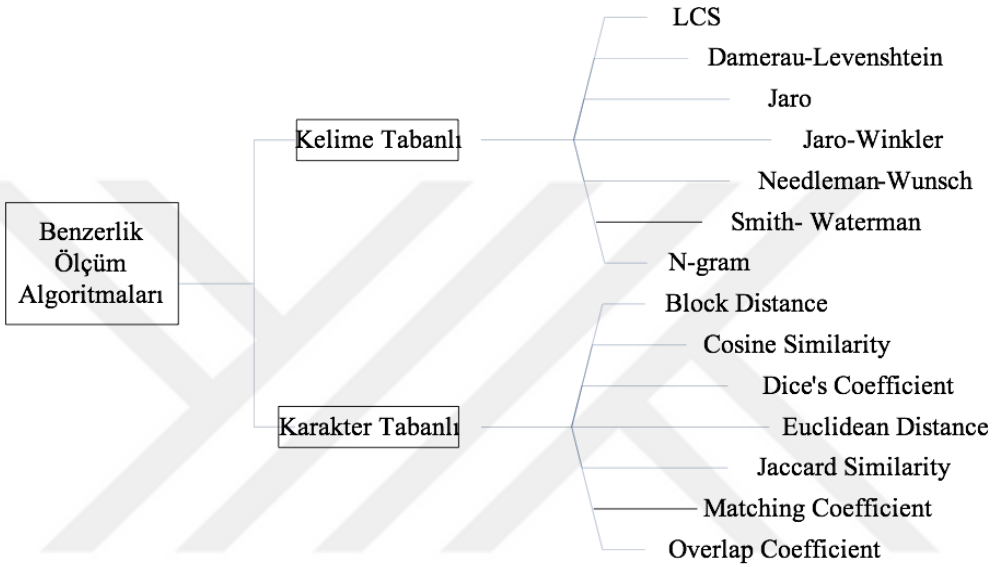
İkinci adımda, konuşma metni üzerinde müşteri performansının ölçülmesi için kullanılan metriklerle ilgili işlemler gerçekleştirilir. Bu amaçla konuşma metni içerisinde, selamlama ve kapanış cümlelerinin geçip geçmediği, argo kelime kullanımı, müşteri isminin tekrar edilip edilmediği gibi çeşitli kontrollerin yapılması gerekmektedir. Bu kontroller, 3.2.6. bölümünde anlatılan metinler arası benzerlik tespit algoritmaları ve sözlük tabanlı kontroller kullanılarak gerçekleştirilmiştir. Sonuç olarak, bu çalışmada, izleme ve performans yönetimi için doğru, eksiksiz ve güvenilirlik gibi kalite kriterlerini sağlayarak çağrı merkezi görüşmelerini analiz etmek için kullanılabilecek bir konuşma analizi sistemi sunulmuştur.

3.2.6. Metinler Arasında Benzerlik Tespiti

Internet kullanımının artması ile birlikte online doküman sayısı yüksek hızla artmaktadır. Online doküman sayısının artışı nedeniyle, çok büyük metin verileri arasındaki benzerlik ölçümlerinin hesaplanması pek çok zorlukla birlikte önemli bir NLP çalışma alanı olmuştur. Bu metinlerin içeriklerine göre benzerliklerinin tespit edilmesi ya da ölçülmesi yüksek hesaplama ve analiz gücü ile birlikte güçlü donanımsal kaynak gerekliliklerini de yanında getirmektedir

Büyük veri üzerinde metin benzerliklerinin ölçümleri bilgi çıkarım sistemleri, yapay zeka ve NLP gibi geniş bir yelpazede metin sınıflandırma, doküman kümeleme, konu tespiti,

soru cevap sistemleri, makine çevrimi, metin özetleme ve diğer metin işlemleri uygulamalar için araştırmacıların tercih ettiği önemli bir alandır. Bu araştırma alanlarındaki yeni çalışmalar, büyük metin verilerinde güvenilir, doğru ve verimli benzerlik hesaplamaları gerçekleştirmeyi amaçlamaktadır. Büyük doküman setlerinde, kısa metinler ya da cümleler arasındaki benzerliği tespit etmek için kullanılan Şekil 3.2'de görüldüğü üzere çeşitli ölçüm yöntemleri ve yaklaşımları geliştirilmiştir [108–110].



Şekil 3. 2. Benzerlik tespit algoritmalarının dağılımı [111]

Metin benzerliği, farklı iki metin arasında eşleşen ya da benzer olan kelimeleri bulmayı amaçlamaktadır. Metin benzerliği çalışmaları kısa metinler ya da cümleler arasındaki sözdizimsel ve anlamsal benzerlik derecesini tespit etmeyi hedeflemektedir. İki cümle arasındaki sözdizimsel benzerliği anlayabilmek için ortak kelime sayısı veya ortak kelime grubu sayısına bakılırken, anlamsal benzerliği hesaplayabilmek için sözdizimsel bilgi ile birlikte kelimelerin vektör uzayındaki uzaklıklarına bakılmaktadır. Metin benzerliğini kelime benzerliğine indirgeyen farklı yöntemleri birbiriyle kıyaslamaya imkân tanıyan bir yöntem [112] aracılığıyla sunulmuştur. Bu yönteme göre iki metnin benzerliğine bakılırken bir metinde yer alan her kelimenin diğer metinde yer alan tüm kelimelerle olan benzerliğine bakılır. En iyi benzerlik puanı benzer kelime sayısı kıstas alınarak ölçülür.

Büyük metin verilerinin içeriklerine göre benzerliklerinin tespit edilmesi yüksek hesaplama gücü ve bellek alanı gerektirmektedir. Bu sebeple, yapılan doküman benzerlik tespiti çalışmalarında zaman karmaşıklığı ve bellek sorunları ile karşılaşılmaktadır. Literatürde zaman karmaşıklığı, bellek sorunu gibi hataların önüne geçmek için büyük veri teknolojileri kullanılarak başarılı sonuçlar elde edilmiştir [113,114].

Bu çalışmada, popüler MapReduce algoritması üzerine inşa edilen büyük veri teknolojileri kullanarak metin benzerlik ölçümüne odaklanılmıştır. Kosinüs, Jaccard ve n-gram benzerlik algoritmaları olmak üzere üç farklı benzerlik tespit yaklaşımı test edilmiştir ve performanları karşılaştırılmıştır. Hadoop kümesi üzerinde yürütülen testlerde, Kosinüs ve Jaccard algoritmalarının başarılı benzerlik ölçüm performansı sağladığı görülmüştür. Benzerlik ölçümlerini karşılaştırmak için, müşteriler ve aracılar arasındaki konuşmaların metin kayıtları, önceden belirlenmiş selamlama ve kapama cümlelerine uygun olup olmadıklarına göre test edilmiştir. Bu sonuçlardan yola çıkarak, çağrı merkezi analizinde, önceden tanımlı metriklere göre kayıtlı aramaları analiz etmek ve aracının performansını ölçmek için Kosinüs Benzerliği kullanılmasına karar verilmiştir. Müşteri temsilcisi ve müşteriler arasında geçen konuşma metinleri üzerinde performansı değerlendirilen benzerlik ölçüm yaklaşımları aşağıda kısaca açıklanmıştır:

1. *Kosinüs Benzerlik:* İki metin arasındaki benzerliği ölçebilmek için, öncelikle metinlerin sayısal vektör gösterimine dönüştürülmesi gerekmektedir. Her kelime, kelime torbası modelleri ile veri uzayında bir boyuta karşılık gelir ve kelimelerden oluşan metinler bir vektör ile temsil edilir. Vektörler ile temsil edilen metinlerin benzerlik derecesi, ilişkili olduğu vektörler arasındaki açıların kosinüs değerine bakılarak ölçülebilmektedir. $\{\mathrm{i} \in \mathbb{Z} : 1 \leq \mathrm{i} \leq \mathrm{M}\}$ olmak üzere M toplam doküman sayısı olarak olarak düşünüldüğünde a_i ve b_i ile adlandırılan herhangi iki metin vektörü arasındaki kosinüs benzerlik şu şekilde hesaplanmaktadır:

$$benzerlik\ (a,b) = \cos(\theta) = \frac{a.b}{||a||||b||} = \frac{\sum_1^M a_i\, b_i}{\sqrt{\sum_1^M a_i^2}\sqrt{\sum_1^M b_i^2}} \tag{3.3}$$

Kosinüs açı değeri [0-1] arasında bir değer vermektedir. Bu durumda birbirine benzer vektörlerin kosinüs değeri 1'e yakın, aksi durumda yani birbirleri ile ilişkili olmayan vektörlerin arasındaki benzerlik değeri 0'a yakın çıkacaktır.

2. *Jaccard Benzerlik:* Metinler ya da buraki ismi ile dokümanlar arasındaki benzerlik tespiti için dokümanlar terimlere ayrılmaktadır. Her dokümandaki terimlerin ayrı vektörleri tutulur. Jaccard katsayısı olarak da bilinen bu yöntemde, iki doküman arasında hani terimlerin paylaşıldığını ve hangi terimler farklı olduğu karşılaştırılır. Öncelikle her iki dokümanda paylaşılan terimler ve her iki dokümandaki toplam terimler hesaplanır, son olarak bu iki sonuç oranlanarak [0-1] arasında bir değer elde edilir.

$$benzerlik\ (A, B) = \frac{|A \cap \mathrm{B}|}{|A \cup \mathrm{B}|} \tag{3.4}$$

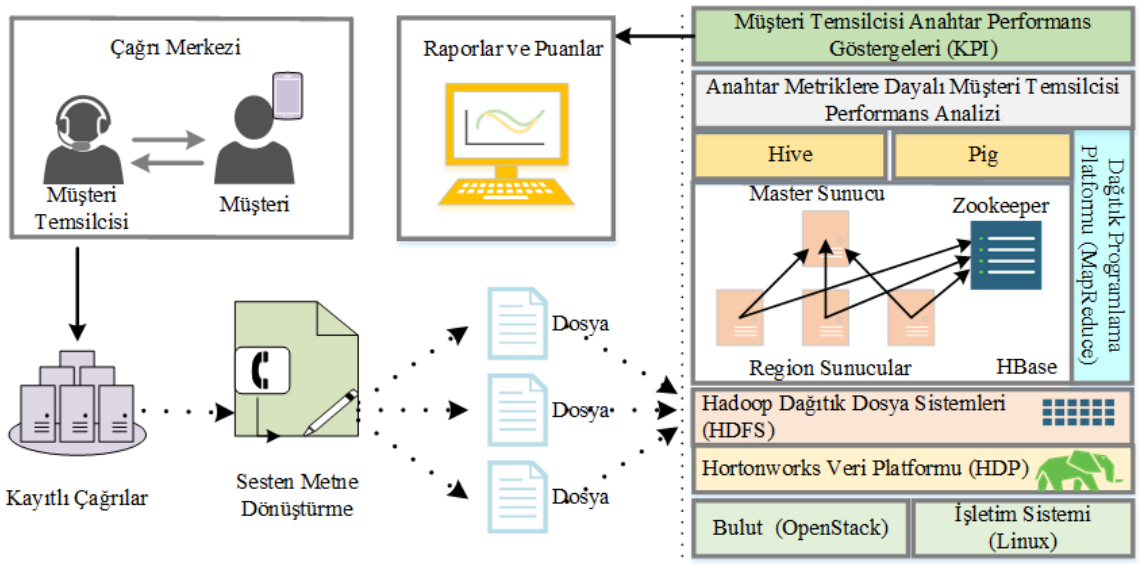
3. *N-gram Benzerlik:* Verilen bir metni n adet harf ya da kelimeden oluşan bileşenlerine ayıran diğer bir benzerlik ölçüm tekniğidir. Örneğin "derin öğrenmeyi seviyorum" cümlesi üzerinde 2-gram, metni iki harf (["de","er", "ri", "in", "öğ", "ğr", "re", "en", "nm", "me",…,"se", "ev", "vi", "iy", "yo", "or", "ru", "um"]) ya da iki kelime (["derin öğrenmeyi", "öğrenmeyi seviyorum"]) bileşenlerine ayıracaktır. N-gram yaklaşımı ile metinler arasındaki benzerlik uzaklığı, benzer n-gram sayısının maksimum n-gram sayısına bölünmesi ile hesaplanmaktadır.

3.2.7. Dağıtık Çağrı Merkezi Analiz Sistemi Tasarımı Ve Geliştirilmesi

Bu çalışmada, çağrı merkezi müşteri temsilcileri ve müşteriler arasında yürütülen tüm kayıtlı telefon görüşmelerini analiz etmek için yeni bir çağrı izleme sistemi önerilmiştir. Önerilen sistemin genel görünümü Şekil 3.3'de verilmiştir. Sistem mimarisi, üç ana bileşenden oluşmaktadır:

- Veri dönüştürme ve depolama sistemi,
- Veri analizi sistemi

- Müşteri hizmetleri temsilcisinin performansının puanlanması ve raporlamasını gerçekleştiren alt sistem



Şekil 3. 3. Çağrı merkezi izleme ve analiz sistemi mimarisi

3.2.7.1. Çağrı merkezi verilerinin dönüştürülmesi ve saklanması

Önerilen sistem ses kayıtlarını metin kayıtlarına dönüştürmek için bir konuşma tanıma uygulamasına ihtiyaç duymaktadır. Google Speech API, IBM Speech to Text, AT&T Speech API, Wit.ai ve diğer teknolojiler gibi birçok konuşma tanıma sistemi bulunmaktadır [115]. Bu deneysel çalışmada, ses verilerini metin verilerine dönüştürmek için Google Speech API kullanılmıştır. Ayrıca önerilen sistemin Google Speech API veya diğer konuşma tanıma teknolojilerine bağlı olmadığı da not edilmelidir. Bu çalışma sadece çağrı merkezi görüşmelerinin analizine odaklanmaktadır.

Çağrı merkezi görüşmelerini saklamak için Hadoop üzerine kurulu bir NoSQL veritabanı olan Apache HBase kullanılmaktadır. Müşteri temsilcileri ve müşteriler arasındaki konuşmaların analiz sonuçları, kullanıcı arayüzünü mümkün olduğunca kolay hale getirmek için HBase tablolarında saklanır ve XML formatında kaydedilir (Şekil 3.4).

Şekil 3. 4. Müşteri temsilcisi performans analiz sonucu XML dosyası formatı

3.2.7.2. Çağrı Verisi Analizi

Çağrı merkezleri, müşteri hizmetleri temsilcisinin performanslarını analiz etmek için genellikle bazı anahtar ölçütler kullanır. Sistem tarafından analiz edilen bazı metrikler aşağıda verilmiştir:

1) Argo kelimelerinin tespiti:

Müşteri temsilcisi, müşteri ile yaptığı konuşma sırasında argo kelimeleri kullanabilir. Müşterinin argo kelimeler kullanması durumunda, müşteri temsilcisi, müşteriyi en az iki kez uyarmalı ve eğer gerekirse iletişimi sonlandırmalıdır. Argo sözcüğünü tespit etmek için öncelikle Türkçe argo kelimelerinin bir listesi oluşturulmuş ve HBase veritabanına atılmıştır. Daha sonra, konuşma metinlerindeki kelimeler argo sözcükleri listesiyle karşılaştırılarak, müşteri temsilcisi veya müşterinin herhangi bir argo sözcüğünü kullanıp kullanmadığı kontrol edilmiştir. Eşleşen argo kelimelerin sayısı veritabanında argo kelime sayısı ve argo puanı olarak kaydedilmiştir.

2) Selamlama ve kapanış cümleleri:

Müşteri temsilcilerinin konuşmalarında dikkate alması gereken açılış ve kapanış cümleleri bulunmaktadır. Bunlar belirlenmiş standart açılış ve kapanış cümleleridir. Müşteri temsilcileri tarafından kullanılan cümleler, önceden belirlenmiş standart cümlelere uygun olmalıdır. Selamlama ve kapanış cümlelerinin uygunluğunu ölçmek için, önceden belirlenmiş cümleler ile konuşma metinlerindeki cümleler arasında Kosinüs Benzerliği hesaplanmıştır. Ayrıca, önerilen yaklaşım, gelen ve giden çağrıları, bu cümlelerle olan benzerliğe göre hesaplayabilmektedir.

3) Müşteri isminin tekrarı:

Müşteri temsilcisi, konuyla ilgili konuşmaya başlamadan önce müşterinin adını sormalı ve sonra tekrarlamamalıdır. Bu çalışmada, öncelikle 7000 isimden oluşan bir Türkçe isimler listesi hazırlanmış ve bu liste HBase veritabanında saklanmıştır. Konuşma metinlerindeki kelimeler isim listesiyle karşılaştırılmış ve müşteri adı tespit edilmiştir. Ardından tekrarlanan isim sayısı kaydedilmiştir.

4) Cümle tekrarı:

Müşteri temsilcisi bir cümleyi, müşterinin tam olarak anlayamadığı durumlarda tekrarlamak zorunda kalabilir. Bu durum, zaman kaybı ve müşteri temsilcisinin performansıyla ilgili bir sorun olarak kabul edilebilir. Bu çalışmada, ilk olarak Zemberek [116] Türkçe NLP Kütüphanesi'ni kullanarak konuşmanın cümlelere bölünmesi gerçekleştirilmiştir. Zemberek noktalama işaretlerini kullanarak cümleleri tespit ettiğinden, cümle tekrarlarını saymak için N-gram Benzerliği kullanılmıştır.

5) Yasaklı kelimelerin tespiti:

Çağrı merkezlerinde, müşteri temsilcilerinin, müşterilerle konuşmalarında kullanmaları istenmeyen kelimeler olabilmektedir. Kelimeleri kullanma ve telaffuz etme şekli ve bunları cümlelere koyma şekli gibi bazı etkenler onları uygun hale getirebilir ancak yine de müşteri temsilcisinin bunları kullanmaktan kaçınması gerekmektedir. Bu yasaklanmış kelimelere bazı örnekler şunlardır: “Ne yazık ki, reddediyoruz, yapamıyorum, imkansız, üzgünüm”. Yasaklı kelimeleri tespit etmek için, Vodafone Çağrı Merkezi ile görüşmeler yapılmış ve elde edilen yasaklı kelimeler “yasakli_kelimeler” tablosunda saklanmıştır. Ardından, konuşmanın herhangi bir yasaklı kelime kullanıp kullanmadığı kontrol edilmiştir.

6) Sinirli müşterilerin uyarılması:

Konuşma sırasında, bazen müşteriler bazı problemler veya yapılan bazı hatalar yüzünden çok agresif olabilir. Bu durumlarda, müşteri temsilcisinin müşteriyi sakinleştirmesi beklenir. Müşteri temsilcisi, müşteriyi sakinleştiremezse, müşteriyi en az iki kez uyarmalıdır. Uyarıdan sonra mümkün olan en iyi uygulama konuşmayı bitirmek olabilir. Aracının öfkeli müşteriyi uyarıp uyarmadığını tespit etmek için, konuşmayı uyarı cümleleriyle karşılaştırarak Kosinüs Benzerliğini ölçeriz. Ardından, müşteri temsilcisinin uyarı puanı hesaplanır ve veritabanında saklanır.

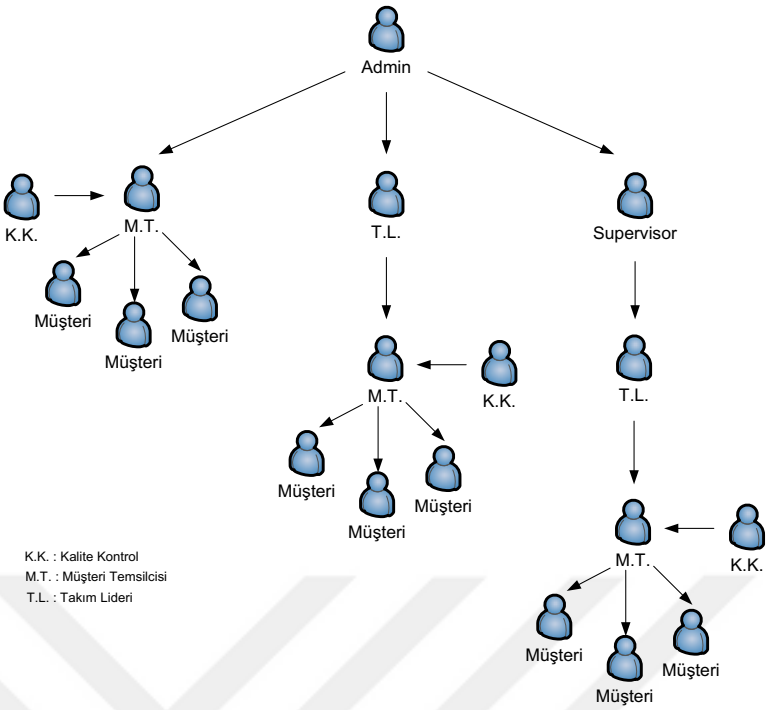
3.2.8. Sistem Arayüzü Tasarımı

Çağrı Merkezlerinin Müşteri Temsilcisi performans değerlendirme sistemini yönetebilmeleri ve değerlendirme sonuçlarını görselleştirebilmeleri için web tabanlı arayüzler hazırlanmıştır. Web arayüzü, örnek olarak alınan bir Çağrı Merkezinin yönetimsel hiyerarşik yapısına uygun bir şekilde yetkilendirme tabanlı olarak çalışmaktadır. Böylece Şekil 3.5'te verilen hiyerarşik yapıda verilen her bir sorumlu personel sadece kendi sorumluluk alanına dahil personellerle ilgili sonuçları görüntüleyebilmektedir. Sistem yöneticisi rolüne sahip personel tüm personellerle ilgili bilgilere ve analiz sonuçlarına erişebilmektedir.

Şekil 3.5'te verilen roller ve sistem arayüzünde sahip oldukları yetkiler aşağıda verilmiştir:

1) Yönetici (Admin) Rolü

Sistem yöneticisi, tüm sistemle ilgili her türlü ekranı görüntüleyebilir ve diğer rollerle ve sistemle alakalı gereken değişiklikleri yapabilir. Diğer rollerle ilgili yetki tanımlaması ve kısıtlaması, yeni görev tanımlaması gibi işlemler gerçekleştirebilir. Admin, Şekil 4'de gösterildiği gibi ***"Takım lideri, Supervisor, Müşteri Temsilcisi"*** rollerindeki kişilerden sorumludur.



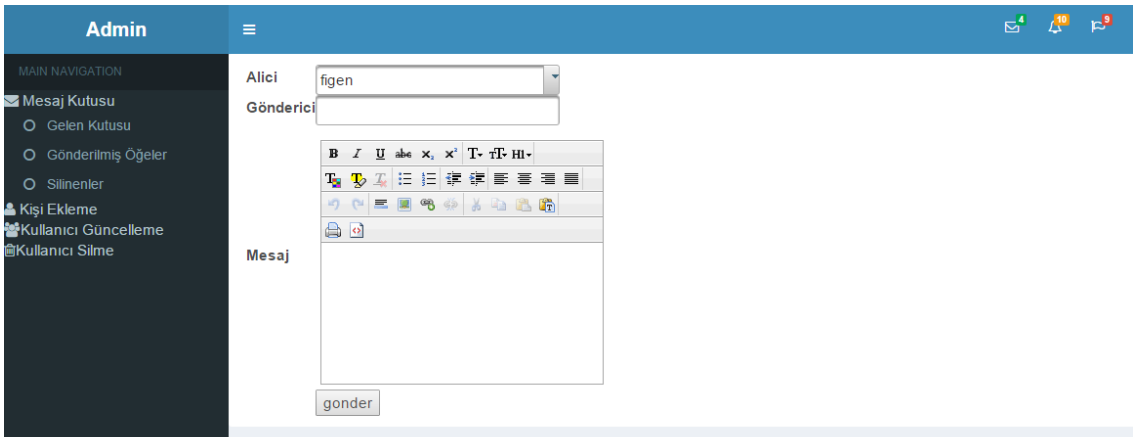
Şekil 3. 5. Çağrı merkezi görev hiyerarşisi

Şekil 3.6'da verilen admin ana sayfasındaki menülerden erişilebilen alt sayfalar ve işlevleri şunlardır:

- Yeni Kişi Ekleme Sayfası: Admin, yeni bir kullanıcı eklemek istediğinde kullanıcı için isim, soy isim, şifre ve rol tanımlar.
- Kullanıcı Silme Sayfası: Sistemde tanımlanmış ancak herhangi bir sebepten dolayı sislinmesi gereken kullanıcıların silinmei işlemi gerçekleştirilir.
- Kullanıcı Güncelleme Sayfası: Admin, mevcut kullanıcılarla ilgili isim, soy isim, şifre ve rol bilgilerini güncelleyebilir.
- Mesaj Kutusu Sayfaları: Admin, sistemde kayıtlı herhangi bir kullanıcıya mesaj gönderebilir ve kendisine gelen mesajları görüntüleyebilir.

2) Supervisor Rolü

Supervisor son kontrolleri yapan, sorumlu olduğu diğer rolleri gözlemleyen, onlar hakkında rapor tutan ve üst merciye bildiren kişidir. Supervisor, Şekil 3.6'da gösterildiği gibi ***"Takım Lideri, Müşteri Temsilcisi"*** rollerindeki kişilerden sorumludur.



Şekil 3. 6. Admin ana sayfası

3) Takım Lideri Rolü

Takım Lideri kendisine bağlı olarak çalışan Müşteri Temsilcilerinin çalışmalarından sorumludur. Her bir takım liderine birden fazla Müşteri Temsilcisi bağlanır ve bunların izin, çalışma süreleri, çalışma performansları gibi konulardan sorumludur.

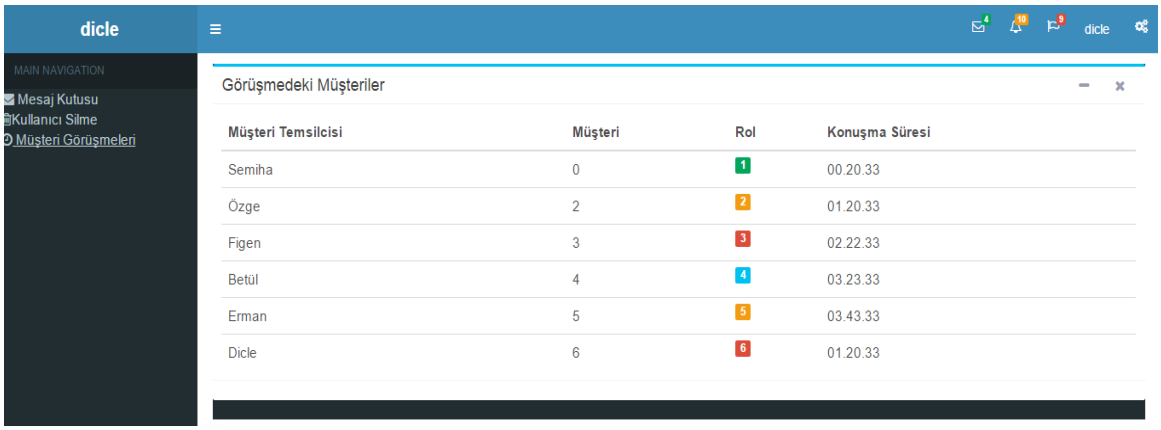
4) Müşteri Temsilcisi Rolü

Müşteri temsilcisi, ilgili çağrı merkezlerini arayan müşterilerin talep ve ihtiyaçlarını karşılamak üzere belirlenen çağrı ve konuşma standartları ya da metrikleri doğrultusunda hizmet veren kişidir. Müşteri temsilcisi rolü için sistem arayüzü Şekil 3.7'de verilmiştir.

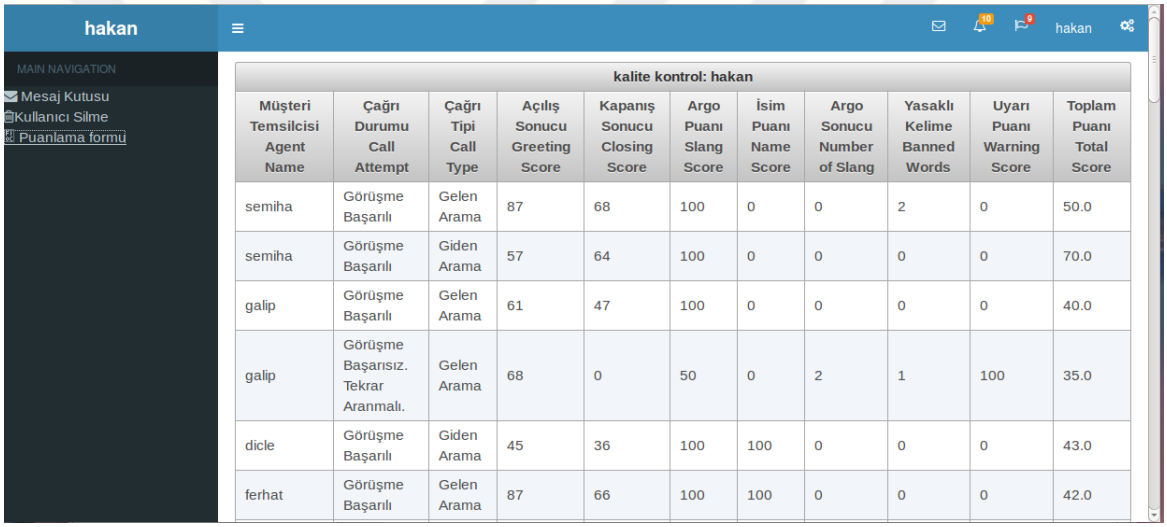
5) Kalite Kontrol Ekranları

Kalite Kontrol rolündeki kişiler, müşteri temsilcisi ile müşteri arasında geçen ve sistemde kaydedilen görüşmeleri analiz ederek müşteri temsilcisine puan verirler. Sistem, önceki bölümde detaylı olarak anlatılan metriklere göre Müşteri temsilcisi görüşmelerini analiz ederek puanlandırma yapar ve verilen bu puanlar sorumlu kişiler tarafından görüntülenir.

Kelite kontrol rolündeki kişiler için tasarlanan Şekil 3.8'deki sistem arayüzünde, çağrı merkezinde çalışan Müşteri Temsilcilerinin gerçekleştirdikleri görüşmelerin her birisi için otomatik olarak hesaplanan performans puanları ve ortalama puan gösterilmiştir.



Şekil 3. 7. Müşteri temsilcisi ekranı



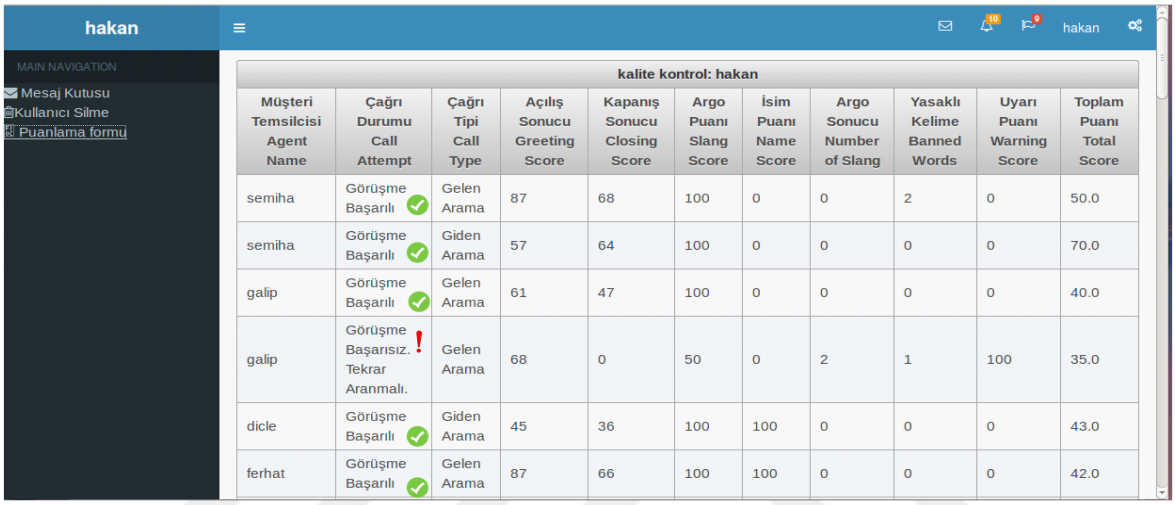
Şekil 3. 8. Müşteri temsilcisi performans analiz ekranı

3.2.9. Müşteri Temsilcisi Performans Değerlendirmesi

Bu çalışma, çağrı kalitesi yönetimi için doğruluk, bütünlük ve güvenilirlik gibi kalite kriterlerini sağlayarak, kaydedilen tüm çağrıları değerlendirmek için dağıtık bir çağrı izleme sistemi sunmaktadır. Yüksek ölçeklenebilirlik elde etmek için, çok sayıda çağrı kaydını analiz etmek için Hadoop MapReduce uygulamasını kullanılmıştır. Açık kaynaklı bir bulut bilişim yazılımı olan OpenStack üzerine kurulu 10 düğümlü bir Hadoop kümesinde testler

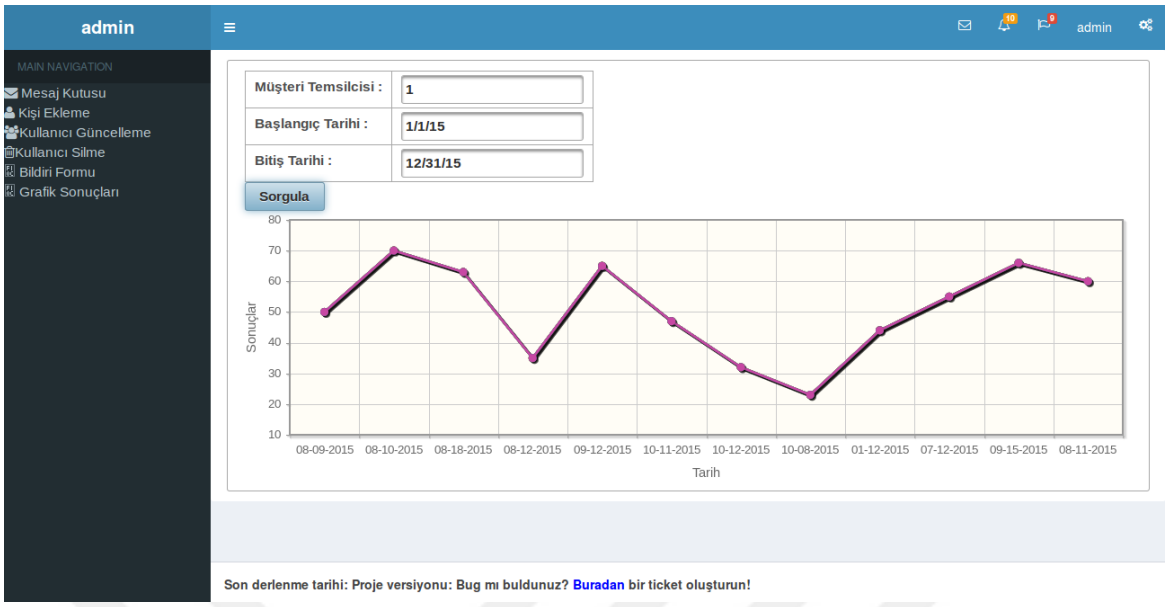
gerçekleştirilmiştir. Her düğüm üzerinde, OpenStack için önerilen işletim sistemi olan Ubuntu 13.04 LTS işletim sistemi kurulmuştur.

Sistem, Müşteri Temsilcilerinin performanslarının günlük, haftalık, aylık veya yıllık olarak değerlendirilebileceği ve grafiksel olarak görselleştirilebileceği arayüzler sunmaktadır. Sorumlu kişi arayüzde Müşteri Temsilcisinin kullanıcı numarası ile analiz yapılmak istenen sürenin başlangıç ve bitiş tarihlerini girerek sistemi çalıştırır.

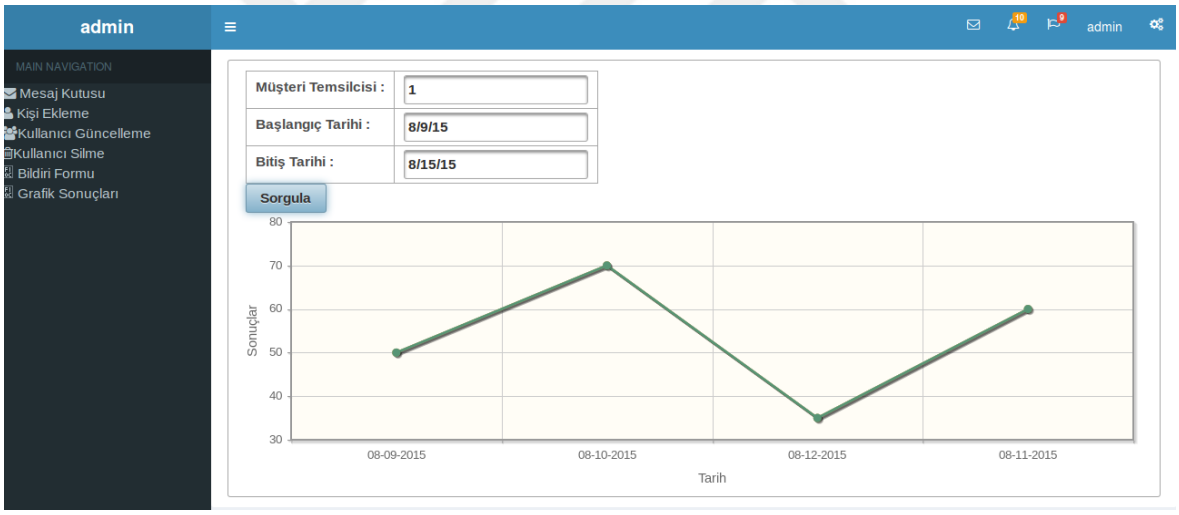


Şekil 3. 9. Müşteri temsilcilerinin çağrı değerlendirme sonuçları için yönetici arayüzü

Şekil 3.9'da, çağrı merkezi yöneticisine sunulan, karşılama cümleleri, kapanma cümleleri, argo kelimeler, yasaklanmış kelimeler, uyarı puanları ve diğerleri gibi her bir anahtar metrik için müşteri temsilcisinin çağrılarını ve performans puanlarını kontrol edip yönetebileceği puanlama arayüzünün görünümünü sunmaktadır. Böylece önerilen sistem, mevcut teknolojilere kıyasla, yöneticiler için erken uyarı sistemi, otomatik müşteri hizmetleri temsilcisi performans skorlaması ve günlük, haftalık (Şekil 3.11), aylık ve yıllık (Şekil 3.10) performans sonuç analizi gibi önemli avantajlar sağlamaktadır.



Şekil 3. 10. Müşteri temsilcisi yıllık performans grafiği



Şekil 3. 11. Müşteri temsilcisi haftalık performans grafiği

Arayüzde gerçekleştirilen sorguların sonuçları sunucu tarafından JSON formatında döndürülmektedir. Kullanıcı arayüzünde ayrıştırılan bu veri kullanılarak grafikler ve tablolar oluşturulmaktadır.

Aşağıda örnek bir JSON çıktısı verilmiştir:


```
{
  "_id": {
    "$oid": "561fac685f6ae10700a97908"
  },
  "cagridurum": "G\u00f6r\u00fc\u015fme Ba\u015far\u0131l\u0131",
  "acilissonuc": 87.5,
  "gid": "gid1",
  "kapanissonuc": 77.151674981046,
  "argopuan": 100,
  "isimpuan": 50,
  "yardimpuan": 100,
  "argosonuc": 0,
  "yasaklikelime": 0,
  "cagritip": "Gelen \u00c7a\u011fr\u0131",
  "cumlepuan": 0,
  "uyaripuan": 0,
  "yardimsonuc": 0,
  "mtid": 100,
  "kapanispuan": 100,
  "cumletekrari": 110,
  "tarih": "08\/10\/2015",
  "puansonuc": 68.75,
  "yasaklipuan": 100,
  "uyarisonuc": 0,
  "acilispuan": 100,
  "isimsonuc": 2
}
```


3.2.10. Sonuç

Bu bölümde çağrı merkezleri için müşteri temsilcisi performansını ölçmeyi amaçlayan dağıtık bir sistem mimarisinin yapısı, bileşenleri ve geliştirme detayları sunulmuştur. Önerilen sistem büyük veri teknolojilerinden MapReduce yaklaşımını kullanan Hadoop, dağıtık dosya sistemi HDFS, HDFS üzerinde depolanan verileri sorgulamak için kullanılan Hive ve dağıtık kaynakları yönetmek için kullanılan Zookeeper teknolojileri kullanılarak geliştirilmiştir. Sistemin dağıtık bir şekilde kurulabilmesi için gerekli olan sanal sunucuları sağlayabilmek için açık kaynak kodlu bulut bilişim yazılımı olan OpenStack tercih edilmiştir. Öncelikle, Hadoop ekosistemi bileşenleri ve geliştirilen MapReduce programlarının yüklü olduğu Ubuntu işletim sisteminin kurulu olduğu bir sunucu oluşturulmuş, daha sonra OpenStack web arayüzü yardımıyla bu sunucudan gerektiği kadar sanal sunucu başlatılarak bir Hadoop kümesi oluşturulmuştur. Oluşturulan bu dağıtık hesaplama kümesi, dağıtık dosya sistemi üzerinde yer alan müşteri temsilcisi çağrılarını okur

ve geliştirilen MapReduce programları paralel olarak çalıştırılır. Böylece müşteri temsilcisi performansını ölçmek amacıyla geliştirilen metin madenciliği ve metin benzerliği kodları eş zamanlı olarak yürütülür. Bu şekilde, müşteri temsilcisi sayısı veya görüşme kayıtlarının sayısının artmasıyla beraber ihtiyaç duyulacak olan ölçekleme yeteneği sisteme kazandırılmış ve ihtiyaç duyulduğu kadar MapReduce görevinin paralel olarak çalıştırılabilmesi sağlanmıştır. Sistem bu yeteneği sayesinde gerçek zamana yakın bir şekilde ihtiyaç duyulduğu kadar veri işleme görevlerini çalıştırabilecektir.

3.3. Dağıtık Okunabilirlik Analizi Sistemi

3.3.1. Giriş ve Motivasyon

Bir dilin kelimeleri, dil gibi canlı varlıklardır ve herhangi bir metni anlamada ve değerlendirmede önemli bir tahlil aracıdır. Bir metnin zorluğu ya da kolaylığı "okunabilirlik" olarak ifade edilmektedir. Okunabilirlik değerlendirmesi; kelimeler, kelimelerle oluşturulan cümleler ve kelimeleri oluşturan heceler gibi gözlemlenebilen dil özelliklerini kullanarak metinleri farklı okuma seviyelerine göre sınıflandırmayı amaçlamaktadır. Metinlerin okunabilirliğinin ölçülmesi amacıyla pek çok okunabilirlik formülü geliştirilmiştir. Günümüzde internet kullanımının artması ile birlikte milyonlarca web sayfası ve ders kitabının elektronik kopyaları hızla arttığı için, bireylerin ve öğrencilerin kendi okuma seviyelerine göre uygun okuma metinlerini seçmesi zor hale gelmiştir. Geliştirilen okunabilirlik formülleri ise internet üzerinde yaşayan milyonlarca dokümanı analiz etmek için teknolojik bir alt yapı sağlamayı zorunlu kılmıştır.

Okunabilirlik çalışmaları genellikle okuma kitabının tamamının analiz edilmesi yerine basit bir paragraf ya da kitabın belirli bölümleri üzerinde geliştirilmiştir. Metin örneklerini kullanan bu çalışmalar, doğru okunabilirlik puanlarını hesaplamak için örnek metni dikkatli bir şekilde seçmek durumundadır. Yani doğru okunabilirlik puanı, seçili örnek metne bağlı olarak değişmektedir. Dahası ülkemizdeki eğitim kurumları tarafından kullanılan herhangi bir okunabilirlik uygulaması yoktur ve tüm Türkçe okuma kitaplarını kısa bir sürede analiz edecek okunabilirlik sistemi geliştirilmemiştir.

Son yıllarda, dijital ortamda saklanan doküman sayısının giderek artması nedeniyle sayısız Türkçe dokümanların incelenmesi ve analiz edilmesi önemli bir çalışma alanı

oluşturmuştur. Bu çalışma alanı, yüksek hesaplama ve analiz gücü ile birlikte güçlü donanımsal kaynak gerekliliğini de beraberinde getirmektedir. Bu çalışmada, Türkçe metinlerin okunabilirliğini ölçen dağıtık alt yapıya sahip bir okunabilirlik analiz sistemi geliştirilmiştir. Önerilen sistemin temel amacı, sisteme yüklenen her büyüklükte kitap, makale vb metinlerin okunabilirlik seviyelerini hızlı bir şekilde hesaplamak ve böylece bireylerin okuma seviyelerine uygun metinleri seçmelerini sağlamaktır.

Okunabilirlik, bir yazının okunmasındaki zorluk derecesi olarak tanımlanabilir. Birçok okunabilirlik testi, yürütme verimliliğini gözönüne almadan, bilimsel kitapların okuma seviyelerini ölçmek için belirli yazılara uygulanmıştır. Bu çalışmada Türkçe ortaokul ders kitaplarının içerdiği metinlerin, popüler MapReduce paradigmasını kullanan, dağıtık ve paralel bir mimari kullanılarak analizi anlatılmaktadır. Hadoop kullanılarak, tam-metin okunabilirlik analizi gerçekleştirmek için oluşturulan Dağıtık Büyük Veri işleme sistemi mimarisi açıklanmıştır. Aynı zamanda ders kitaplarının okunabilirlik dereceleri ve sistem performansı da verilmiştir.

3.3.2. Okunabilirlik

Seçilen bir metnin zorluğu (veya kolaylığı), o metnin okunabilirlik derecesi olarak ifade edilebilir; bu da çeşitli okuma materyallerinin farklı derecelere sınıflandırılmasında kullanılabilir. Öğretmenler, okunabilirlik derecelerini öğrenciler için uygun okuma metinleri bulmak için kullanabilirler. Eğitimin yanında, okunabilirlik uygulamalarının, iş yayınları, karmaşık finansal raporlar [117], online medya, sağlık [118] veya askeri alanlarda hazırlanan başvuru veya teknik manuellerde veya web sayfaları [119] gibi birçok yerde potansiyel kullanım alanları bulunmaktdır.

Metinlerin zorluk derecelerini ölçmek için birçok farklı okunabilirlik formülü önerilmiştir. Flesch Okuma Kolaylığı [120], Flesch-Kincaid Derecesi [121], SMOG Indeksi [122], Gunning Fog İndeksi[123], Otomatik Okunabilirlik İndeksi [124] and Dale-Chall okunabilirlik formülü [125] en çok bilinen okunabilirlik formüllerinden bazılarıdır. Bu popüler formüller geniş bir şekilde sağlık literatürü, iş ve finans yayınları, askeri ve resmi dokümanlar, web içerikleri gibi çok çeşitli metinleri geliştirmek için kullanılmaktadır. Farklı araştırmacılar tarafından 1920'lerden beri geliştirilen okunabilirlik formülleri, verilen bir metnin zorluğunu ölçmeyi amaçlar. Bu çalışmalardan önce metnin okuma zorluğu,

okuduğunu anlamaya dayalı bir metotla belirlenmeye çalışılmaktaydı. Ancak Fletcher [126], okuduğunu anlamanın ölçülmesinin zor olduğunu çünkü bu sürecin doğrudan gözlemlenemez olduğunu belirtmiştir

Okunabilirlik analizi doküman analizinde önemli bir rol oynar. İnternetteki web sayfaları büyük miktarda değerli bilgi barındırmaktadır. Online doküman sayısı inanılmaz hızlarda artmaktadır. Ancak kullanıcılar genelde kendi okuma seviyelerine uygun okuma materyali bulabilecek imkanlara sahip değildir. Diğer taraftan, çok sayıda dokümanı analiz etmek için yüksek hesaplama güçlerine ve yeterli depolama alanlarına ihtiyaç vardır. Örneğin Wikipedia sitesinde 5 TB boyutunda İngilizce doküman bulunmaktadır. Benzer şekilde, ders kitapları veya elektronik kopyalarının analiz edilebilmesi için de güçlü hesaplama kaynaklarına ihtiyaç duyuylur. Dolayısıyla ders kitaplarının tam-metin olarak okunabilirlik analizine tabi tutulabilmesi için bir büyük veri çözümüne ihtiyaç duyulmaktadır.

Büyük veri, çeşitlilik, hacim, hız ve doğruluk gibi çeşitli yönlerden birçok zorluk barındırmaktadır. Çeşitlilik, farklı formatlardaki verilere; hız, verinin ne kadar hızlı üretildiğine ve ne kadar hızlı analiz edilmesi gerektiğine; doğruluk, kritik kararlarda verinin güvenilirliğine işaret etmektedir. Büyük veriyle başa çıkabilmek için bazı dağıtık çözümler mevcuttur. En çok bilineni, Google tarafından yayınlanan MapReduce çerçevesidir [29].

Okunabilirlik analizi ile ilgili araştırmaların çoğu, geliştirilen çözümlerin çalışma zamanı performansına odaklanmıştır. Bu çalışmada, 5, 6, 7 ve 8. Sınıf öğrencileri tarafından kullanılan Türkçe ortaokul kitaplarının okunabilirlik analizini gerçekleştirmek için geliştirilen Dağıtık Okunabilirlik Analizi Sistemi sunulmuştur

3.3.3. İlgili Çalışmalar

Metin okunabilirliği ile ilgili araştırma, 19. Yüzyılın son çeyreğinde, L.A. Sherman tarafından yazılan “Analytics of Literature” [127] adlı eserle başlamıştır. Sherman bu çalışmasında, ortalama cümle uzunluğunun önemine dikkat çekmiş ve daha kısa cümlelerin okunabilirliği arttırdığını göstermiştir.

Takip eden yüzyıldaki okunabilirlik çalışmaları, öğrencilerin yeteneklerine uygun ders kitaplarının seçilmesi için okunabilirlik formüllerinin geliştirilmesine odaklanmıştır. Lively ve Pressey [128], ders kitaplarındaki kelime dağarcığı yükünü ölçümleyip hafifletecek ilk okunabilirlik formülünü geliştirmişlerdir. Bu çalışmada özellikle, geliştirdikleri formül ile

ders kitaplarındaki anlaşılmayan teknik sözcükleri azaltmayı hedeflemişlerdir. Flesch, İngilizce okunabilirlik üzerine yaptığı bir dizi çalışmadan sonra popüler Okuma Kolaylığı formülünü yayınlamış [120] ve 1940-1990 yılları arasında yayınlanan Dale-Chall Formülü [125], SMOG Okunabilirlik Formülü [122], Gunning Zorluk Göstergesi [123] ve Otomatik Okunabilirlik Formülü [124] gibi diğer okunabilirlik formüllerinin öncülüğünü yapmıştır. En çok kullanılan okunabilirlik formülleri Tablo 3.1'de verilmiştir.

Tablo 3. 1. Okunabilirlik formülleri

Okunabilirlik Formülü	**İndeks**	**Formül**	**Sığ Özellikler**
Flesch Okuma Kolaylığı	6	$206.835 - 1.015\left(\frac{kelime\ sayısı}{cümle\ sayısı}\right) - 84.6(\frac{hece\ sayısı}{kelime\ sayısı})$	Ortalama cümle uzunluğu, Hece-tabanlı
Flech-Kincaid Okuma Seviyesi	7	$0.39\left(\frac{kelime\ sayısı}{toplam\ cümle}\right) + 11.8\left(\frac{hece\ sayısı}{kelime\ sayısı}\right) - 15.59$	Ortalama cümle uzunluğu, Hece-tabanlı
SMOG	8	$1.0430\sqrt{uzun\ kelime\ sayısı \times \frac{30}{cümle\ sayısı}} + 3.1291$	Uzun kelime tabanlı (üç heceden fazla)
Gunning Fog	9	$0.4\times\left[\frac{kelime\ sayısı}{cümle\ sayısı} + \left(100\times\frac{uzun\ kelime\ sayısı}{kelime\ sayısı}\right)\right]$	Ortalama cümle uzunluğu, Uzun kelime tabanlı
Otomatikleştirilmiş Okunabilirlik	10	$4.71\left(\frac{harf\ sayısı}{kelime\ sayısı}\right) + 0.5\left(\frac{kelime\ sayısı}{cümle\ sayısı}\right) - 21.43$	Ortalama cümle uzunluğu, Karakter tabanlı
Dale-Chall	11	$0.1579\left(\frac{zor\ kelime\ sayısı\ (listede\ olmayan)}{kelime\ sayısı}\times 100\right) + 0.0496\left(\frac{kelime\ sayısı}{cümle\ sayısı}\right)$	Ortalama cümle uzunluğu, Kelime listesi tabanlı
Ateşman [20]	27	$198.825 - 40.175\left(\frac{hece\ sayısı}{kelime\ sayısı}\right) - 2.610(cümle\ sayısı)$	Ortalama cümle uzunluğu, Hece-tabanlı

Son zamanlarda, okunabilirlik formüllerinin ders kitaplarının okuma seviyelerini ölçmek için yaygın olarak kullanıldığı ve bu formüllerin doğruluğunun sadece İngilizce dilinde değil birçok farklı dilde de değerlendirildiği görülmektedir. Ancak, Flesch Okuma Kolaylığı gibi klasik formüllerin genelde sadece İngilizce dilinde etkili oldukları görülmektedir [129]. Diğer taraftan, birçok dil, ortalama cümle uzunluğu, ortalama hece sayısı, cümle başına düşen ortalama kelime sayısı, ve üç heceden fazla heceye sahip kelimelerin ortalama sayısı gibi genel sözcüksel özellikler açısından birbirine benzerler. Kuo ve arkadaşları [130] Tayvan dilinde okunabilirlik analizi problemini incelemiş ve lise öğrencileri için linguistik özellikler kullanarak kısa makaleleri sınıflandırmışlardır. Analiz sonuçları, Flesch-Kincaid Seviye testi, SMOG ve Otomatik Okunabilirlik İndeksi formüllerinin okunabilirlik seviyelerini ölçmede iyi sonuçlara ulaşamadıklarını göstermiştir.

Polonya dilindeki diğer bir çalışmada [131], metinlerdeki belirli sözcüksel özellikleri kullanarak Gunning-Fog indeksi ve Flesch tabanlı Pisarek metodunu uygulamış ve karşılaştırmışlardır. Tayland dilinde, ilkokul öğrencileri için okunabilirlik seviyelerini ölçmek için Daowadung ve Chan [132] kelime bölütleme ve TF-IDF hesaplamaları içeren bir teknik geliştirmiştir. Benzer şekilde, TF-IDF vektörleri, Çince ilkokul kitaplarının okunabilirlik seviyelerini belirlemek için de kullanılmıştır [133]. Deneysel sonuçlar, alt sınıflar için önerilen yöntemin iyi çalıştığını, orta sınıflar için ise efektif olmadığını göstermiştir.

François ve Fairon [134], Fransızca okunabilirlik formüllerini, ders kitaplarından oluşan bir veri seti üzerinde, sözcüksel, sentaktik ve semantik bir dizi metinsel özelliklerle ilişkili olarak oluşturmuşlardır. Alusio ve arkadaşları [135] da Portekizce metinler üzerinde okunabilirlik değerlendirmelerini metinlerin linguistik yapılarını kullanarak yapmışlardır. Japonca metin okunabilirliği ile ilgili diğer bir çalışmada, 127 ders kitabından alınmış 1478 örnek metin kullanılarak bir okunabilirlik ölçüm yöntemi önerilmiştir [136]. Bu çalışmada basit bir Perl programı kullanılarak ders kitabı okunabilirliği analiz edilmiştir. Yabancı metinlerin okunabilirlik zorluğunu ölçen popüler formüllerden birisi de Lasbarhetsindex İsveç Okunabilirlik Formülüdür (LIX) [137]. Diğer popüler formüller ortalama kelime uzunluğunu hesaplamak için hece sayısını kullanırken, LIX, okunabilirlik analizinde geleneksel özellik olarak kullanılan sığ özellik yerine harf sayısını kullanmıştır. Sjöholm [138], LIX formülünü değerlendirmek için İsveç dilinde metinleri giriş olarak alan bir Java modülü geliştirmiştir.

Her ne kadar klasik okunabilirlik formülleri İngilizce dilinde yazılmış metinler üzerinde yaygın olarak kullanılsa da, Arapça gibi semantik diller için, uygun ders kitaplarını seçmek amacıyla yeni okunabilirlik formüllerinin üretilmesine ihtiyaç duyulmaktadır [139]. Al-Khalifa ve Al-Ajlan [140], ortalama cümle uzunluğu, ortalama kelime uzunluğu, kelime başına düşen ortalama hece sayısı ve kelime frekanslarından oluşan dört özelliği kullanan bir vektörü hesaplayan bir Java programı geliştirmiştir. Çalışmada kullanılan derlem Suudi Arabistan'daki okullarda okutulan kitaplardan alınmış 57089 kelimeden oluşan 150 metinden oluşmaktadır.

Türkçe okunabilirlikle ilgili ilk çalışmalar 1990'larda başlamıştır. Türkçe okunabilirlik seviyesini ölçebilmek için iki popüler formül önerilmiştir: Ateşman [141], Flesch Okuma Kolaylığı formülünün bir adaptasyonu olan (Detaylar için Tablo 1'e bakınız) ilk formülü, Çetinkaya ve Uzun [142] ise üç okunabilirlik seviyesi içeren ikinci okunabilirlik formülünü sunmuştur. İkinci formülde 10, 11 ve 12. sınıflar yüksek seviye, 8 ve 9. sınıflar orta seviye, 5, 6 ve 7. sınıflar giriş seviyesi olarak kabul edilmiştir.

Okur ve Arı [143], Ateşman ve Çetinkaya formüllerini kullanarak, 6-8. sınıflara ait ders kitaplarından seçilmiş 298 metni analiz etmiş ve sonuçları karşılaştırmışlardır. Bu çalışmada, seçilmiş metinler, bilgilendirici metinler ve hikaye edici metinler olmak üzere iki kategoriye bölünmüştür. Analiz sonucunda bilgilendirici metinlerin hikaye edici metinlere göre daha zor olduğunu, bunun da bilgilendirici metinlerdeki ortalama cümle uzunluğunun ve ortalama kelime uzunluğunun daha yüksek olmasından kaynaklandığını belirtmişlerdir. Güven [144] yaptığı çalışmada, yabancılar tarafından kullanılan Türkçe ders kitaplarının okunabilirliğini çalışmıştır.

Tüm bu çalışmalar şu gerçeği göstermektedir ki, çoğu okunabilirlik çalışmaları ders kitaplarının tümünü analiz etmek yerine seçilmiş metinler veya örnek paragraflar üzerinde yapılmıştır. Örnek metinler kullanarak gerçekleştirilen çalışmalarda, okunabilirlik sonuçlarının hatalı hesaplanmasına engel olabilmek için, bu örnek metinlerin dikkatli bir şekilde seçilmesi gerekmektedir. Dahası, tam-metin okunabilirlik analizi yapabilen sistemlerin olmaması ve kapsamlı bir Türkçe ders kitabı derleminin olmaması eğitim kurumlarının kolay bir şekilde okunabilirlik analizi yapabilmesinin önünde bir engel olarak durmaktadır.

Okunabilirlik çalışmaları uzun bir geçmişe sahip olmasına rağmen, ülkemizde Türkçe okunabilirlik ile ilgili yapılan çalışmalar yetersiz kalmaktadır. İngilizce dili üzerine

geliştirilen okunabilirlik formülleri ise, Türkçe metinler üzerinde doğru olmayan sonuçlar vermektedir. Internet kullanımının artması ile birlikte çevrimiçi (online) dokümanların hızla artması, farklı okuma seviyelerindeki bireylerin ve öğrencilerin kendilerine uygun okuma metinlerini bulmaları için geleneksel uygulamalar yetersiz kılmaktadır. "Büyük veri" olarak adlandırılan bu sayısız dokümanların analizi için yüksek hesaplama gücü ve depolama kapasitesi gereklidir.

3.3.4. Okunabilirlik Analizi

Okunabilirlik formüllerinde kullanılan değişkenler, metnin iskeletini gösterir [145]. Geleneksel okunabilirlik formülleri, okuma güçlüğünü etkileyen çok sayıda değişken üzerine kurulmuştur. Aşağıda, en önemli değişkenleri tanımlamamıza yardımcı olacak, ilgili çalışmaların kısa bir özeti verilmiştir.

- Okunabilirlik, Hargis [146] tarafından "kelime ve cümleleri okuma kolaylığı düzeyi" olarak tanımlamıştır. Kelime ve cümle karmaşıklığını tahmin etmek için en yaygın özellikler ortalama cümle uzunluğu, ortalama kelime uzunluğu, kolay ve zor kelime sayıları ve basit cümle sayılarıdır.
- Metindeki farklı kelime sayısı veya sözlük çeşitliliği okuma güçlüğünde önemli bir rol oynar. Dale-Chall [125] (Formül Tablo 1'de verilmiştir.) farklı kelimeleri karşılaştırarak 3000 kelimeden oluşan kolay kelime listesi hazırlamıştır.
- Okunabilirlikle ilgili zorluklar sadece sözcüksel konuları değil aynı zamanda yapısal konuları da içermektedir. Çeşitli formüllerde kullanılan yapısal özellikler edat sayıları, cümlelerin türleri ve ilgeç öbeklerinin sayılarıdır. Ancak bu formüller Kintsch ve Miller [147] tarafından incelenmiş ve kelime ve cümle uzunlukları gibi sözcüksel özelliklerin metin zorluğunu belirlemede, yapısal özelliklere göre daha güçlü ölçümler sağladığı sonucuna varılmıştır.

Bilgisayar yazılımları ile ilgili gelişmeler okunabilirlik analizleri ile ilgili çalışmaları hızlandırmış ve bugün bu konudaki çalışmalar geçmişe göre daha popüler hale gelmiştir. En popüler kelime işlemci yazılımı olan Microsoft Word yazılımı, metnin okunabilirlik seviyesini Flesch Okuma Kolaylığı ve Flesch-Kincaid Okuma Seviyesi formüllerine göre

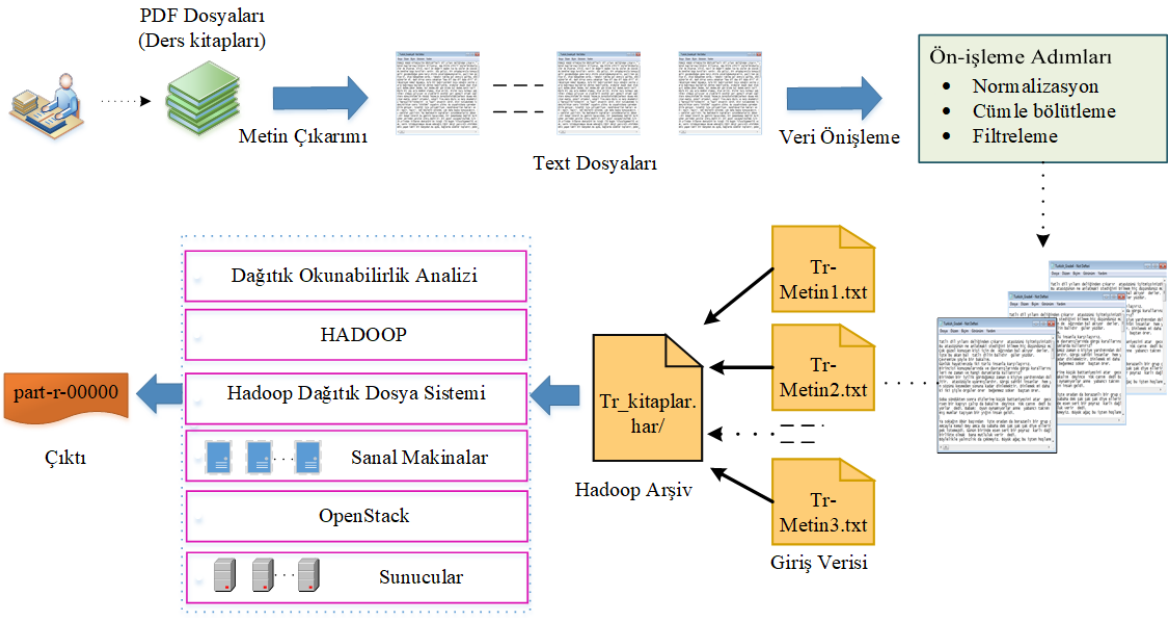
otomatik olarak hesaplayabilmektedir. Okunabilirlik çalışmaları geçtiğimiz yirmi yılda çok önemli değişimler göstermiştir. Örneğin, üç temel özellik (ortalama cümle uzunluğu, ortalama kelime uzunluğu ve ortalama kelime zorluğu seviyesi) üzerine kurulan ATOS okunabilirlik formülü [148] önemli değişimler göstermiştir. 2000 yılında birinci seviye kelime listesi yaklaşık 24.000 kelime içerirken, bu listeye büyük bir güncelleme ile 2013 yılında 75.000 yeni kelime eklenmiştir. Çalışma aynı zamanda geçmişteki okunabilirlik çalışmalarının kitapların tamamını kullanmak yerine küçük miktarda örnek metinler kullanmasından dolayı hatalı sonuçlara ulaştığını vurgulamıştır.

İnternette ulaşılabilen ders kitaplarının elektronik kopyalarının ve web sayfalarının miktarı arttıkça, bireysel okuyucular ve farklı seviyelerdeki öğrenciler için uygun okuma materyalleri bulmak için bu dokümanların geleneksel uygulamalarla analiz edilmesi güçleşmektedir [149]. [150]'de anlatılan Hadoop tabanlı elektronik kitap dönüşüm sistemi, büyük sayıda elektronik kitabın işlenebilmesi için dağıtık bir mimari sunmaktadır. Çalışmada, sistemin tek bir kişisel bilgisayarda çalıştırılması durumunda çok uzun işlem zamanı aldığı belirtilmiştir.

Çok büyük sayıda elektronik dokümanın analizi için temel gereksinimler, yüksek hesaplama gücü ve büyük miktarda depolama kapasitesidir. Büyük veri teknolojileri, paralel ve dağıtık hesaplama yapabilmemize yardımcı olacak şekilde dağıtık depolama ve veri işleme yetenekleri sağlamaktadır. Bu çalışmada, büyük veri teknolojileri kullanarak, ders kitaplarının okunabilirliğinin analizi için kullanılabilecek bir dağıtık sistem sunulmaktadır.

3.3.5. Dağıtık Okunabilirlik Analizi Sistemi

Bu çalışmada, Türkçe ders kitapları için Hadoop tabanlı, Dağıtık Okunabilirlik Analizi Sistemi sunulmuştur. Sistemin genel görünümü Şekil 3.12'de verilmiştir. Sistem mimarisi üç temel alt sistemden oluşmaktadır: Veri dönüşüm sistemi, veri ön işleme sistemi ve veri analiz sistemi. Çalışmanın temel amacı, eğitimciler ve aileler için öğrencilerin okuma seviyelerine uygun okuma materyalleri bulma konusunda kullanılabilecek etkili bir okunabilirlik analizi sistemi sunmaktır. Sistem, Ateşman okunabilirlik formülü tarafından kullanılan iki anahtar özelliği kullanmaktadır: ortalama cümle uzunluğu ve ortalama kelime uzunluğu. Ayrıca, okunabilirlik formüllerinin doğruluğunu ölçmek amacıyla, her bir ders kitabında geçen toplam özgün kelime sayısı da hesaplanmaktadır.



Şekil 3. 12. Dağıtık okunabilirlik analizi sisteminin genel görünümü

3.3.5.1. Ders Kitabı Verilerinin Dönüşümü

Türkiye Cumhuriyeti Milli Eğitim Bakanlığı, Türkçe ders kitaplarını PDF dosyaları olarak sunmaktadır. Sistemde ilk olarak, Hadoop üzerinde çalışan ve metin analizi için kullanılan Apache Tika [151] yazılımı kullanarak PDF dosyalarını metin dosyalarına dönüştürür. PDF dosyalarını metin dosyalarına dönüştürmek için MapReduce görevleri çalıştırılmış ve dosyalar yerel depolama ortamına aktarılmıştır. Din Kültürü kitapları Arapça metinler içermesi dolayısıyla ve sadece resim içeren 12 kitap başarılı olarak dönüştürülememiştir. Veri dönüşümü ile ilgili yürütme zamanı verimi Performans Değerlendirmesi bölümünde Tablo 3.4'te verilmiştir.

3.3.5.2. Ders Kitapları İçin Veri Ön İşleme

Türkçe ders kitaplarının analizinden önce aşağıda verilen ön işleme adımları gerçekleştirilmiştir:

- Normalizasyon: Yazı karakter setlerinin UTF-8 Unicode kodlama standardına dönüştürülmesidir.
- Cümle bölütleme: Doküman içeriğinin cümlelere bölünmesidir. Bu adım için Türkçe Doğal Dil İşleme Kütüphanesi olan Zemberek [116] yazılımı kullanılmıştır.
- Filtreleme: Özel karakterlerin (+,-,*, #, /, \, $, =, & vb.), sayıların ve sık kullanılan kelimelerin (stop words – bir, o, bu, şu, siz vb) ve yazar isimlerinin temizlenmesi.

Literatürdeki birçok Türkçe okunabilirlik analizi çalışması, okunabilirlik değerlendirmelerinde sık kullanılan kelimelerin (stop words) etkisi konusuna dikket etmemiştir. Bu çalışmada biz, ders kitaplarında geçen bu kelimeleri çıkararak bu eksikliği gidermiş bulunmaktayız. Ayrıca, her bir kelimenin terim frekansı hesaplanarak Türkçe sık kullanılan kelime listesi de oluşturulmuştur. Sık kullanılan kelime listesi, kelimelerin terim sıklığına, diğer bir ifadeyle dokümanda kaç defa geçtiklerine göre sıralanması ile bulunmuştur. Sık kullanılan kelimeler, en yüksek terim frekansına sahip kelimelerdir ve diğer kelimelere göre çok daha az önemlidir. Bundan dolayı en sık kullanılan kelimeler metinden çıkarılmıştır.

3.3.5.3. Ders Kitapları Verilerinin Analizi

Önişlemeden geçirilmiş ders kitapları verileri, Hadoop Dağıtık Dosya Sistemi (HDFS) üzerinde depolanarak, Hadoop tarafından analiz edilmek üzere giriş olarak kullanılır. Hadoop [30], büyük veri analizinde en yaygın olarak kullanılan sistemdir. Geleneksel okunabilirlik yöntemleri, tipik olarak tek bir bilgisayarda çalışan ve ders kitaplarından seçilen örnek kontrollü metinleri analiz eden sistemlerdir. Ancak tam metin ders kitapları, küçük hesaplama kaynakları ile işlenebilmek için büyük ve genelde daha gürültülüdür. Bu problemi aşabilmek için, İlkokul ders kitaplarının okunabilirlik analizinde Hadoop platformu kullanılmıştır.

1) Map Aşaması:

Map görevi HDFS üzerinde depolanan ve ders kitapları içeriklerinden oluşan her bir giriş verisi bloğunu ayrıştırır. İlk map görevi cümle, kelime ve hece sayılarını bulur (Tablo II) ve <dosya_ismi, ders kitabının içeriği> şeklinde bir anahtar-değer çifti oluşturur. İkinci

map görevi, her bir kelimeyi ayrıştırır ve tüm kitapta bu kelimenin kaç defa geçtiğini sayar. Anahtarlar, ders kitabı dosyalarındaki özgün kelimelerden (Tablo II) ve değerler ise her bir kelime için verilen sayısal değerleri içerir. Map aşamasında, özel karakterler içeren kelimeler, rakamlar ve sık geçen kelimeler çıkarılır, böylece veriler filtrelenmiş olur ve özgün kelimelerin kaç defa geçtiğinin belirlenmesi aşamasında bunların kullanılmasına engel olunur.

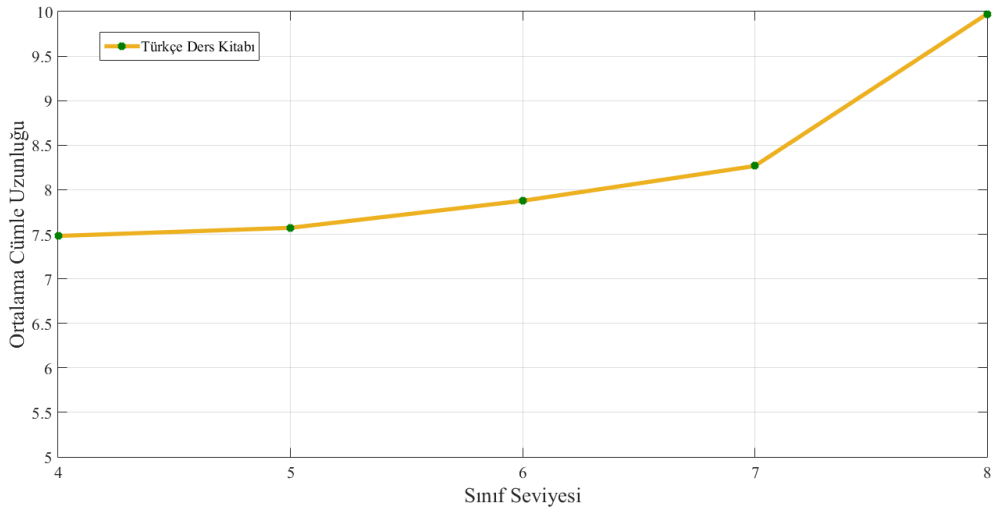
2) Reduce aşaması:

Reduce işlemcisi, map aşamasından gelen sonuçları giriş olarak alır ve cümle, kelime ve hecelere verilen rakamsal değerleri toplar. Daha sonra toplam cümle sayısını toplam kelime sayısına bölerek ortalama cümle uzunluğunu, toplam kelime sayısını toplam hece sayısına bölerek de ortalama kelime uzunluğunu hesaplar. Sonraki adımda, Reduce görevi Ateşman okunabilirlik değerini hesaplayarak <dosya_ismi, ders kitabı içeriği> anahtar-değer çiftlerini çıkışa yazdırır. İkinci reduce görevi, her bir kitapta geçen özgün kelimelerin geçiş sayılarını toplar ve çıkışa dosya isminin anahtar, her bir kitaptaki toplam özgün kelime sayısının değer olduğu <dosya_ismi, n kelime listesi> şeklinde bir anahtar-değer çifti yazar.

Tablo 3. 2. Türkçe ders kitapları için hesaplanan özellikler

Ders Kitabı Sınıfı	Kelime Sayısı	Cümle Sayısı	Hece Sayısı	Özgün Kelime Sayısı
4. Sınıf	14917	1994	38396	4243
5. Sınıf	17484	2309	45293	5095
6. Sınıf	18232	2315	47726	5195
7. Sınıf	20517	2482	52692	5835
8. Sınıf	16895	1694	54680	5798

Ortalama cümle uzunluğu ve okunabilirlik puan sonuç grafikleri sırasıyla Şekil 3.13 ve Şekil 3.14'te verilmiştir. Grafikler, ders kitaplarının ait oldukları sınıflara göre ortalama cümle uzunluğu ve ortalama kelime uzunluklarının okunabilirlik seviyeleriyle uyumlu bir şekilde artış gösterdiğini ortaya koymaktadır. Bu sonuçlara göre 4.,5. sınıflara ve 6., 7. sınıflara ait kitapların okunabilirlik skorları 60 ile 80 arasında, 8. sınıf ders kitaplarının okunabilirlik skoru 40 civarında hesaplanmıştır.



Şekil 3. 13. Sınıflara göre ortalama cümle uzunluğu

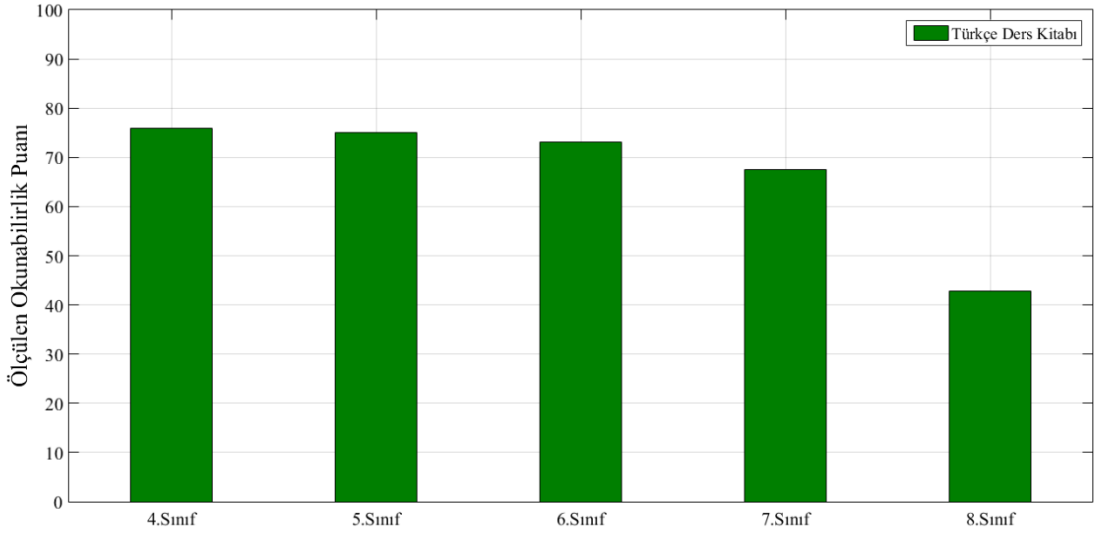
3.3.6. Performans Değerlendirmesi

Bu çalışmada anlatılan Türkçe ilköğretim kitaplarının okunabilirlik değerlerini hesaplayabilmek için iki farklı uygulama geliştirilmiştir:

- İlk uygulama, Java dilinde, geleneksel programlama yaklaşımıyla yazılan bir konsol uygulamasıdır. Bu uygulama 4 çekirdekli ve 4.1 GHz Intel işlemciye ve 16 GB ana hafızaya sahip, Ubuntu 14.04 LTS işletim sistemi kullanan bir bilgisayarda çalıştırılmıştır.
- İkinci uygulama dağıtık Hadoop tabanlı bir sistem kullanmaktadır. OpenStack bulut bilişim yazılımı kullanılarak 10 düğümden oluşan bir Dağıtık Hadoop kümesi mimarisi oluşturulmuştur.

Tablo 3. 3. Performans karşılaştırması

Ders Kitabı Sayısı	Konsol Uygulaması (sn)	MapReduce Uygulaması (sn)
1	1.5	2
2	6.5	3
5	141.692	5
10	Out of memory	10
100	Out of memory	57



Şekil 3. 14. İlköğretim kitapları için okunabilirlik dereceleri.

İki uygulamanın performans karşılaştırması Tablo 3.3'te verilmiştir. Performans karşılaştırması, kullanılan ders kitabı sayısının ikinin üzerinde olması durumunda, Hadoop kümesi üzerinde çalışan MapReduce uygulamasının ortalama yürütme zamanının, ders kitabı sayısı arttıkça düştüğünü ve konsol uygulamasına göre daha iyi performans sunduğunu göstermektedir.

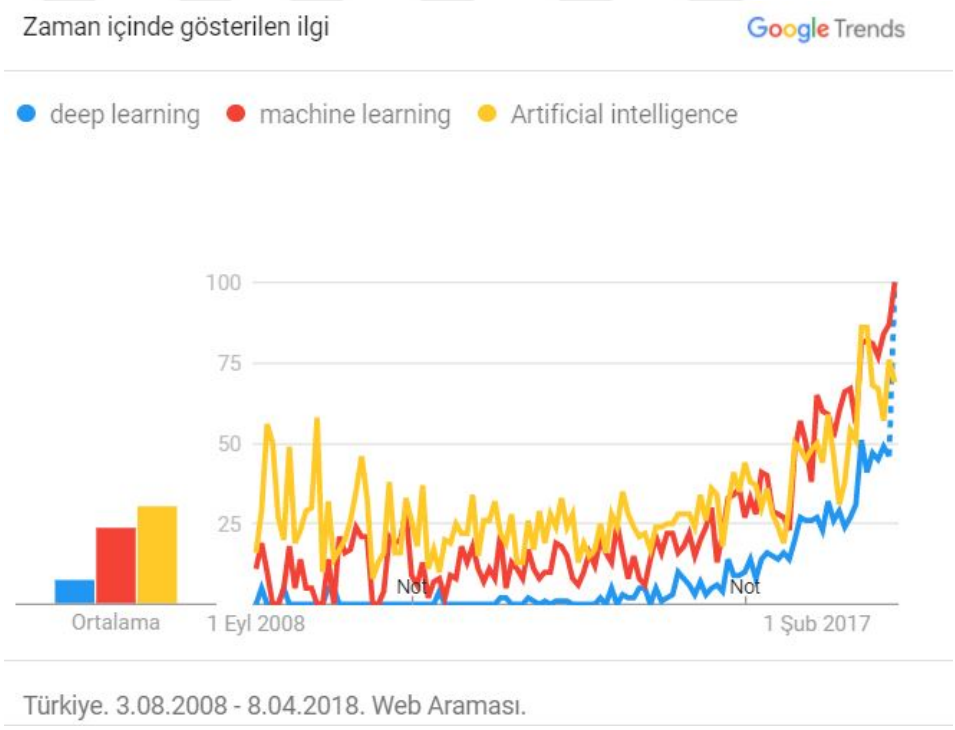
Tablo 3. 4. Hadoop platformunda ders kitabı dönüştürme performansı

Ders Kitabı Sayısı	Dosya Boyutu (GB)	Çalışma Zamanı (Dakika)
10	1.3	4.25
20	2.5	6.26
50	5.6	8.78
100	11.2	14.23

4.YAPAY SİNİR AĞLARI VE DERİN ÖĞRENME

4.1. Yapay Zeka Tarihi

"Yapay Zeka" ismini ilk kez 1956 yılında bu alanın babası olarak bilinen John McCarthy akademik bir konferansta duyurdu. McCarthy, yapay zekanın insan seviyesine ulaşmasının ne kadar süreceğini tahmin edemediğini söylese de, makineler insanlar için çözülmesi zor problemleri çözmeye başlamıştı bile. Örneğin ikinci dünya savaşında Alman askeri iletişimi için kullanılan Enigma kodlarını çözümlemek için, Alan Turing bir yapay zeka sistemi geliştirdi. Enigma kodunun bir insan tarafından kırılması haftalar ve belki aylarca sürebilecekken, geliştirilen yapay zeka makinesi ile saatler içinde kodun çözülmesi başarıldı [152].



Şekil 4. 1. Son 10 yıl içerisinde yapay zeka, makine öğrenmesi ve derin öğrenme

Bilgisayarlar, insanların çözmesinin zaman alacağı ya da çözmesinin zor olduğu karmaşık hesaplama gerektiren sinyal işleme, işletim sistemi yönetimi gibi bazı problemlerde başarılıydı. Fakat insanların rahatlıkla sınıflandırabildiği kedi ve köpekleri ayırt edebilme (nesne tanıma), sadece yüzlere bakarak üzgün ya mutlu olduğunu görebilme, bir metnin ya da söylemin duygusunu ve fikrini anlayabilme (duygu analizi), duyduğunu metne aktarabilme (konuşma tanıma) gibi sezgisel problemlerin çözümünde insan seviyesine ulaşması çok zor olmaktadır. Yapay zeka, ilk yıllarında belirlenmiş kurallara göre çalışan ve yeni koşullara adapte olamayan (hard coding) yöntemler geliştirildiği için mantıksal problemlerin çözümünde iyi olsa da, yüz tanıma, nesne tespiti, konuşma tanıma ve doğal dil anlama gibi karmaşık problemlerde yetersiz kalmıştır. Bu nedenle makine öğrenimi ve derin öğrenme gibi yeni yapay zeka yaklaşımları geliştirilmiştir.

Makine öğrenmesi yapay zekanın en temel yaklaşımı olmasıyla birlikte, derin öğrenme yapay zekada yeni bir dönem açan bir araştırma alanı olmuştur. Google Trend görselinden alınan Şekil 4.1'deki resme bakıldığında, derin öğrenmenin çok sıcak bir araştırma alanı olması rağmen, büyüme oranının makine öğrenmesinin büyüme oranından daha hızlı olduğu gözlemlenebilmektedir. Son yıllarda ise derin öğrenme ile birlikte yapay zekanın da popülerlik kazandığı ve derin öğrenmenin makine öğrenmesi seviyesine ulaşmaya çalıştığı görülmektedir.

4.2. Yapay Sinir Ağlarının Temelleri

Uzun bir geçmişi olan sinir ağları üzerine yapılan ilk çalışmalar insan beyninin nasıl çalıştığına odaklanmıştır. Daha sonraları sinirbilimciler (neuroscientists), zekanın nasıl oluştuğuna dair soruların cevaplarını arayan tartışmalara girmişlerdir. Sinirbilimin öncüsü olduğu düşünülen Santiago Ramon y Cajal, Golgi yöntemi olarak adlandırılan renklendirme tekniğini kullanarak sinir sisteminin temel birimi olan nöronu keşfetmiştir [153]. İtalyan bilimci Camillo Golgi (1843-1926) ile birlikte İspanyol bilimci Santiago Ramon y Cajal (1852-1934), sinir sisteminin yapısı üzerine yaptığı çalışmalar için 1906 yılında nobel ödülüne layık görülmüştür. Sinir sistemi 100 milyardan fazla nörondan oluşmaktadır. Nöron üç kısımdan oluşmaktadır: Dentritler, hücre gövdesi ve akson [154]. Nöronlar birbirine sinaps adı verilen

yapısal ve fonksiyonel bağlantılar aracılığıyla bağlanmıştır. Yapay bir nöron da biyolojik nörona benzer yapıdadır. Hücre gövdesi ya da çekirdek işleme birimine, dentritler girdilere ve akson da çıktıya benzemektedir. Nöronlar arasındaki bağlantılar da sinapslara benzer bir şekilde yapay sinir ağını oluşturmaktadır. Cajal, nöronlar arasındaki sinaptik bağlantıların değişmez olmadığını ve öğrenme yolu ile modifiye edilebileceğini önermiştir. Sinaps sayısındaki artışın, öğrenme ve hafıza mekanizmalarından birisi olabileceği yönündeki önerisi, daha sonraki çalışmalarda kanıtlanmıştır [155].

4.2.1 Algılayıcılar

Warren McCulloch ve Walter Pitts [156], nöronlardan oluşan bir ağın her hangi bir mantıksal önermeyi hesaplayabileceğini kanıtlayarak bilgisayar tasarımı, sinir ağları ve yapay zeka gibi alanların gelişimini sağlamıştır. Beynin ilk modern hesaplama teorisini sundukları çalışma, sinir ağı araştırmalarının başlangıç noktası olarak referans edilmektedir. Yapay nöronun ilk modeli olan McCulloch-Pitts model, aynı zamanda doğrusal eşik kapısı olarak adlandırılmaktadır. McCulloch-Pitts modelinin basit ve kesin bir matematiksel tanımı vardır. Fakat modelin ağırlık ve eşik değerleri sabit olduğu ve yalnızca ikili çıktı ürettiği için farklı özellikli çeşitli uygulamalara uygulanabilirliği zordur. Bu nedenle, daha esnek hesaplama özellikleri olan sinir modellerine ihtiyaç duyulmuştur.

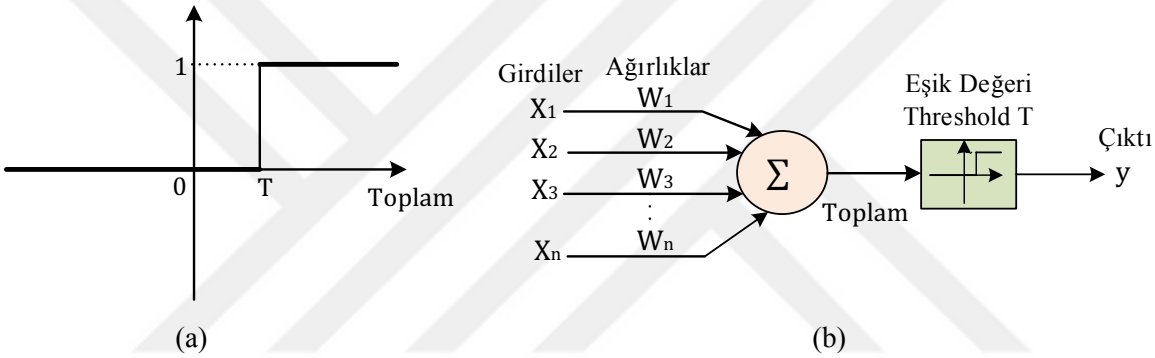
1957 yılında Frank Rosenblatt [157], algılayıcı (perceptron) olarak adlandırılan sinir ağlarının ilk yayınlanan algoritmasını keşfetmiştir. Sinir ağlarının en basit yapısına sahip olan algılayıcı, verilen bir girdi üzerinde doğrusal denklem hesaplamaları gerçekleştirerek 0 ya da 1 gibi iki sınıf üretmektedir. Daha genel anlamıyla tek katmanlı sinir ağı olarak adlandırılan algılayıcı, doğrusal bir sınıflandırma gerçekleştiren en basit sinir ağı modelidir. McCulloch-Pitts modeline ek olarak algılayıcılar, değişken ağırlık değerleri ve bias ismi verilen ek bir girdi değeri almaktadır.

Bir nöron, $X_1, X_2, X_3, \dots, X_n$ girdilerinden, her bir girdinin $W_1, W_2, W_3, \dots, W_n$ ağırlık değerlerinden ve y çıktısından oluşmaktadır. Doğrusal eşik kapısı, girdi kümesini iki farklı

sınıfa sınıflandırdığı için çıktı ikilidir (binary). Matematiksel olarak bu fonksiyon şu şekilde ifade edilir:

$$\text{Toplam} = \sum_{i=1}^{n} X_i W_i \ \ , y = f(Toplam) \tag{4.1}$$

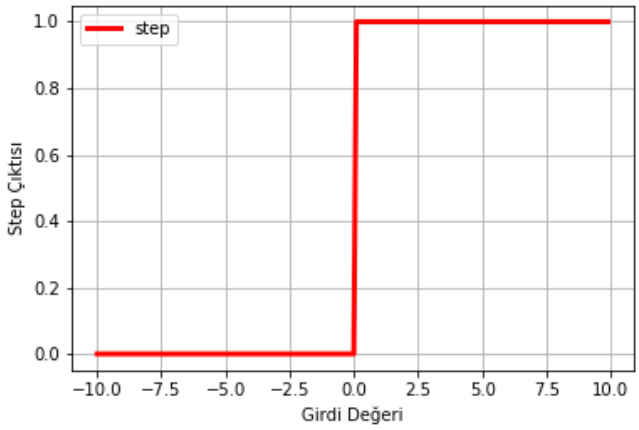
$Toplam,$ ağırlıklı toplamı ve T, sabit bir eşik değeri ifade etmektedir. Bir f fonksiyonunun, T eşik değerindeki doğrusal fonksiyonu ve algılayıcı nöronunun sembolik gösterimi sırasıyla Şekil 4.2(a) ve Şekil 4.2(b)' de gösterilmiştir.



Şekil 4. 2. Algılayıcı modeli (a) Doğrusal eşik fonksiyonu , (b) Algılayıcı nöronu

Doğrusal eşik fonksiyonu olarak geçen step fonksiyonunun $x_1, x_2, x_3, \ldots x_n$ girdilerinin ve $\hat{y}$ tahmini çıktının olduğu durumda $\hat{y} = f(x)$ şartının 4.3'teki sonuç grafiği ve denklemi aşağıda verilmiştir:

$$\hat{y} = \begin{cases} 1 \ eğer \ wx + b \geq 0 \\ 0 \ eğer \ wx + b < 0 \end{cases} \tag{4.2}$$



Şekil 4. 3. Standart step fonksiyonu

Algılayıcı algoritmasının sözde kod adımları aşağıda özetlenmiştir:

Algoritma 1: Algılayıcı

1. Rastgele ağırlık değerleri ile başla.
2. Her bir yanlış sınıflandırılmış $(x_1, x_2 \ldots x_n)$ noktası, y gerçek çıktı ve $\hat{y}$ tahmini çıktısı için $\propto$ öğrenme katsayısı ile w ağırlık ve b bias değerlerini güncelle;
   - Pozitif durum $y - \hat{y} = 1$:
   - For i=1…n:
   - İf $\hat{y} = 0$:
   - $w_i := w_i + \alpha x_i$
   - $b := b + \alpha$
   - Negatif durum $y - \hat{y} = -1$:
   - For i=1…n:

     İf $\hat{y} = 1$:
   - $w_i := w_i - \alpha x_i$
   - $b := b - \alpha$
3. Yukarıdaki adımları t zaman süresince tekrarla.

4.2.2 Lojistik Regresyon

Bir sinir ağı kabul edilen bu modelde, algılayıcılardan farklı olarak aktivasyon fonksiyonu için kullanılan eşik fonksiyonu yerine sigmoid fonksiyonu kullanılmaktadır. Sınıflandırma problemleri için yaygın olarak tercih edilen lojistik regresyon, olasılıksal ve doğrusal bir sınıflandırıcıdır. Girdi özelliklerini tutan bir vektörün, herhangi bir sınıfa ait olma olasılığı aşağıdaki denklemle ifade edilebilir:

$$P(Y = i|x, w, b) = \sigma(z)$$
$$z = w \times x + b \tag{4.3}$$

Denklem 4.3'te Y değeri sınıflandırıcının çıktısını, i değeri sınıflardan birisini, x, w, b değerleri ise sırasıyla girdileri, ağırlıkları ve bias değerlerini temsil etmektedir. z, ağırlık ve girdi değerlerinin çarpımına bias değerinin eklenmesi ile oluşan regresyon eşitliğini gösterir. w ve b değerleri verildiğinde x girdisinin i sınıfına ait olma olasılığı olan $\sigma(z)$ değeri ise model fonksiyonu yani sigmoid fonksiyondur.

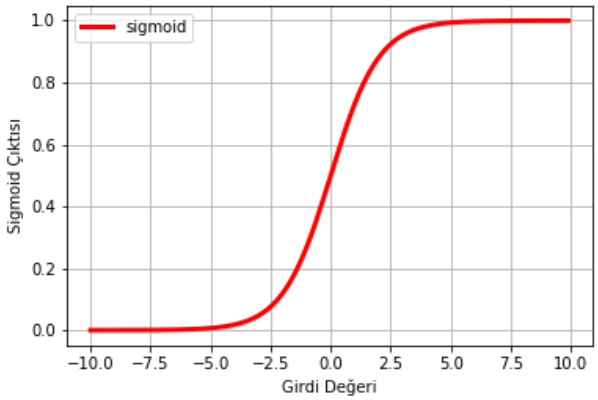
Lojistik regresyon modelde ikili ve çoklu sınıflandırma gerçekleştirmek için gereken adımlar aşağıda verilmiştir:

- Verilerin hazırlanması, girdilerin, parametrelerin ve diğer değişkenlerin tanımlanması
- Modelin tanımlanması
- Hata fonksiyonun tanımlanması
- Belirli bir devir sayısında modelin eğitilerek, hata fonksiyonlarının hesaplanması, hatanın en aza indirgenmesi ve daha iyi bir model elde edilmesi

İkili sınıflandırma için $\sigma(z)$ sigmoid fonksiyon Şekil 4.4'te verilen grafiği ve denklemi aşağıda verilmiştir:

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(w \times x + b)}} \tag{4.4}$$

Sigmoid fonksiyon, [0-1] aralığında bir y değeri üretmektedir ve $y > 0.5$ ise sınıf 1 değerine aksi durumda 0 değerine eşit olan $\hat{y} = \sigma(z)$ eşitliği kurulmaktadır. İkili lojistik sınıflandırma için hata (loss) fonksiyonu $E(w, b)$ (bazı yerlerde $J(w)$ olarak geçmektedir) ise matematiksel olarak aşağıdaki gibi yazılmaktadır:



Şekil 4. 4. Standart sigmoid fonksiyonu

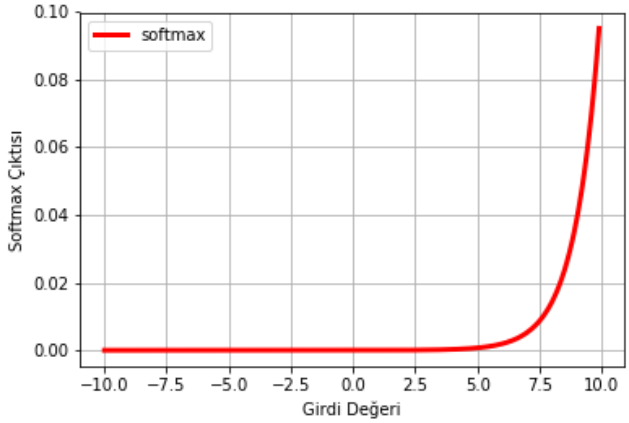
$$E(w, b) = -\frac{1}{m}\sum_{i=1}^{m}(y_i \times \log(\hat{y}) + ((1 - y_i) \times \log(1 - \hat{y})) \quad (4.5)$$

Çoklu sınıflandırma yani ikiden fazla sınıfın olduğu durumda, sigmoid fonksiyon yerine genellikle en popüler aktivasyon fonksiyonu olan softmax fonksiyon kullanılmaktadır. Softmax matematiksel olarak Denklem 4.6'da ve Şekil 4.5'teki grafikte sunulmuştur:

$$softmax\ \sigma(z) = \frac{e_i^z}{\sum_j e_j^z} = \frac{e_i^{(w \times x + b)}}{\sum_j e_j^{(w \times x + b)}} \quad (4.6)$$

Softmax fonksiyonu her sınıf için tahmin üretir, en yüksek softmax değerine sahip sınıf, tahmini sınıf ya da çıktı olur. Çoklu lojistik sınıflandırma için hata fonksiyonu $E(w, b)$ ise matematiksel olarak aşağıdaki gibi yazılmaktadır:

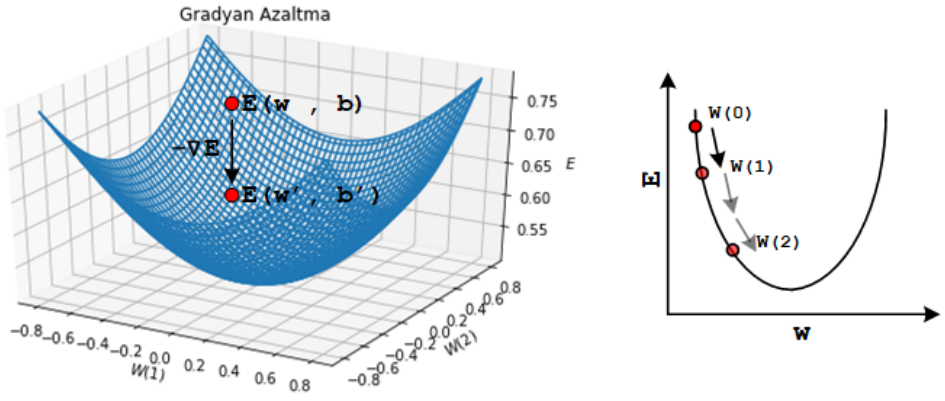
$$E(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij} \times \log\left(\hat{y}_{i,j}\right) \ , \hat{y} = \sigma(z) \tag{4.7}$$



Şekil 4. 5. Standart softmax fonksiyonu

4.2.3. Gradyan Azaltma

Gradyan azaltma/iniş (gradient descent), sinir ağları için kullanılan bir optimizasyon algoritmasıdır ve istenilen bir noktadaki parametrelere göre E hata fonksiyonunun türevini alarak hesaplanmaktadır. Hesaplama sonucu negatif gradyan yönünde parametreler güncellenmektedir.



Şekil 4. 6. E (hata fonksiyonu) üzerinde gradyan azaltma güncellemelerinin gösterimi

Şekil 4.6'da E hata fonksiyonunu, w_1 ve w_2 değerleri hata fonksiyonunun giriş ağırlıklarını göstermektedir. Hata fonksiyonunu en aza indirgemek için, w_1 ve w_2 değerlerine göre E değerinin parçalı türevini yani hata fonksiyonun gradyanını hesaplamak gereklidir.

Hata fonksiyonunun türevini alma işlemi için bazı matematiksel formüller aşağıda verilmiştir:

1. $\sigma(z)$ sigmoid fonksiyonunun türevi olan $\sigma'(z)$ şu şekilde hesaplanmaktadır:

$$\sigma'(z) = \frac{\partial}{\partial z}\frac{1}{1+e^{-z}} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \times \frac{e^{-z}}{1+e^{-z}} = \sigma(z)\big(1-\sigma(z)\big) \tag{4.8}$$

2. Hata değerini en aza indirgemek için, verilen bir $x = [x_1, x_2 \dots x_m]$ noktasında, tahmin edilen $\hat{y}_i = \sigma(Wx_i + b)$ eşitliğinin kurulduğu yerde E hata fonksiyonunun gradyanı yani parçalı türevi alınmalıdır. İşaretli m noktadaki hata formülü daha önceden Denklem 4.5'te verilmişti. Basit olarak her bir noktada üretilen hata ise şu şekilde verilmektedir:

$$E = -\mathrm{y}\log(\hat{\mathrm{y}}) - (1-\mathrm{y})\log(1-\hat{\mathrm{y}}) \tag{4.9}$$

Ağırlıklara göre bu hatanın türevini hesaplamak için ilk olarak $\frac{\partial \hat{y}}{\partial w_j}$ değeri hesaplanır:

$$\begin{aligned}
&\frac{\partial \hat{\mathrm{y}}}{\partial w_j} = \frac{\partial}{\partial w_j}\sigma(Wx+b) \\
&= \sigma(Wx+b)(1-\sigma(Wx+b)) \times \frac{\partial}{\partial w_j}(Wx+b) \\
&= \hat{\mathrm{y}}(1-\hat{\mathrm{y}}) \times \frac{\partial}{\partial w_j}(Wx+b) \\
&= \hat{\mathrm{y}}(1-\hat{\mathrm{y}}) \times \frac{\partial}{\partial w_j}(w_1x_1 + \cdots + w_jx_j + \cdots + b) \\
&= \hat{\mathrm{y}}(1-\hat{\mathrm{y}}) \times x_j
\end{aligned} \tag{4.10}$$

3. Son olarak, E hatasının w_j ağırlıklarına göre verilen bir x noktasındaki türevi aşağıdaki gibi hesaplanmaktadır:

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j}[-\mathrm{y}\log(\hat{\mathrm{y}}) - (1-\mathrm{y})\log(1-\hat{\mathrm{y}})]$$

$$= -y\frac{\partial}{\partial w_j}\log(\hat{\mathrm{y}}) - (1-\mathrm{y})\frac{\partial}{\partial w_j}\log(1-\hat{\mathrm{y}})$$

$$= -y\times\frac{1}{\hat{\mathrm{y}}}\times\frac{\partial}{\partial w_j}\hat{\mathrm{y}} - (1-\mathrm{y})\times\frac{1}{1-\hat{\mathrm{y}}}\times\frac{\partial}{\partial w_j}(1-\hat{\mathrm{y}})$$

Hatırlatma: $\frac{\partial}{\partial w_j}\hat{\mathrm{y}} = \hat{\mathrm{y}}(1-\hat{\mathrm{y}})x_j$

$$= -y\times\frac{1}{\hat{\mathrm{y}}}\times\hat{\mathrm{y}}(1-\hat{\mathrm{y}})x_j - (1-\mathrm{y})\times\frac{1}{1-\hat{\mathrm{y}}}\times(-1)\hat{\mathrm{y}}(1-\hat{\mathrm{y}})x_j$$

$$= -y(1-\hat{\mathrm{y}})x_j + (1-y)\hat{\mathrm{y}}x_j$$

$$= -(y-\hat{\mathrm{y}})x_j \qquad (4.11)$$

Sonuç olarak $(x_1, x_2 \ldots x_m)$ koordinatları, y gerçek değeri ve ŷ tahmini değer ile verilen bir noktadaki hata fonksiyonun gradyanı şu denklem ile gösterilmektedir:

$$\nabla\mathrm{E} == -(y-\hat{\mathrm{y}})\ (x_1, x_2 \ldots x_m) \qquad (4.12)$$

Gradyan azatma algoritmasının sözde kod adımları aşağıda özetlenmiştir:

Algoritma 2: Gradyan Azaltma

1. Rastgele ağırlık değerleri ile başla.
2. Her bir $(\mathrm{x}_1, \mathrm{x}_2 \ldots \mathrm{x}_\mathrm{n})$ noktası için α öğrenme katsayısı ile w ağırlık ve b bias değerlerini güncelle;

   For i=1…n:

   $\mathrm{w}_\mathrm{i}' \coloneqq \mathrm{w}_\mathrm{i} - \alpha\frac{\partial \mathrm{E}}{\partial \mathrm{w}_\mathrm{i}}$

$$w_i' \coloneqq w_i + \alpha(y - \hat{y})x_j$$

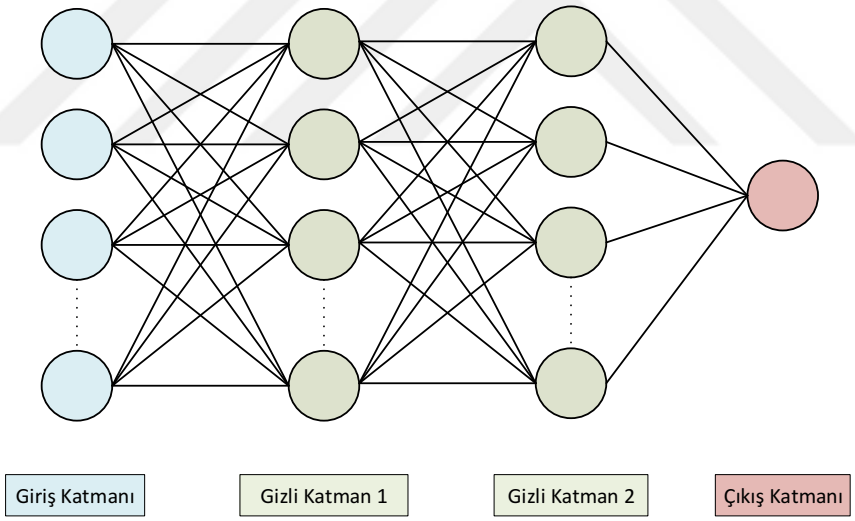
$$b' := b - \alpha\frac{\partial E}{\partial b}$$

$$b' := b - \alpha(\hat{y} - y)$$

3. Hata en aza indirgene kadar yukarıdaki adımları tekrarla.

Gradyan azaltma ve algılayıcı algoritmalarındaki temel farklar, gradyan azaltma algoritmasında $\hat{y}$ çıktı değeri [0,1] arasında herhangi bir değer alabilirken, algılayıcı algoritmasında yalnızca 0 ya da 1 değerini alabilmektedir. Algılayıcı algoritması bir x noktası doğru sınıflandırılmışsa güncelleme yapmazken, gradyan azaltma algoritması daha iyi tahmine ulaşana kadar güncelleme yapabilmektedir.

4.2.4 Çok Katmanlı Algılayıcılar



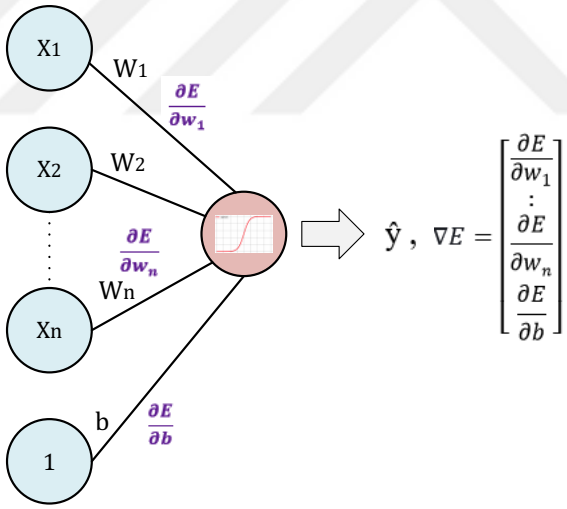
Şekil 4. 7. Çok katmanlı algılayıcı sinir ağları gösterimi

Tek katmanlı sinir ağları (algılayıcılar ve lojistik regresyonlar) doğrusal olarak sınıflandırılabilen problemlerde etkili bir çözüm olmasına rağmen, doğrusal olmayan

problemleri çözememektedir. Gerçek dünyadaki problemlerin çoğu doğrusal değildir ve bu doğrusal olmayan problemlerin çözümü için çok katmanlı algılayıcılar ya da çok katmanlı sinir ağları geliştirilmiştir. Derin ileri beslemeli ağlar ya da ileri beslemeli sinir ağları olarak da bilinen çok katmanlı algılayıcılarda (Multilayer Perceptrons-MLP), giriş ve çıkış katmanları arasına gizli katman olarak adlandırılan yeni bir katman eklenmiştir. Çok katmanlı algılayıcı sinir ağlarının gösterimi Şekil 4.7'de verilmiştir.

4.2.5 İleri Besleme ve Geriye Yayılım

Hata fonksiyonunu ve hesaplanan hata fonksiyonunun gradyanını hesapladıktan sonra, ağırlıklar güncellenerek daha küçük hataya sahip yeni bir model üretilir. Bu süreç en iyi model tahmini gerçekleşene kadar devam eder. Gradyan, Şekil 4.8'de görüldüğü gibi, hata fonksiyonunun her bir $w_1, w_2 \ldots w_n$ ağırlıklarına ve b bias değerlere göre parçalı türevlerinden oluşan ve ∇E ile gösterilen bir vektördür.

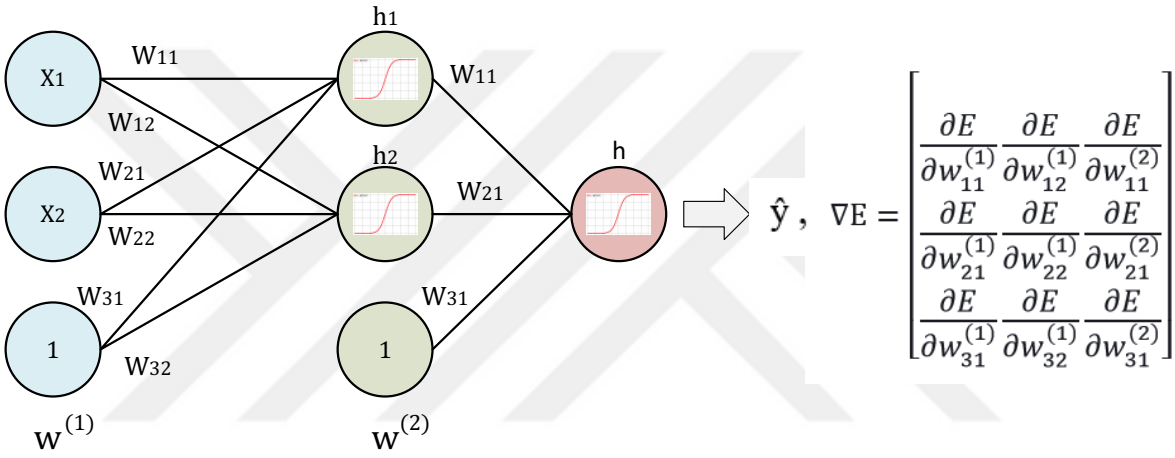


Şekil 4. 8. Tek katmanlı algılayıcılar için hata fonksiyonunun gradyanı

Basit bir algılayıcı için hata fonksiyonu ve gradyanını hesaplamak kolay olsa da, çok katmanlı algılayıcılar ve derin sinir ağları için bu hesaplamalar karmaşıklaşmaktadır. Şekil

4.9'da gösterilen çok katmanlı algılayıcıda gradyan ∇E bir matris görünümündedir. Gradyan azaltma her bir $w_{ij}^{(k)}$ ağırlığı alır ve bu ağırlığa göre hata fonksiyonun gradyanını α öğrenme katsayısı ile çarpıp ekleyerek bu ağırlığı aşağıdaki denklemde gösterildiği gibi güncellemektedir. Güncellenen $w_{ij}^{\prime(k)}$ ağırlıkları ile daha iyi sınıflandırma yapan yeni modeller üretilmektedir.

$$w_{ij}^{\prime(k)} := w_{ij}^{\prime(k)} - \alpha \frac{\partial E}{\partial w_{ij}^{(k)}} \tag{4.13}$$



Şekil 4. 9. Çok katmanlı algılayıcılar için hata fonksiyonunun gradyanı

Şekil 4.9'da $W^{(1)}$ ile gösterilen ağırlıklar birinci katmana $W^{(2)}$ ile gösterilen ağırlıklar ikinci katmana aittir. Verilen bu ağda ileri besleme (feedforward) ve geriye yayılım (backpropagation) işlemleri aşağıda verilmiştir:

1. İleri besleme işlemi için ilk katmanda, ağırlıklar ile girdilerin çarpımı sırasıyla h_1 ve h_2 değerini vermektedir:

$$h_1 = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} x_3 \tag{4.14a}$$

$$h_2 = \mathrm{w}_{12}^{(1)} x_1 + \mathrm{w}_{22}^{(1)} x_2 + \mathrm{w}_{32}^{(1)} x_3 \quad (4.14b)$$

İkinci katmanda, hesaplanan h_1 ve h_2 değerlerine sigmoid fonksiyon uygulanır ve bu katmandaki ağırlık değerleri ile çarpılarak h değeri elde edilir:

$$h = \mathrm{w}_{11}^{(2)} \sigma(h_1) + \mathrm{w}_{21}^{(2)} \sigma(h_2) + \mathrm{w}_{31}^{(2)} \quad (4.15)$$

Son olarak üçüncü katmanda h değerinin sigmoid fonksiyonu alınarak [0,1] arasında bir tahmin çıktı üreten ŷ değeri elde edilir:

$$\hat{\mathrm{y}} = \sigma(h) \quad (4.16)$$

2. Geriye yayılım işlemi ileri beslemenin tersi yönde çalışmaktadır. Zincir kuralı kullanarak her bir ağırlığa göre hata fonksiyonunun türevi alınmaktadır. E hata fonksiyonu ve $\mathrm{w}_{11}^{(1)}$ ağırlığına göre zincir kuralı kullanılarak alınan türev sonucu aşağıdaki denklemde verilmiştir:

$$E(w,b) = -\frac{1}{m} \sum_{i=1}^{m} (y_i \times \log(\hat{\mathrm{y}}) + ((1 - y_i) \times \log(1 - \hat{\mathrm{y}})) \quad (4.17)$$

$$\frac{\partial \mathrm{E}}{\partial \mathrm{w}_{11}^{(1)}} = \frac{\partial \mathrm{E}}{\partial \hat{\mathrm{y}}} \frac{\partial \hat{\mathrm{y}}}{\partial \mathrm{h}} \frac{\partial \mathrm{h}}{\partial h_1} \frac{\partial h_1}{\partial \mathrm{w}_{11}^{(1)}} \quad (4.18)$$

Geriye doğru hata yayılımında daha basit bir hesaplama ile sigmoid fonksiyonun h_1 değerine göre türevi şu şekilde hesaplanabilmektedir:

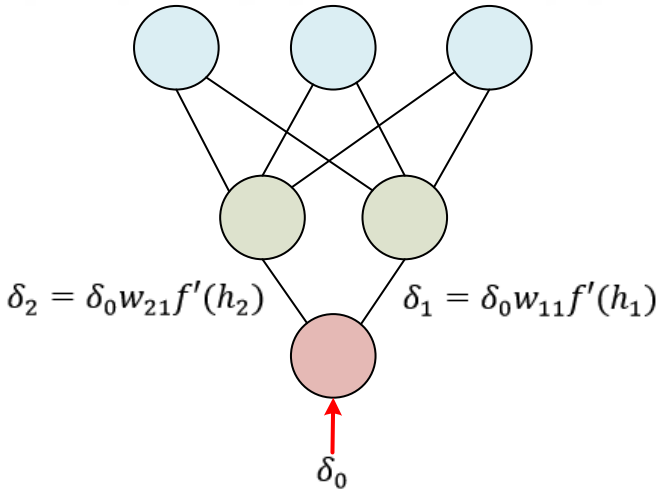
$$\frac{\partial \mathrm{h}}{\partial h_1} = \mathrm{w}_{11}^{(2)} \sigma(h_1)(1 - \sigma(h_1)) \quad (4.19)$$

Son olarak hata fonksiyonunun her bir ağırlığa göre parçalı türevlerinden oluşan ve ∇E ile gösterilen aşağıdaki vektör elde edilmektedir:

$$\nabla E = [\frac{\partial \mathrm{E}}{\partial \mathrm{w}_{11}^{(1)}}, \frac{\partial \mathrm{E}}{\partial \mathrm{w}_{21}^{(1)}} \dots, \frac{\partial \mathrm{E}}{\partial \mathrm{w}_{31}^{(2)}}] \tag{4.20}$$

4.2.6. Geriye Yayılım Algoritması

Geriye yayılım algoritması, iki katmanlı bir ağ için girdi katmanını gizli katmana bağlayan ağırlıklara göre, ikiden fazla katmanlı bir ağ için gizli katmanları birbirine bağlayan ve girdi katmanını gizli katmana bağlayan ağırlıklara göre hatanın zincir kuralı ile bulunmasıdır. Gradyan azaltma ile gizli birimlerin ağırlıklarını güncellemek için her bir gizli birimin son çıktıya ne kadar hata verdiğinin bilinmesi gerekmektedir. Bir katmanın çıktısı katmanlar arasındaki ağırlıklar ile hesaplanır, çıktıdaki hata bilindiğinden dolayı gizli katmanlardan geriye doğru gidilerek ağırlıklar kullanılabilmektedir.



Şekil 4. 10. Geriye yayılım gösterimi

Şekil 4.10'daki geriye yayılım gösterimini daha genel bir denklem ile ifade edersek, her bir k çıktı birimi için δ_k hata değerleri üretilsin. h gizli katmanındaki j biriminin ya da düğümünün δ_j^h hatası şu şekilde genelleştirilmektedir:

$$\delta_j^h = \sum w_{jk}\, \delta_k f'(h_j) \tag{4.21}$$

Şekil 4.10'da gösterilen girdi katmanı ile gizli katman arasındaki w_{ij} ağırlık değerleri ve x_i girdi değerleri için gradyan azaltma ile ağırlık güncellerken δ_j^h hatası şu şekilde kullanılmaktadır:

$$\Delta w_{ij} = \alpha \delta_j^h x_i \tag{4.22a}$$

$$w'_{ij} = w_{ij} + \Delta w_{ij} \tag{4.22b}$$

Yukarıdaki matematiksel işlemler sözel olarak özetlenecek olursa bir sinir ağı eğitimi için kullanılacak geriye yayılım yöntemi aşağıdaki adımlardan oluşacaktır:

- İleri besleme işleminin yapılması
- Modelin çıktısı ile istenilen çıktının hesaplanması
- Model çıktısı ve istenilen çıktı arasındaki hatanın hesaplanması
- İleri besleme işlemini geriye doğru (geriye yayılım) çalıştırarak hatanın her bir ağırlığa dağıtılması
- Ağırlıkları güncellenmesi
- İyi bir model elde edene kadar bu adımların tekrarlanması.

4.3. Derin Öğrenmeye Genel Bakış

Son zamanlarda sıklıkla karşılaştığımız yapay zeka, yalnızca akademik dünyada değil iş çevrelerinde de sıcak bir araştırma alanı olmuştur. Yapay zeka fikri yıllardır bizimle yaşamasına rağmen, birden bire sıcak bir konu haline gelmesinin sebebi derin öğrenmedir. İnsan beyninin yapısını taklit eden derin öğrenme, sinir ağlarının bir uyarlaması ve makine öğrenmesi algoritmalarından birisidir. Fakat sinir ağlarını içeren derin öğrenme, herhangi bir makine öğrenmesi algoritmasından çok daha fazla önem arz etmektedir.

4.3.1 Makine Öğrenmesinin Zorlukları

Bir öğrenme sistemi oluşturmanın bilinen bazı zorlukları ve problemleri vardır. Makine öğrenme sistemi oluşturulurken karşılaşılan sorunlar aşağıda maddeler halinde verilmiştir:

- ***Özellik çıkarımı:*** Özellik çıkarma geleneksel makine öğrenimi sistemleri için en önemli adımlardan birisidir. Çünkü sistemin doğru çalışması, doğru özelliklerin ve özellik sayılarının seçilmesine bağlı olarak değişmektedir. Ancak özellik çıkarım işlemi alan (domain) bağımlıdır, yani yüz tanıma sistemi için seçilen özellikler ile parmak izi tanıma sistemi için seçilen özellikler farklı olacaktır. Aynı şekilde İngilizce bir metin sınıflandırma görevi için geliştirilen sistemin özellik seçimi Türkçe için olandan farklı olacaktır. Makine öğrenimi özellikleri otomatik öğrenebilirse, tüm öğrenme süreci otomatik gerçekleştirilerek daha kolay bir hale gelebilir ve pek çok problem çözülebilir. Bu noktada, derin öğrenme otomatik özellik öğrenimi sağlayarak kolay bir yol sunmuştur.
- ***Gürültü:*** Bir yüz tanıma işleminde, görüntülerdeki yüzler farklı boyutlarda, renklerde, açılarda, ışıklarda ya da düşük çözünürlükte olabilir. Bir metin sınıflandırma işleminde, farklı site kaynaklarından alınan metinler yanlış yazımlar, noktalama işaretleri, yaygın kullanımı olmayan kısaltmalar, eksik cümleler ve sisteme girdi olarak verilecek metinler farklı uzunluklarda cümleler içerebilmektedir.

- ***Aşırı öğrenme:*** Bir yüz tanıma ya da metin sınıflandırma işleminde, model karmaşıklığı arttırılarak her bir eğitim örneği çok iyi bir şekilde sınıflandırılabilir. Ancak modelin test verileri yani daha önceden görülmemiş veriler üzerindeki performansı düşük olabilmektedir. Bir model eğitim örneklerinde çok iyi çalışırken test verilerinde kötü performans sergiliyorsa aşırı öğrenmiştir. Eğitim verisi üzerinde eğitilmiş modelin hatası doğru bilgiyi vermez, dolayısıyla eğitilmiş modelin test örnekleri üzerindeki performans yani genelleme hatasına bakılması gerekmektedir.
- ***Öğrenme algoritmasının seçimi:*** Belirli bir iş için kullanılan model doğru çalışmayabilir ve alternatif algoritmalara ihtiyaç duyulabilir. Araştırmacının, problemin ihtiyacına yönelik iyi bir öğrenme algoritmasını seçilebilmesi için pek çok modeli test etmesi ve ihtiyaca uygun algoritmayı geliştirmesi ya da kullanması gerekmektedir. Ayrıca bir tanıma ya da sınıflandırma problemi için açıklayıcı özellikler hakkında ön bilgiye sahip olmak farklı türde sınıfları ayırmak için daha özgün modeller ortaya koymada önemlidir.
- ***Eksik değerler:*** Eksik özellikler temel olarak veri eksikliğinden ya da tercih etmeme seçeneğinin kullanılmasından kaynaklanmaktadır. Örneğin, tanıma zorluğundan dolayı bazı ırklardaki yüzler eğitim setine dahil edilmeyebilir ya da haber sınıflandırma işleminde kültür sınıfında az veri elde edildiği için sınıf kategorisinden çıkarılabilir.

4.3.2. Metin Sınıflandırma Problemi

İnsan dilinin gösterimi ve otomatik analizi için pek çok hesaplama teknikleri içeren doğal dil alanında, metin sınıflandırma problemi yaşanan zorluklardan birisidir. Cambria ve White [158], bu alandaki zorlukların üstesinden gelmek için üç farklı dönem ya da eğri ise NLP araştırmasının ön görülen gelişimini açıklamıştır: Sözdizim (Syntactics), anlambilim (Semantics) ve edimbilim (Pragmatics). Metnin sözdizimsel gösterimini kullanan yaklaşımlar kelime frekanslarına daha açık anlamıyla kelimelerin metinde oluşma sıklığına dayanmaktadır. Günümüzde mevcut yaklaşımların çoğu hala metnin sözdizimsel gösterimini kullanmaktadır. İkinci eğilim olan anlambilim, metnin içeriğine yoğunlaşarak anlamı üzerinde çalışır.

Sözdizimsel eğilimde metinler kelime torbası (bag of words) model ile temsil edilirken, anlambilim eğiliminde kavram torbası (bag of concepts) modeli ile gösterilir. Kavram torbası, bir vektör uzayındaki kelimelerin anlamlarını temsil eden modelleri temel almaktadır. Edimbilim eğilimini gösteren anlatım torbası (bag of narratives) modelinde, her bir metin parçası daha küçük ve birbirine bağlı bölümler ile gösterilerek daha detaylı bir metin anlama ve hesaplama seviyesine çıkılacağı ön görülmemektedir.

Metin sınıflandırma sürecini zorlaştıran pek çok faktör vardır. Bu faktörlerden birisi, her biri farklı yapı ve kurallarda yüzlerce farklı doğal dilin olmasıdır. Diğer bir zorluk, kelimelerin anlamı bağlamına (context) bağlı değiştiğinden belirsizlik (ambiguous) olabilmektedir.

4.3.3. N-gram ve Kelime Torbası Modellerinin Zayıf Yönleri

Özellik mühendisliği teknikleri olarak da bilinen n-gram ve kelime torbaları, metinlerin sabit uzunluklu vektör gösteriminde kullanılan en yaygın modellerdir. Bunun ile birlikte bu yaklaşımların pek çok dezavantajı bulunmaktadır:

1. Kelime torbalarında kelime sırası göz önünde bulundurulmaz. Cümle yapısı ve kelime yerleşimi göz ardı edilir. Bu nedenle aynı kelimelerin farklı dizilimlerini kullanan iki ayrı cümle, benzer vektör gösterimi ile temsil edilir. Örneğin "yapay zeka programı AlphaGo Lee Sedol ile oynadığı oyunu kazandı" cümlesi ile "Lee Sedol yapay zeka programı AlphaGo ile oynadığı oyunu kazandı" cümlesi farklı ve önemli sonuçlar içeren cümleler olmasına rağmen, kelime torbası modeli bu cümlelerin gösterimini benzer çıkarır. Böylece cümlelerin dolayısıyla dilin anlam ve içeriğini yakalayamaz.
2. N-gram torbaları kısa metinlerde kelime sırasını kaybetmese de, yüksek boyutluluk ve veri seyrekliği (data sparsity) problemleri ile karşılaşmaktadır. Ayrıca kelime torbası modelleri ile aynı sonucu ve performansı yakalamak için daha fazla hafıza gereksinimine ihtiyaç duymaktadır.

N-gram ve kelime torbası modelleri, kelimelerin anlamlarından çok kelimeler arasındaki uzunlukların ölçümüne odaklanmaktadır. Bir dokümandaki kelimeler arasındaki ilişkileri göz

önünde bulundurmazlar. Metin sınıflandırma problemleri üzerine kurulan tez çalışmasının bu bölümünde, bu modellerin zayıf yönlerini aşmak, farklı uzunlukta ve içerikte metinlerin sözdizimsel eğiliminden çok anlambilimine ulaşmak hedeflenmiştir. Bu amaç doğrultusunda dokümanların ve cümlelerin dağıtık gösterimlerine yönelim gerçekleştirilmiştir [159].

4.3.4. Dağıtık Gösterimler

Yapay zekanın en önemli alanlarından biri olan Doğal Dil İşleme, metinleri etkili bir şekilde işleyerek farklı yapılarda ve kurallarda insan dillerinin anlaşılmasını amaçlamaktadır. NLP çalışmalarında, kelime ve dokuman vektörleri oluşturulurken genellikle özellik olarak kelimelerin sayısı temel alınmaktadır ve vektörler arasındaki mesafeler ölçülerek anlamsal benzerlik çıkarılmaktadır [160]. Kelimeler üzerinde sayıma dayalı modellerde eğitim setinde bulunmayan kelimeler, test sürecinde performansa zarar vermektedir. Diğer bir problem ise boyutsallıktır. Büyük bir kelime sözlüğü üzerinde indekslenen vektörlerin çok seyrek (sparse) olması, modellerin eğitim verisini ezberlemesine (overfitting) yol açar [161]. Derin öğrenme ile oluşturulan dil modelleri, farklı kelimeleri sayma yerine dağıtık kelime vektör gösterimlerini kullanmaktadır [162].

Kelime torbası modellerinin problemlerini çözmeyi hedefleyen kelime temsil modelleri (word embedding models), anlamsal olarak benzer kelimelerin yakın noktalarda eşlendiği (temsil edildiği) her bir kelime için bir vektör öğrenmektedir. Bu modeller ayrıca boyut azaltma sağlayarak kelime torbası modellerinin boyutsallık sorununu aşmaktadır. Böylece tüm kelimelerin anlamını yakalayan daha küçük boyutta bir vektör uzayı elde edilmektedir. Kelime temsilleri, bir dokümandaki kelimelerin anlamlarını ve birbirleriyle olan ilişkilerini öğrenebildiği için aşağıda tanımlanan duygu analizi ve doküman sınıflandırma problemlerinin çözümünde çok önemli bir katkı sağlamıştır.

1. ***Duygu Analizi:*** Duygu analizi, bir dilin anlamını analiz eden bir anlambilim problemidir. Bu problemde olumlu (pozitif), olumsuz (negatif), mutlu, üzgün ve benzeri duyguları sınıflandırarak bir kişinin hislerinin anlaşılması hedeflenir. Bu

sınıflandırmanın yapılabilmesi için öncelikle iyi bir veri seti gereklidir. İnternet ve sosyal ortamda milyonlarda yorum ve sözel ifadeler bulunmaktadır fakat bu verilerin çoğu genellikle işaretsizdir. Etiketli veri bulmak da oldukça zordur. Bu tip sınıflandırıcının çoğu ticari ve akademik uygulaması, müşterinin, kişinin ya da yazarın bir ürün, film, kitap ya da diğer hizmetler hakkında pozitif olup olmadığını tahmin etmeyi amaçlayan ikili bir sonuç almayı hedefler.

2. ***Doküman Sınıflandırma:*** Doküman sınıflandırma, duygu sınıflandırması ile teknik olarak benzer bir anlam taşımaktadır. Her iki sınıflandırma da, metinlerin kategorilere ayrılması söz konusudur. Doküman sınıflandırması, dokümanları türüne göre ayırmayı ya da kategorize etmeyi hedeflemektedir. Doküman sınıflandırma sistemlerinin en bilinen uygulamaları haber ve akademik yayın sınıflandırmasıdır. Bilgisayarın doğal dili anlama yeteneğine dayanan ve anlambilimin çözümünde çok güçlü bir yere sahip olan duygu analizi ve doküman sınıflandırma problemi, bu tez çalışmasının odaklandığı temel konulardır.

4.3.5. Derin Öğrenmenin Tarihi

Derin öğrenmede kullanılan yaklaşımların ya da tekniklerin zaman çizelgesi üzerinde gösterimi Tablo 4.1.'de verilmiştir.

Rosenblatt, 1957 yılında ilk algılayıcıyı keşfettikten sonra, sinir ağları gittikçe dikkat çekmeye başladı. Fakat bellek ve işlemci hızı kısıtlamaları sebebiyle potansiyel uygulamalar ve araştırmalar büyük bir sorunla karşı karşıya geldi. Son on yılda, daha güçlü ve daha ucuz bilgisayarlar sayesinde, makine öğrenimi araştırmacıları, metinler, resimler ve sesler içeren çok geniş veri kümeleri kullanan birkaç derin katmana sahip karmaşık ve büyük modeller inşa ettiler. Bölüm 4.3.1'de verilen zaman çizelgesindeki bir dizi çalışmalar ve araştırmalar sonucunda Derin Öğrenme ile ilk kez 2006 yılında Toronto Üniversitesinden Profesör Hinton ve çalışma ekibinin yayınladığı bir makale [3] ile tanışılmıştır. Bu makalede, derin fikir ağları (deep belief nets) olarak adlandırılan bir method sunuldu. Derin Fikir Ağları (DBN), her bir görüntü tanıma yönteminin doğruluğunu ölçmek ve tahmin etmek için tercih edilen MNIST veri

tabanı kullanılarak test edilmiştir. Kullanılan MNIST veri tabanı, 0-9 sayıları arasında değişen 70.000 tane 28 x 28 piksel el yazısı karakterlerinin görüntü verilerini tutmaktadır.

Tablo 4. 1. Derin öğrenme zaman çizelgesi

Geliştirilen Yaklaşım	**Yıl**	**Literatürdeki Çalışması**
Sinir Ağları	1943	Warren McCulloch ve logician Walter Pitts [156]
Bilgisayar Mekanizması ve Zeka	1950	Alan Turing [163]
Algılayıcı	1957	Frank Rosenblatt [157]
Adaline	1960	Bernard Widrow ve Marcian Hoff [164]
Algılayıcılar ve XOR Problemi	1969	Marvin Minsky and Seymour Papert [165]
Geriye Yayılım Algoritması	1974	Paul Werbos [166]
Neocognitron Ağı	1980	Kunihiko Fukushima [167]
Hopfield Ağı	1982	John Hopfield [168]
Boltzmann Makinesi	1985	David Ackley ve arkadaşları [169]
Çok Katmanlı Algılayıcı	1986	Rumelhart, Hinton ve Williams [170]
Kısıtlanmış Boltzmann Makinesi	1986	Paul Smolensky [171]
Tekrarlı Sinir Ağları	1990	Jeffrey Elman [172]
LeNet	1990	Yann LeCun ve arkadaşları [11]
İki-yönlü Tekrarlı Sinir Ağı	1997	Mike Schuster ve Kuldip Paliwal [173]
Uzun Kısa Süreli Bellek	1997	Sepp Hochreiter [174]
Derin Fikir Ağları	2006	Geoffrey Hinton ve arkadaşları [3]
Derin Boltzmann Makinesi	2009	Salakhutdinov ve Hinton [175]
Dropout	2012	Geoffrey Hinton ve arkadaşları [176]
Üretici Çekişmeli Ağlar (GANs)	2014	Ian Goodfellow ve arkadaşları [177]
Kapsül Ağları	2017	Sara Sabour, Nicholas Frosst ve Geoffrey Hinton [178]

Yeni yaklaşımların ortaya çıkışı, çok büyük miktarda veriler, bu verileri hızlı bir şekilde hesaplamak ve eğitmek için güçlü hesaplama kaynaklarının ortaya çıkışı ile birlikte derin

öğrenmedeki gelişim hızlıca artmıştır. Derin öğrenme mimarilerinin akademik ve iş dünyasında çok sıcak bir çalışma alanı olmasındaki unsurlar aşağıda sıralanmıştır:

1. Metin, görüntü ve ses gibi farklı formatlarda, çok büyük ve yüksek kalitede etiketli veri setlerine erişme kolaylığı
2. Paralel bir şekilde karmaşık hesaplamalar yapabilen güçlü GPU birimleri, Google Colaboratory gibi derin öğrenme modellerinin eğitilmesine olanak sağlayan ve tamamen bulutta çalışan ücretsiz GPU ortamlarının sunulması
3. AlexNet [179], GoogleLeNet [180], VGG [181], R-CNN [182], GAN [177], ResNets [183] ve Inception modülleri gibi yeni derin öğrenme mimarilerinin geliştirilmesi
4. TensorFlow, Keras ve PyTorch gibi kullanımı kolay ve hızlı uygulanabilir açık kaynak derin öğrenme platformlarının olması
5. ReLU gibi sıfırlanan gradyan problemine çözüm geliştiren aktivasyon fonksiyonlarının keşfedilmesi
6. Dropout gibi yeni düzenlileştirme teknikleri ve veri arttırma (data augmentation) yöntemleri ile aşırı öğrenme problemiyle karşı karşıya kalmadan çok büyük ağları eğitebilme olanağı
7. Model parametrelerini güncellerken daha iyi ve hızlı sonuçlar üreten RMSprop, Adagrad, Adamax, TFOptimizer ve ADAM gibi optimizatörler (optimizer).

4.3.6. Derin Sinir Ağları Yöntemleri

Derin Öğrenme Yöntemleri dört temel kategoride sınıflandırılabilir:

- Konvolüsyonel Sinir Ağları
- Tekrarlı Sinir Ağları
- Kısıtlanmış Boltzmann Makineleri
- Otomatik Kodlayıcılar

Bu temel yöntemlere yönelik geliştirilen ve ön plana çıkan en yeni çalışmalar aşağıda verilmiştir:

- Konvolüsyonel Sinir Ağı tabanlı yöntemler: AlexNet [179] , VGG [181], GoogleLeNet [180], Clarifai [184].
- Tekrarlı Sinir Ağı tabanlı yöntemler: Uzun Kısa Süreli Bellek [174], Kapılı Tekrarlayan Hücre [185], İki yönlü Tekrarlı Sinir Ağı [173], İki-yönlü Uzun Kısa Süreli Bellek [186].
- Kısıtlanmış Boltzmann Makinesi tabanlı yöntemler: Derin Fikir Ağları [3], Derin Boltzmann Makineleri [175].
- Otomatik Kodlayıcı tabanlı yöntemler: Seyrek Otomatik Kodlayıcı [187], Gürültü Temizlemeli Otomatik Kodlayıcı [188], Varyasyonel Otomatik Kodlayıcı [189].

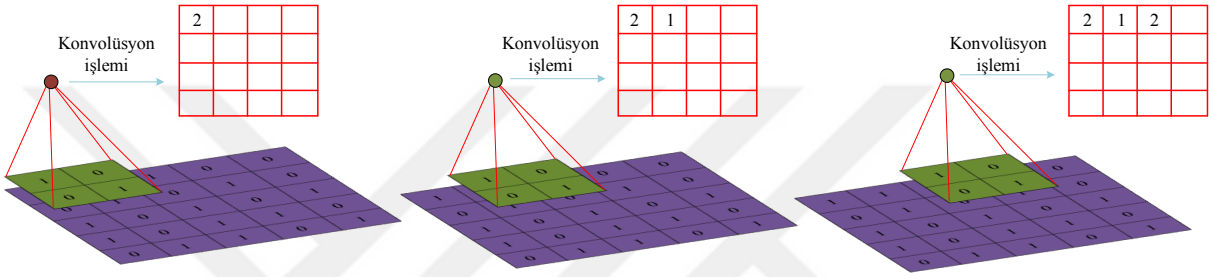
Bu tez çalışması Konvolüsyonel Sinir Ağı tabanlı yöntemler ve Tekrarlı Sinir Ağı tabanlı yöntemlere odaklanmıştır. Bu derin sinir ağları yöntemlerinin ayrıntılı analizi aşağıda verilmiştir.

4.4. Konvolüsyonel Sinir Ağları

Evrişimsel Sinir Ağları olarak da geçen Konvolüsyonel Sinir Ağları (Convolutional Neural Networks - CNN), görüntü tanıma ve doğal dil işlemine dayalı sınıflandırma modellerinin oluşturulmasında önemli rol oynamaktadır. Derin öğrenme sinir ağlarının bir kategorisi olan Konvolüsyonel Sinir Ağları, görüntü tanıma, nesne tanıma, otomatik video sınıflandırma ve bilgisayar görmesi alanındaki diğer pek çok alanda insanüstü performans göstermiştir. Bu tez çalışmasında anlatılacağı gibi literatür çalışmalarında yaygın olarak kullanılmasalar da NLP alanında da popüler bir uygulama alanı haline gelmeye başlamıştır. Konvolüsyon Sinir Ağları popüler konvolüsyon işlemi ve sinir ağlarının birleşiminden oluşmaktadır.

4.4.1. Konvolüsyon İşlemi

Konvolüsyon (evrişim) işleminin temel amacı verilen bir görüntüden ya da metinden özelliklerin ya da bilgilerin çıkarılmasıdır. Herhangi bir görüntü ya da cümleler ve kelimelerden oluşan metin, bir değerler matrisi olarak düşünülebilmektedir. Konvolüsyon işleminin amacı ise bu matrisi belirli filtreler ile Şekil 4.11'deki gibi tarayarak istenilen görüntüler ve metinler için açıklayıcı özellikleri çıkarmaktadır.



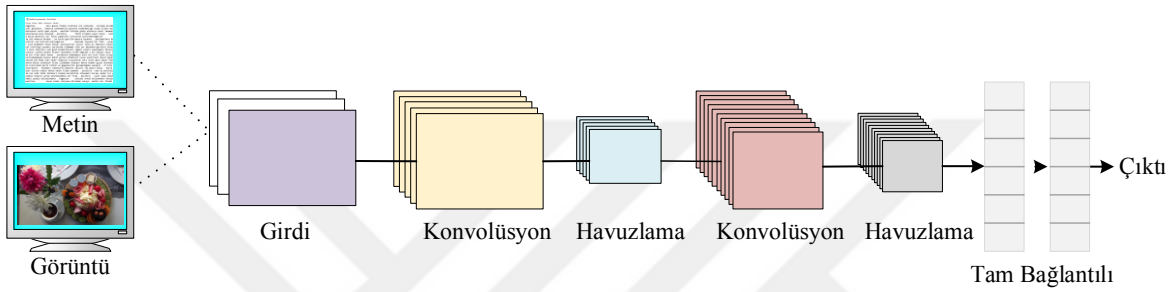
Şekil 4. 11. 5x5 boyutlu bir matris üzerinde 2x2 filtre uygulanan konvolüsyon işlemi

Şekil 4.11'de mor renk ile gösterilen 5x5 boyutlu bir görüntü ya da cümle-kelime matrisi düşünüldüğünde, her bir hücre sıfır ya da birlerden oluşan piksel ya da kelimeler ile ilişkili yoğunluk değerlerinden oluşur. Burada yeşil renkle gösterilen 2x2'lik bir konvolüsyon filtresi kullanılarak birer kaydırma (stride) ile girdi matris ve filtrenin ilişkili değerleri arasında birebir çarpım gerçekleştirilir. Her bir çapım sonucu alınan tam sayılar toplanarak, kırmızı renkli özellik haritası denilen 4x4'lük çıktı matrisinin ilgili yerine yerleştirilir. Havuzlama katmanında ise özellik haritasındaki en büyük değerler ya da değerlerin ortalaması alınarak matris boyutu azaltılır ve dolayısıyla ağdaki parametre sayısı azaltılmaktadır. Daha fazla filtre ile verilen girdiden daha fazla özellik çıkarılabilmektedir. Ayrıca rastgele ağırlıklar ile değişen filtre değerleri, her seferinde farklı özellik haritaları verecektir. Matematiksel olarak konvolüsyon işlemi aşağıdaki denklemde tanımlanmıştır:

$$a(t) = (x \circledast w)(t) = \int_{-\infty}^{\infty} x(b)w(t-b)\,db \tag{4.23}$$

Burada x girdi verisi, w filtre ve a özellik haritasıdır.

4.4.2. CNN Mimarisi



Şekil 4. 12. Tipik bir CNN mimarisi

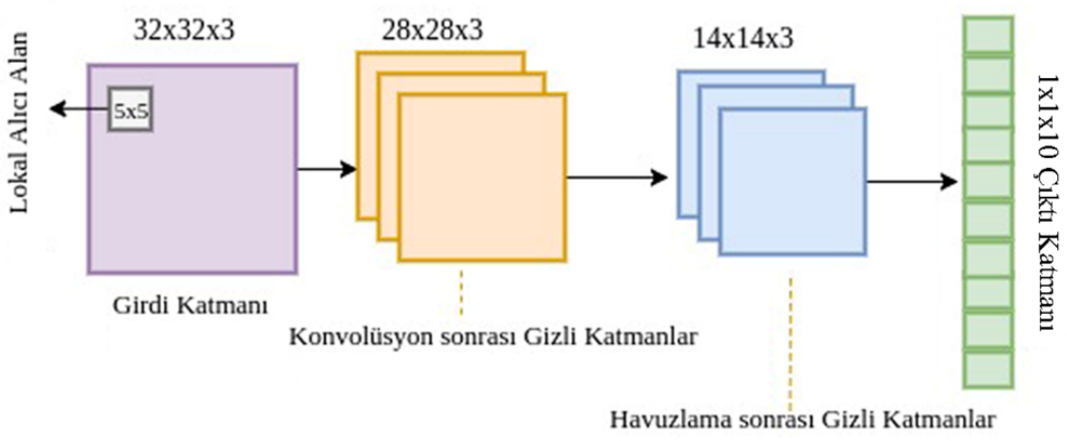
Şekil 4.12'de gösterilen tipik bir CNN mimarisinde her bir konvolüsyon katmanını doğrusal olmayan bir fonksiyon olan doğrultulmuş lineer birim (ReLU) ve sonrasında havuzlama katmanı takip etmektedir. Bu mimari katmanları şu şekilde özetlenebilmektedir:

[Girdi->Konvolüsyon->ReLU->Havuzlama->Konvolüsyon->ReLU->Havuzlama->Tam Bağlantılı]

Seçilen belirli bir filtrenin ya da filtrelerin (çekirdek de denilmektedir), girdi matrisi üzerinde kaydırılması ile konvolüsyon katmanı sonrası özellik haritaları oluşturulmaktadır. Oluşan özellik haritaları daha sonra bir sonraki katmanın yeni girdileri olacaktır. Havuzlama (pooling) katmanında kullanılacak olan maksimum havuzlama veya ortalama havuzlama teknikleri gelen girdi boyutunu azaltarak ağın hesapsal yükünü, hafıza kullanımını ve parametre sayısını azaltmaktadır. Böylece modelin eğitim setini aşırı öğrenmesinin önüne geçilmektedir.

Son olarak birkaç tam bağlı katmanın birleşiminden oluşan son katman bir tahmin yani çıktı üretmektedir. Örneğin çoklu sınıflandırma için son katmanda softmax fonksiyonu kullanılarak verilen girdilerin tahmini sınıf olasılıkları çıktı olarak verilmektedir.

Geleneksel sinir ağları büyük ölçekli görüntüleri sınıflandırmada pek çok problem ile karşı karşıya kalmaktadır. Şekil 3'te gösterilen örnek bir görüntü senaryosunda 32x32x3 boyutunda sırasıyla genişlik, derinlik ve yüksekliğe sahip bir girdimiz olduğunu düşünelim. Geleneksel sinir ağlarında, konvolüsyon katmanı kullanmadan ilk gizli katmanda her bir tam bağlantılı nöronun $32\times32\times3 = 3072$ ağırlık sayısı olacaktır. Bu ağırlık sayısı başlangıç olarak uygun görülse de daha yüksek boyutlu görüntülerde hesaplama zorluğu ile karşılaşılacaktır. Örneğin 224x224x3 piksel boyutunda her bir nöronun ağırlık sayısı $224\times224\times3 = 150528$ olacaktır. Görüntü boyutuna ek olarak katman sayısının da artışı ile birlikte oluşturulan sinir ağı hızla aşırı öğrenme problemine maruz kalacaktır. Bu sorunun çözümü için Şekil 3'te görüldüğü gibi her girdi matrisi üzerinde matris çarpımı gerçekleştirmek yerine 5x5 boyutunda bir filtre kullanılarak konvolüsyon işlemi gerçekleştirilmektedir. Böylelikle her bir nöron konvolüsyon katmanında girdi matrisinin 5x5x3 boyutlu bölgesine bağlanacaktır ve toplam ağırlık $5\times5\times3 = 75$ ile sonuçlanacaktır.



Şekil 4. 13. CNN mimarisi oluşturmak için üç temel işlem: yerel alıcı alanlar, konvolüsyon ve havuzlama

Konvolüsyon katmanından sonra ReLU fonksiyonu kullanılmaktadır. ReLU fonksiyonu aşağıdaki denklemde verildiği üzere negatif girdiler için 0 değerini alırken x pozitif girdiler için x değerini almaktadır.

$$f(x) = \max(0, x) \tag{4.24}$$

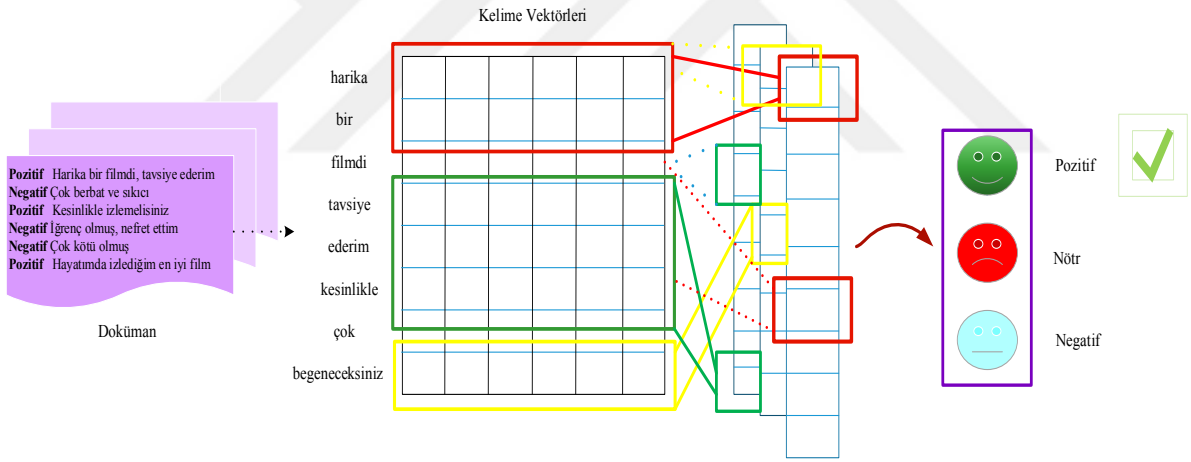
ReLU fonksiyonunun en önemli avantajlarından birisi eğitim süresince Olasılıksal Gradyan Azaltma (SGD) gibi öğrenme algoritmalarının yakınsamasını önemli bir ölçüde hızlandırmasıdır. Havuzlama katmanında, konvolüsyon katmanı sonrası çıktı matrisin boyutu her iki yükseklik ve genişlik için azaltılmaktadır. Şekil 3'te sarı renkle gösterilen 28x28x3 boyutlu konvolüsyon katmanı çıktı matrisi üzerinde 2x2'lik bir havuzlama işlemi sonucu boyut yarıya düşürülerek mavi renkle gösterilen 14x14x3 boyutunda havuzlama katmanı çıktısı alınmıştır. Yaygın olarak kullanılan iki tane havuzlama tekniği vardır:

1. *Maksimum havuzlama:* Özellik haritası üzerindeki 2x2'lik havuzlama alanındaki dört değerin maksimumu seçilerek tek bir değer olarak çıktı matrise yerleştirilir.
2. *Ortalama havuzlama:* Özellik haritası üzerindeki 2x2'lik havuzlama alanındaki dört değerin ortalaması alınarak tek bir değer olarak çıktı matrise yerleştirilir.

Sıklıkla tam bağlantılı katman ya da dense katman olarak adlandırılan çıktı katmanında tüm matris 1x1x10 boyutunda tek bir sınıf vektörü haline verilmiştir. Bu katmandaki nöron sayısı sınıf sayısına eşittir ve katman çıktısı sınıf puanını yani verilen girdinin bir sınıfa ait olma olasılığını vermektedir. Sonuç olarak verilen bir girdi, bir dizi katmandan geçerek çıktı katmanında tahmini sınıf puanına dönüşmektedir. Her katmanın parametre ya da ağırlık gereksinimi farklıdır ve bu ağ parametreleri geriye yayılım sürecinde gradyan azaltma tabanlı algoritmalar ile öğrenilmektedir. Örneğin olasılıksal gradyan azaltma yöntemi başlangıç olarak rastgele verilen ağırlık değerlerini kullanır ve eğitim verisinde verilen etiket değerlerine göre bir hata değeri hesaplar. Bu hatayı azaltmak için ağırlıkları günceller ve bu işlem en iyi model oluşana yani yakınsama elde edilene kadar devam etmektedir.

4.4.3. Metin Sınıflandırma için CNN mimarisi

CNN'ler metin verileri gibi diğer giriş veri tiplerine de uygulanabilmektedir. Metinler, 2 boyutlu (2B) resimlerden farklı olarak, 1 boyutlu giriş verilerine sahiptir. Bu nedenle, 1 boyutlu (1B) konvolüsyon katmanları kullanılmaktadır. Şekil 4.14'te gösterilen örnek bir duygu sınıflandırması mimarisinde, pozitif ve negatif yorumları içeren bir dokuman girdisinin temsil katmanı çıktısı 2B bir matris olacaktır. Bu matriste her satır benzersiz bir kelimeyi gösterir ve satırdaki tüm elemanlar kelime vektörleridir. Örneğin 8x128 boyutlu bir matriste, 8 kelime satırı vardır ve her kelime 128 elemanlı vektör uzayında temsil edilir. Değişken sayıda filtreler kullanarak bu matris üzerinde tarama işlemi gerçekleştirilir. Genellikle 3, 4 ve 5 olarak seçilen filtre yüksekliği bir kere de kaç kelime kaydırılacağını belirlemektedir. Bir dokümanın duygusunu sınıflandıran problemlerde genellikle CNN mimarisi LSTM mimarilerine göre daha hızlı ve iyi performans ile çalışmaktadır [190].



Şekil 4. 14. Metin sınıflandırmada CNN mimarisi

4.4.4. Literatürde CNN

Derin yapay sinir ağları araştırmacıları ve özellikle CNN mimarisi için ilham kaynağı olan Hubel ve Wiesel [191], görsel kortekstеki nöronların, alıcı alan olarak da bilinen görsel alandaki küçük bir bölgeye bireysel olarak cevap verdiğini göstermiştir. Daha sonra da basit hücreler (simple cells) ve karmaşık hücreler (complex cells) olmak üzere öncelikli görsel kortekste iki çeşit hücre bulmuştur. 1965 yılında Ivakhnenko ve Lapa [192], derin öğrenme algoritmalarının ilk uygulamalarını gerçekleştirmiştir. Her katmanda en iyi özellikler çokterimli (polinom) aktivasyon fonksiyonları ile istatiksel olarak analiz edilerek iyi özellikler seçilip manuel bir şekilde bir sonraki katmana aktarılmıştır. 1971 yılında ise *alpha* olarak adlandırılan bir grup veri işleme algoritmaları ile eğitilen 8 katmanlı bir derin sinir ağı önermiştir [193]. İlk Konvolüsyonel Sinir Ağları uygulaması olarak geçen ve Hubel ve Wiesel'ın çalışmalarından esinlenen Kunihiko Fukushima, önerdiği *neocognitron* ağı [167] ile sıfırdan dokuza kadar değişen el yazısı rakamlarını öğrenmeyi amaçlamıştır. Önerilen ağda, her katmanın basit hücreler ve karmaşık hücrelerden oluştuğu 9 katman vardır. Görsel korteksin matematiksel modellemesi için önemli bir çalışma olsa da, bu modelin parametrelerinin öğrenilmesi zor olmuştur. Rumelhart, Williams ve Hinton [170] sinir ağlarını ve dağıtık gösterimleri öğrenebilen geriye yayılım algoritmasını başarılı bir şekilde uyguladıktan sonra Yann LeCun ve arkadaşları [11] bu geriye yayılım algoritmasının pratik uygulamalarda da çalıştığını göstermiştir. Böylece Konvolüsyonel Sinir Ağlarının gelişimi, Yann LeCun ve arkadaşlarının el yazısı rakamlardan oluşan görüntüleri sınıflandırmak için önerdiği yeni bir sinir ağı mimarisi ile 1990 yılında başlamıştır. Fakat hafıza ve donanım kısıtlamaları, eğitim örneklerinin eksikliği nedeniyle, bu sinir ağlarının gelişiminde önemli ilerlemeler kaydedilememiştir. Sinir ağı modellerinin karmaşıklığı arttıkça, bu modelleri eğitmek için çok büyük miktarda veri seti ihtiyacı doğmuştur. Daha büyük veri setleri daha fazla hesaplama gücü gerektirmektedir.

Son zamanlarda, ImageNet [194] gibi mevcut veri setlerindeki artış ve Grafiksel İşleme Birimi (GPU) ve Merkezi İşleme Birimlerinin (CPU) hesaplama gücündeki gelişmeleriyle birlikte daha karmaşık ve daha büyük modelleri eğitmek mümkün hale gelmiştir. Böylece pek çok CNN mimarisinin oluşturulmasının önündeki engeller kalkmıştır. 2010 yılından beri

düzenlenen ImageNet görsel tanıma yarışması ile birlikte bugün bilinen en yeni mimariler ortaya çıkmıştır. 2012 yılında en iyi buluş Toronto Üniversitesi' ndeki Alex Krizhevsky ve takım arkadaşları tarafından geliştirilen AlexNet mimarisidir. 2014 yılında ise yarışmada iki mimari ön plana çıkmıştır. İlki Oxford Üniversitesi' nde görsel geometri grubu tarafından sunulan VGG mimarisidir. Bu mimarinin VGG-16 ve VGG-19, sırasıyla 16 ve 19 katman sayısına sahip iki versiyonu vardır. İkincisi ise Inception olarak bilinen GoogLeNet mimarisidir. 2015 yılında ise ResNet olarak adlandırılan ve Microsoft araştırma grubu tarafından sunulan bir ağ mimarisi bu yarışmayı kazanmıştır. Bahsedilen bu popüler CNN mimarileri aşağıda verilmiştir:

AlexNet Mimarisi: Bilgisayarlı görmede derin öğrenmenin ilk popüler kullanımı AlexNet modeli ile başlamıştır. 10 milyon görüntü ve 1000 farklı görüntü kategorisi olan ImageNet veri tabanındaki görüntüleri sınıflandırmayı amaçlayan ve 8 katmandan oluşan bu mimari dizilimi şu şekildedir:

[Girdi -> Konv1 -> Havuzlama1 -> Konv2 -> Havuzlama2 -> Konv3 -> Konv4 -> Konv5 -> Havuzlama3 -> Tam Bağlantılı1 -> Tam Bağlantılı2 -> Tam Bağlantılı3)]

Konv ile gösterilen her bir konvolüsyon katmanının filtre boyutu ve filtre sayısı Tablo 4.2'de verilmiştir:

Tablo 4. 2. AlexNet konvolüsyon katmanı parametre değerleri

Konvolüsyon Katmanı	**Filtre Boyutu**	**Filte Sayısı**
Konv1	11x11	96
Konv2	5x5	256
Konv3	3x3	384
Konv4	3x3	384
Konv5	3x3	256

AlexNet modelinde tüm havuzlama katmanları 3x3 boyutunda olup, Tam Bağlantılı-1 ve Tam Bağlantılı-2 katmanlarında 4096 nöron ve son katman olan Tam Bağlantılı-3' de 1000

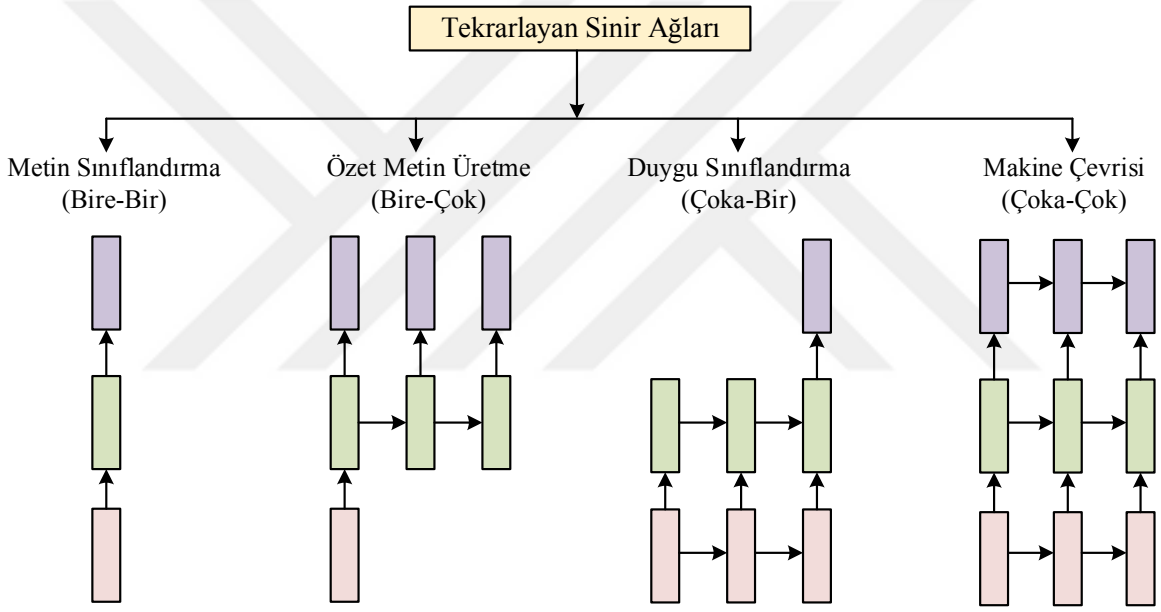
nöron vardır. En son katman dolayısıyla 1000 sınıfı temsil etmektedir. AlexNet ayrıca ReLU aktivasyon fonksiyonu ve dropout tekniğinin derin sinir ağlarında kullanımının öncüsüdür.

VGG Mimarisi: 2014 yılında Simonyan ve arkadaşları tarafından geliştirilen bu mimari daha derin ağların daha iyi ağlar olduğu fikrine dayanarak tasarlanmıştır. AlexNet'e göre daha yüksek doğruluk performansı sağlasa da çok fazla parametresi olduğundan (yaklaşık 140 milyon) çok fazla bellek kullanım ihtiyacı olmuştur. Diğer yandan AlaxNet'e göre daha küçük filtreler kullanılmıştır. Bu mimari tüm konvolüsyon katmanlarında değişken sayıda 64, 128, 256 filtre sayısı ile sabit 3x3 boyutlu filtreler kullanmaktadır.
Inception Mimarisi: Bu mimarinin en büyük katkısı toplam 22 katman ile parametre sayısı 60 milyon olan AlexNet mimarisine karşılık parametre sayısını 5 milyona düşürmesidir (yaklaşık 12 kat daha az parametre). Inception-v1, Inception-v2, Inception-v3 ve Inception-v4 olmak üzere dört versiyonu vardır.

ResNet Mimarisi: Araştırmacılar derin CNN mimarisi oluştururken bir problem ile karşı karşıya kalmışlardır. Önceki mimarilerde görüldü ki derin mimarilere katman ekledikçe bir noktaya kadar performans artarken bir noktadan sonra hızlıca düşüş yaşandı. Sıfırlanan gradyan olarak bilinen bu problem ağ eğitimi sırasında geriye yayılım ile ortaya çıkmaktaydı. Görüntü tanımadaki en son teknoloji ResNet mimarisi de önceki mimariler gibi "ağ ne kadar derin olursa performans o kadar artar" fikri üzerine kurulmuştur. Ancak artan ağ derinliği ile birlikte, bir önceki katmandan gelen gradyanlara göre her katmanın gradyanı zincirleme kuralı ile hesaplandığından sıfırlanan gradyan problemi de artmaktadır. Böylece katman sayısı arttıkça gradyan değerleri küçülür ve sıfıra yaklaşır. Bu problemi çözmek için ResNet, gradyan hesaplamak yerine $f(x)$ fonksiyonuna aritmetik olarak x girdisi ekleyerek kısa yol bağlantısı sunmaktadır. Böylece $f(x)$ yerine her katman sonucunda $f(x) + x$ kullanılarak değerlerin bir sonraki katmana kaybolmadan geçilmesi sağlanmış ve 152 katmana kadar çıkılabilmiştir [183].

4.5. Tekrarlayan Sinir Ağları

Geleneksel sinir ağlarında genellikle, her girdi ve çıktının diğerlerinden bağımsız olduğunu varsayılmaktadır. Fakat bu varsayım, pek çok problemde doğru sonuç vermemektedir. Örneğin, bir cümleden sonra gelen kelime tahmin edilmek istense, cümlenin öncesinde kullanılan kelimeleri bilmek kesinlikle önemli olacaktır. Girdideki sıralı bilgiyi kullanmayı hedefleyen Tekrarlayan Sinir ağları (Recurrent Neural Networks-RNN) bir girdi dizisinin tüm elemanları için aynı hesaplamaları yapar ve her çıktı eleman mevcut girdiye ek olarak önceki tüm hesaplamalara bağlıdır. Bu sinir ağları bu nedenle tekrar eden olarak adlandırılır.



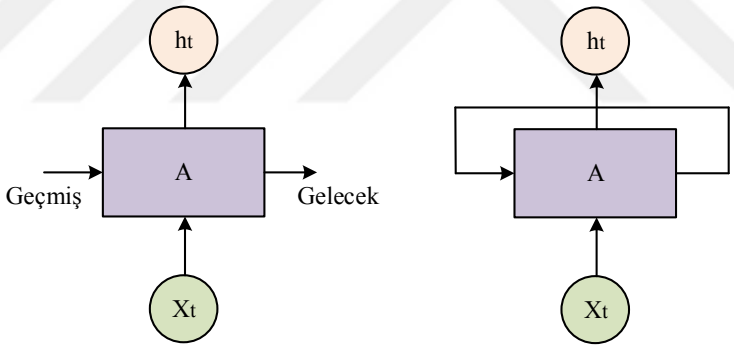
Şekil 4. 15. RNN dizilerinden bazı örnekler

Geleneksel sinir ağlarının bir diğer sınırlaması ise, girdi olarak sabit uzunluklu bir vektör alması ve yine çıktı olarak sabit uzunlu bir vektör üretmesidir. Örneğin bir haber sınıflandırma görevinde, her bir haber metni sayısal bir vektöre dönüştürülür ve ağın çıktısı sınıf olasılığını gösteren bir vektör üretir. Tekrarlayan Sinir Ağları bu noktada yalnızca sabit girdi ve çıktı

uzunluğu arasında bire-bir ilişki oluşturmamızı değil, aynı zamanda birden-çok, çoktan-bir ve çoktan-çok ilişkiler oluşturabilecek bir dizi vektörü (sequences of vectors) girdi olarak alabilmektedir. Tekrarlayan Sinir Ağları ile metin, stok değeri ve ses gibi çoklu veri dizilimleri (sequences of data) paralel bir şekilde eğitilmektedir. Uygulama alanlarına göre kullanılabilecek farklı RNN örnekleri Şekil 4.15'te verilmiştir [195]. Burada dikdörtgen çubuk bir vektörü ve ok işareti matris çarpımı gibi bir fonksiyonu ifade etmektedir.

4.5.1. RNN Mimarisi

İnsanlar geçmiş ile gelecek bilgi arasında ilişki kurabilen kalıcı bir belleğe sahiptir. İnsan beynini taklit etmeyi amaçlayan geleneksel sinir ağları ise geçmiş bilgileri ihmal etmektedir. Bu problemi çözmek için geliştirilen Şekil 4.16'daki RNN mimarisi, geçmiş ile ilgili bilgiyi saklayan ve verilen herhangi bir noktadaki gelecek tahminini yapmak için önceden öğrenilmiş bu bilgiyi kullanan döngülere sahiptir.

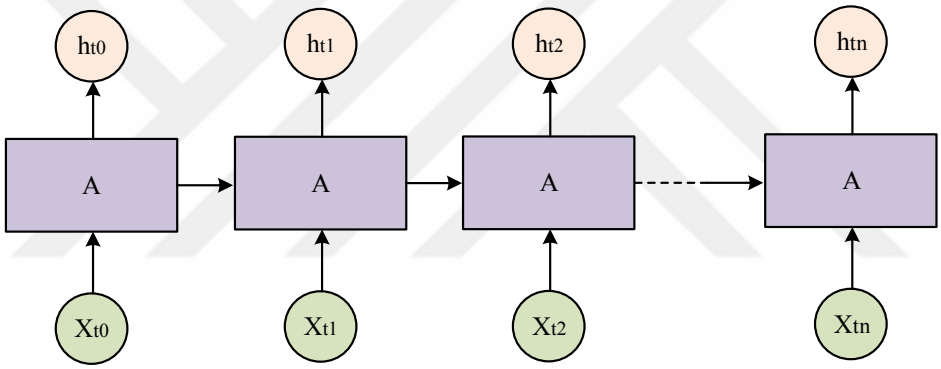


Şekil 4. 16. RNN mimarisi

Şekil 4.16'daki RNN döngüsüne bakıldığında A ağının, t zamanda X_t girdisi alıp h_t çıktısı ürettiği görülmektedir. RNN mimarisi bir zaman boyutunda tekrar eden bir hücreler zinciri olarak düşünüldüğünde RNN' in her bir kopyası bir mesajı diğerine iletmektedir. Aynı RNN mimarisinin bu açık (unrolled) gösterimi, Şekil 4.17'de verilmiştir. Matematiksel ifade ile A

nöronu, t_0 zaman adımında X_{t_0} girdisi alıp h_{t_0} çıktısı üretir. Daha sonra t_1 zaman adımında, aynı nöron X_{t_1} girdisini ve bir önceki zaman adımındaki sinyali alıp h_{t_1} çıktısı üretir. Bu şekilde t_n zaman adımına kadar aynı işlem tekrarlanır.

Örnek bir RNN senaryosunda, modelin girdisi *"bu filme gitmek için sabırsızlanıyorum…"* metnini içeren bir film yorumu olsun. Öncelikle modele *"bu"* kelimesi girdi olarak alınacak ve model çıktı olarak durum vektörü ve çıktı vektörü üretecek. Durum vektörü yorumdaki bir sonraki *"filme"* kelimesi ile birlikte modele girdi olarak verilecek ve yeni bir durum vektörü üretilecek. Bu şekilde son dizilime kadar model çıktı üretecektir. RNN mimarisi, geleneksel sinir ağları mimarilerinden farklı olarak zamanlar arasında bilgi aktarımı için bir *W* ağırlık matrisi tutmaktadır. Geçmiş dizilimlerin bilgisini içeren bir çeşit vektör durumunu güncelleyerek sıralı bilgiyi işlemektedir.



Şekil 4. 17. RNN mimarisinin açık gösterimi

RNN, bir dizilim üzerinde aynı fonksiyonu tekrarlı bir şekilde uygulamaktadır. Şekil 4.18 tipik bir RNN mimarisinin tekrarlı ilişkisini göstermektedir. RNN' de bir fonksiyonla tanımlanan bu tekrarlı ilişki şu denklemlerle ifade edilmektedir:

$$S_t = f(S^{t-1}, X_t) \tag{4.25}$$

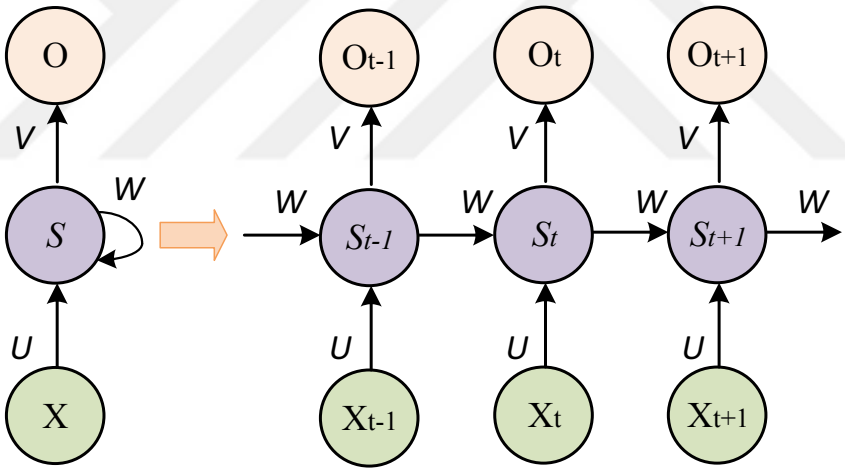
Burada S_t, t adımdaki durumu ifade eder. S_t durumu, bir önceki $t-1$ zaman adımındaki durum ile mevcut t zaman adımındaki X_t girdisinin f fonksiyonunun alınması ile bulunur. Daha açık matematiksel ifade ile S_t durumu Denklem 4.24'te verilmektedir:

$$S_t = S_{t-1} * W + X_t * U \tag{4.26a}$$

$$S_t = \tanh(S_{t-1} * W + X_t * U) \tag{4.26b}$$

$$O_t = V * S_t \tag{4.26c}$$

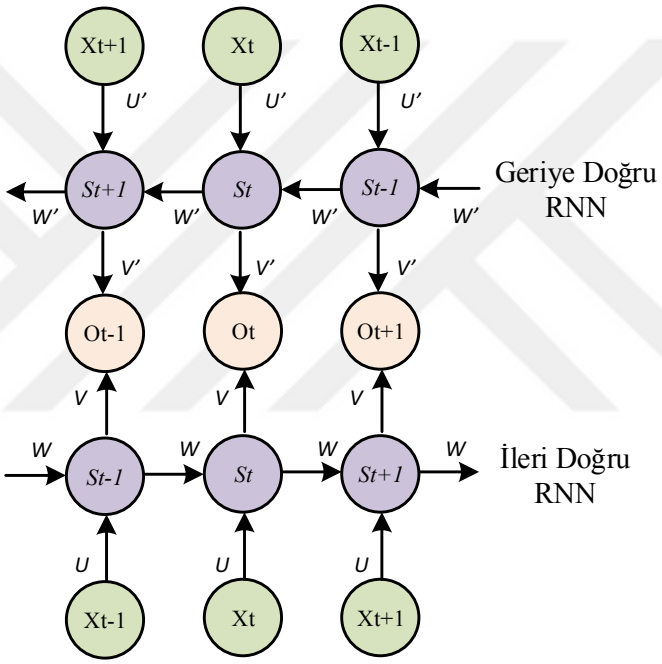
Durumdan duruma doğrusal bir geçişi temsil eden W değeri, bir önceki S_{t-1} durumu ile çarpımına; girdiden duruma doğrusal bir geçişi gösteren U değeri, mevcut t zaman adımındaki x_t girdisi ile çarpımı eklenmektedir. Burada, Denklem 1'deki f fonksiyonu yerine tanh fonksiyonu kullanılmıştır. İstenildiği takdirde ReLU gibi diğer aktivasyon fonksiyonları da kullanılabilmektedir. Ağın çıktısı O_t ile gösterilmiştir.



Şekil 4. 18. RNN mimarisinin tekrarlı ilişki denkleminin görsel gösterimi

4.5.2. İki Yönlü RNN Mimarisi

Tekrarlı sinir ağları, sadece geçmiş bilgileri kullanarak t zamanda sıralı bilgiyi yakalamayı hedeflemektedir. İki yönlü RNN ise, hem pozitif zaman yönünde yani $t-1$ zamandan $t+1$ zamana doğru ileriye hareket eden, hem de tersi yön yani negatif zaman yönünde $t+1$ zamandan $t-1$ zamana doğru dizilimin başlangıcına ilerleyen iki tane RNN katmanından oluşmaktadır. Geçmiş ve gelecek içeriğine sahip veri ile bir dizilim üzerinde tahmin yapmayı hedefleyen bu model Şekil 4.19'da sunulmuştur.



Şekil 4. 19. İki yönlü RNN mimarisi

4.5.3. Zaman içinde Geriye Yayılım

Geriye yayılım algoritmasına dayanan zaman içinde geriye yayılım (backpropagation through time-BBT) [196], belirli bir zaman adımı içinde gerçekleşen tekrarlayan ağları eğitmek

için kullanılan bir algoritmadır. X_t mevcut girdi, S_{t-1} bir önceki durum, U ve V her katmanda paylaşılan parametreler ise ağın gradyanı ve parametrelerin gradyanı aşağıdaki denklemlerde verilmiştir:

$$\frac{\delta E}{\delta S_{t-1}} = \frac{\delta E}{\delta S_t}\frac{\delta S_t}{\delta S_{t-1}} = \frac{\delta E}{\delta S_t}W$$

$$\frac{\delta E}{\delta U} = \sum\nolimits_{\text{i=0}}^{n} \frac{\delta E}{\delta S_t} X_t$$

$$\frac{\delta E}{\delta W} = \sum\nolimits_{\text{i=0}}^{n} \frac{\delta E}{\delta S_t} S_{t-1} \tag{4.27}$$

4.6. Uzun Kısa Süreli Bellek Ağları (LSTM)

4.6.1. Sıfırlanan Gradyan Problemi

Derin öğrenme modellerinin eğitiminde hataların kullanımı yani geriye yayılma kavramı ilk ortaya sürüldüğünde, her katmandan istatiksel olarak en iyi seçilen özellikler bir sonraki katmana çok yavaş bir şekilde ve otomatik olmadan iletiliyordu. 1970'lerde geriye yayılım algoritması [166] ile çok önemli çalışmalar gerçekleştirilse de bu algoritma 1985 yılına kadar yapay sinir ağlarına uygulanamadı. 1986 yılında, Hinton ve arkadaşları [170] derin modeller olarak adlandırılan ikiden fazla gizli katman ile çok katmanlı sinir ağları sunarak bu alana ilgiyi yeniden canlandırdı.

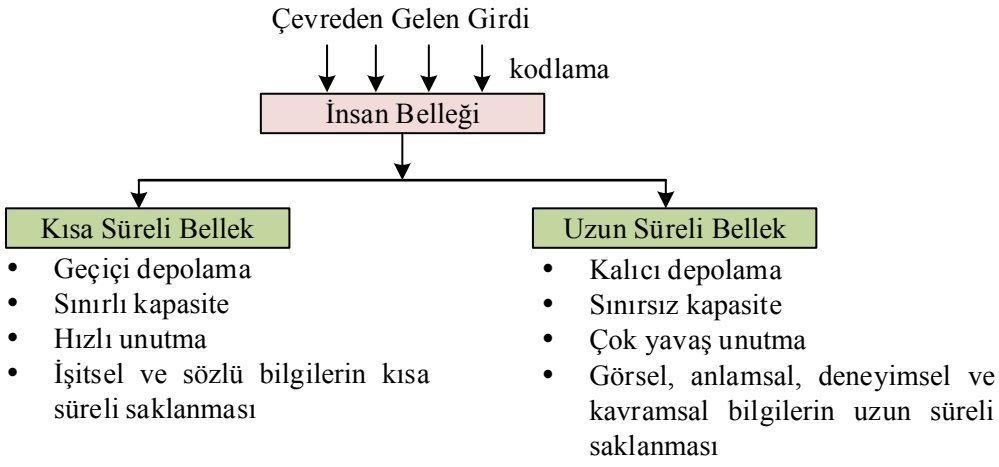
LeCun ve arkadaşları 1989 yılında [11], geriye yayılma öğrenme algoritmasını ilk defa bir gerçek dünya problemine uyguladı. El yazısı rakamlarını tanımak için kullandığı geriye yayılım algoritmasını konvolüsyonel sinir ağlarında test etti. Fakat çok katmanlı algılayıcıların öğrenmesi çok zordu ve çok iyi sonuçlar elde edemiyordu. Çok katmanlı ağların neden öğrenemediği sorusu ile sıfırlanan gradyan (vanishing gradient) problemi keşfedildi. Son

katmanda hesaplanan çok büyük hataların, yalnızca çok küçük bir kısmı bir önceki katmanlara aktarılıyordu yani önceki katmanlara öğrenme sinyalleri nerdeyse hiç ulaşamıyordu ve bu katmanlarda öğrenilen özellikler çok zayıf kalıyordu.

Geriye yayılım ile yapay sinir ağlarını eğitirken karşılaşılan en büyük problem önceki katmanları eğitmeyi ve ağı öğrenmeyi zorlaştıran sıfırlanan gradyan problemidir [197]. Sıfırlanan gradyan probleminde, önceki katmanların parametrelerine göre ağ çıktısı sıfıra yakın çok küçük gradyan değerleri üretir ve ağırlık sonucu önemli bir değişiklik göstermez. 1997 yılında, bu problemi çözmek için Hochreiter ve Schmidhuber [174], uzun kısa süreli bellek ağları yaklaşımını önermiştir.

4.6.2. İnsan Belleğinin Modellenmesi

Hafıza (bellek), geçmiş deneyimler hakkında edinilen, toplanan, kaydedilen ya da kodlanan bilgileri yeniden eğiten, işleyen, depolayan ve kullanılabilir hale getiren süreçler bütünüdür. Amerikalı psikolog William James [198], 1890 yılında insan hafızasını kısa süreli (birincil bellek ya da çalışan bellek) ve uzun süreli bellek (ikincil bellek) olarak ilk kez ikiye ayırmıştır. Şekil 4.20' de bellek olarak adlandırdığımız insan hafızasının sınıflandırma modeli gösterilmiştir.

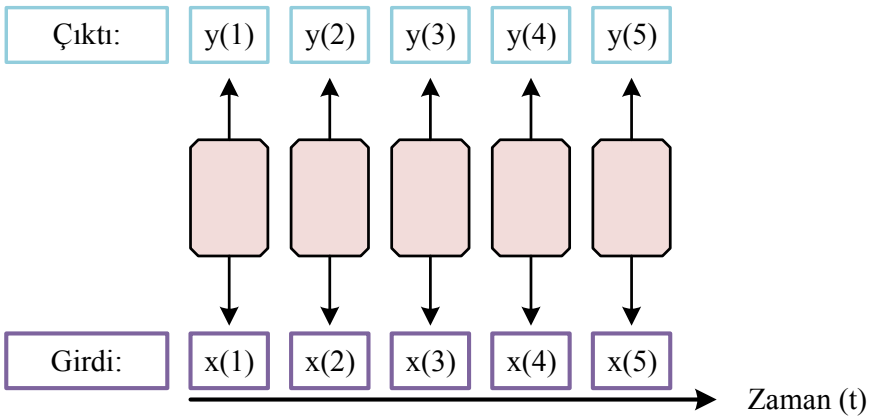


Şekil 4. 20. İnsan belleğinin kısa-süreli ve uzun-süreli olarak sınıflandırılması

Kısa-süreli bellek, duyma, görme, dokunma, tat alma gibi duyulardan gelen tüm bilgileri bir saniye ya da birkaç dakika gibi çok kısa bir zaman periyodunda tutan ve bu bilgileri unutan ya da uzun süreli belleğe gönderen bir birimdir. Örneğin bir telefon rehberine bakıp arama yapmak için bakılan numaranın kısa süreli hafızada tutulması gibi. Çalışan bellek olarak da adlandırılan kısa süreli bellek, öğrenme ve anlama gibi karmaşık görevleri gerçekleştirebilmektedir. Uzun süreli hafıza ise haftalar, aylar hatta bir ömür sürebilen uzun bir zamanda periyodunda bilgileri tutan ve geçmişe dönük bu bilgileri işleyen bir bellek birimidir. Burada kısa süreli bellekten alınan tüm bilgilere saklanmaktadır. İlkokuldaki sınıf öğretmeninin hatırlanması, Fransa'nın başkentinin bilinmesi, nasıl bisiklet kullanacağının bilinmesi, sevilen bir yemek ismi geçtiğinde istenmesi, yeni duyulan bir kelime ile diğer kelimeler arasında ilişki kurulması gibi anlamsal, kavramsal ve deneyimsel pek çok hafızayı içermektedir.

4.6.3. Sıralı Öğrenme

Zamana bağlı olarak değişen ve birbiriyle ilişkili olan her şey bir sekansı diğer bir ismi ile dizilimi (sequence) temsil etmektedir [199]. Örneğin bir video, görüntüler dizilimi ve bir metin cümleler ya da kelimeler dizilimidir. Örneğin bir video, görüntüler dizilimi ve bir metin cümleler ya da kelimeler dizilimidir.



Şekil 4. 21. Zaman serileri verisi olarak düşünülen sıralı öğrenme örneği

Şekil 4.21'de zaman serisi verilerinde sıralı öğrenme örneği sunulmuştur. Burada zaman eksenine göre x ve y dizilimleri gösterilmektedir. Bir $(x^1, x^2 \ldots x^n)$ diziliminde n değeri dizilim uzunluğunu gösterirken, belirli bir t zamana ait her bir örnek x girdi verisi zaman serileri verisi olarak düşünülmektedir.

Video, konuşma ve metin gibi veri dizilimi üzerinde makine öğrenimi gerçekleştirmek için geleneksel sinir ağları kullanılabilmektedir. Ancak verilerin girdi boyutunun sabit olması kısıtlayıcı bir problemdir. Her bir zaman adımında, rastgele uzunluktaki veri dizilimlerini besleyebilen ve geçmişte gerçekleşen önemli olayları hatırlayabilmek için bir çeşit hafıza belleğine sahip bir ağa ihtiyaç duyulmaktadır. Bu ihtiyaçlar ve problemler, farklı çeşitlerde tekrarlayan sinir ağlarının oluşumuna sebep olmuştur [200].

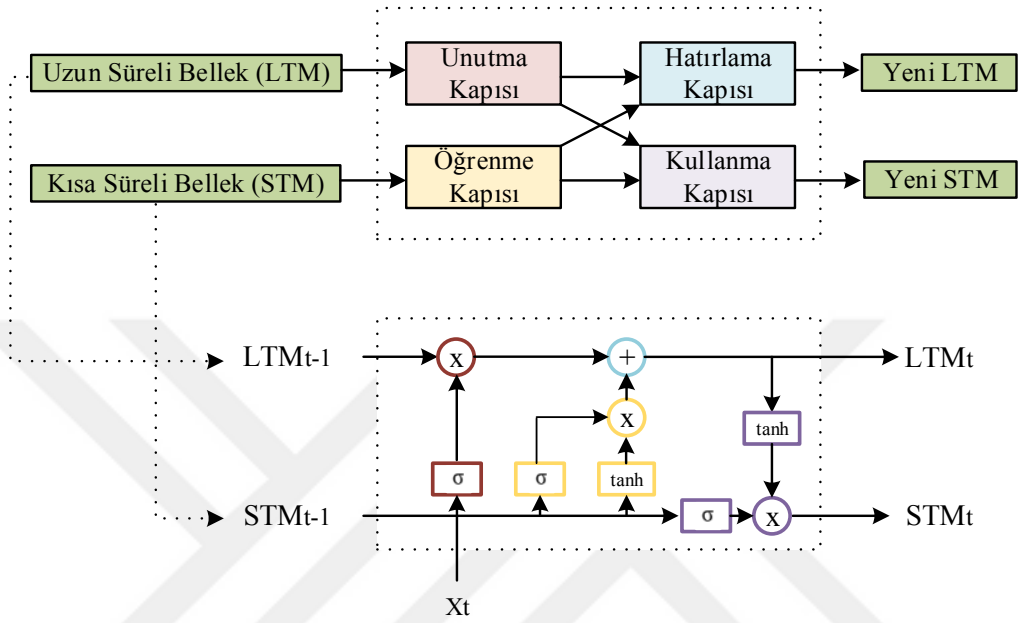
4.6.4. Sıralı Öğrenmeye dayalı LSTM Mimarisi

Sıralı öğrenmede amaç, kısa süreli ve uzun süreli belleği modellemektir. Kısa süreli bellek, standart RNN tarafından çok iyi bir şekilde modellenir ancak gradyan hesaplamalarının belirli bir zaman adımından sonra hızlıca sıfırlanması ya da sonsuza gitmesi problemleri nedeniyle, uzun süreli bağımlılıklarda etkili olamamaktadır.

RNN modellerin bir diğer kısıtlaması ise yalnızca mevcut girdileri hesaba almasıdır. Özellikle metinlerde bir cümlenin içeriğinin anlaşılması için uzun süreli bağımlıklara da ihtiyaç duyulmaktadır. Örneğin "yıldızlar gökyüzünde parlıyor" cümlesinin son kelimesini tahmin etmek RNN ile kolay olacaktır. Ancak daha uzun süreli bellek bağımlılığı gerektiren "Ahmet Türkiye'nin Elazığ şehrinde doğdu, ilkokulu Almanya'da ve liseyi Fransa'da okudu. Ahmet'in konuşabildiği diller…" cümlesinde RNN ile dizilimdeki önceki cümleleri hatırlamak zor olacaktır. Bu uzun-süreli bağımlılık probleminin çözümü için, uzun bir zaman periyodunda verilen girdi bilgisini hatırlayabilen ve hangi verilerin ne kadarını hatırlayabileceğine karar verebilen LSTM ağlarına ihtiyaç duyulmuştur.

LSTM mimarisinde, RNN mimarisinden farklı olarak uzun süre bir önceki girdiyi hatırlamak için kullanılan ve hafıza hücreleri (memory cells) olarak adlandırılan özel gizli birimler bulunmaktadır. Bir LSTM mimarisi, bir girdinin kaydedilebilecek kadar önemli olup

olmadığını belirleyen kapılar içerir. Bu mimari Şekil 4.22'de gösterildiği gibi unutma (forget), öğrenme (learn), hatırlama (remember) ve kullanma (use) kapıları olmak üzere dört birimden oluşmaktadır [201]:



Şekil 4. 22. LSTM mimarisinin teorik (üstteki) ve matematiksel (alttaki) gösterimi

- ***Öğrenme Kapısı*:** Öğrenme kapısı, kısa süreli bellek tarafından alınan girdi bilgilerin bir kısmını unuturken önemli olan kısmını da saklamaktadır. Matematiksel olarak h_{t-1} kısa süreli bellek değeri ile x_t girdisinin doğrusal fonksiyonu W_n ağırlık matrisi ile çarpılıp, b_n bias değeri eklenir. Son olarak, $tanh$ aktivasyon fonksiyonu ile sonuç n_t değeri hesaplanır. Burada n_t yeni değeri göstermektedir. Gelen bilginin bir kısmını unutmak yani ihmal etmek için i_t ihmal faktörü kullanılmaktadır. Bu değerin alınması için kısa süreli belleğin önceki bilgisi ve girdiler kullanılmaktadır. Oluşturulan bu yeni matris değerine σ sigmoid fonksiyonu uygulanarak i_t faktörü [0,1] arasında tutulmaktadır. Öğrenme kapısının çıktısı olarak gösterilecek olan $n_t i_t$ değeri aşağıdaki denklemde hesaplanmıştır:

$$n_t = \tanh\left(W_n[h_{t-1}, x_t] + b_n\right) \quad (4.28a)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (4.28b)$$

- **Unutma *Kapısı:*** Unutma kapısı, uzun süreli belleği alır ve hangi kısımlarının tutulup hangilerinin unutulacağına karar verir. Matematiksel olarak, unutma kapısının çıktısı olan $C_{t-1}f_t$ değeri aşağıdaki denklemde hesaplanmıştır. Burada f_t unutma faktörüdür.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (4.29)$$

- **Hatırlama *Kapısı:*** Hatırlama kapısının çıktısı olan yeni C_t uzun süreli bellek değeri, $n_t i_t$ öğrenme kapısının çıktısı ve $C_{t-1}f_t$ unutma kapısının çıktısının toplamı ile aşağıdaki denklemde gösterildiği gibi elde edilmektedir. Burada f_t , n_t ve i_t değerleri bir önceki Denklem 4.26a, 4.26b ve 4.27'de verilmiştir.

$$C_t = C_{t-1}f_t + n_t i_t \quad (4.30)$$

- **Kullanma *Kapısı:*** Çıktı kapısı olarak adlandırılan kullanma kapısı, unutma kapısı ve öğrenme kapısını kullanarak yeni bir kısa süreli bellek ve çıktı üretmektedir. Matematiksel olarak kullanma kapısının çıktısı olan $U_t V_t$ değeri aşağıdaki denklemde hesaplanmıştır:

$$U_t = \tanh\left(W_u C_{t-1}f_t + b_u\right) \quad (4.31a)$$

$$V_t = \sigma(W_V[h_{t-1}, x_t] + b_V) \quad (4.32b)$$

Özet olarak LSTM mimarisinde dört kapı ve iki bellek bulunmaktadır. Unutma kapısı uzun süreli belleği alır ve bu belleğin bir parçasını unutur. Öğrenme kapısı daha önceden kısa süreli bellekten öğrenilen bilgiler ile birlikte girdiyi alır ve yeni bilgi öğrenir. Hatırlama kapısı, uzun süreli bellekten gelen (kaydedilen) bölümlere öğrenilen yeni bilgileri ekleyerek uzun süreli hafızayı günceller. Kullanma kapısı uzun süreli hafıza ile birlikte öğrenilen bilgiyi alır ve bir

tahmin yapmak ve kısa süreli belleği güncellemek için bu bilgiyi kullanır. LSTM biriminde, geriye yayılım sürecinde ağırlıkları güncellerken kolaylık sağlayan sigmoid (σ), hiperbolik tanjant ($tanh$), çarpım (x) ve toplam ($+$) olmak üzere dört faklı fonksiyon kullanılmaktadır. Bu yönüyle türevlenebilirdir, yani fonksiyonların gradyanı alınabilmektedir.

LSTM mimarisinin en temel avantajları sıfırlanan gradyan problemine çözüm oluşturması ve girdilerin unutulmadan saklanabilmesi yani bilgi kaybını engelleyebilmesidir. LSTM biriminde üç sigmoid bulunmaktadır ve her bir sigmoid çıktısı sıfır ve bir arasında değer almaktadır. Bir x girdisi için $\sigma(x) \approx 1$ ifadesi tüm verinin geçmesi, $\sigma(x) \approx 0$ ise verinin geçmemesi anlamını taşımaktadır. Böylelikle tek bir filtre ile hangi verinin hücreden geçeceği, hangisinin hücrede kalacağı ve hangisinin çıktıya ulaşacağına karar verilebilmektedir.

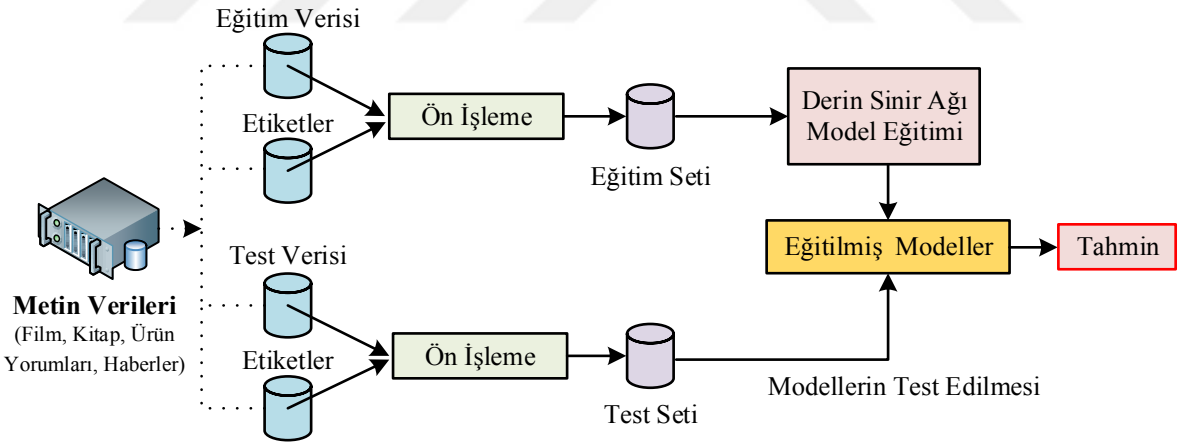
LSTM mimarisine göre daha az karmaşık hesaplama gerektiren diğer bir RNN çeşidi Kapılı Tekrarlı Birim (Gated Recurrent Unit-GRU) mimarisidir. Bu mimari, unutma ve öğrenme kapısını bir güncelleme kapısı adı altında birleştirir ve daha sonra bir sıfırlama (reset) kapısında çalıştırır. Çıktı olarak uzun ve kısa süreli bellek çifti yerine çalışan yeni bir bellek üretir. Güncelleme kapısı geçmiş bilgilerin ne kadarının bir sonraki gelecek adıma aktarılacağını belirler. Sıfıra yakınsa, önceki bellek dikkate alınmaz ve sadece yeni bilgiler saklanır. Sıfırlama kapısı, geçmiş bilgilerin ne kadarının unutulacağına karar verir. Özet olarak güncelleme ve sıfırlama kapılarını kullanarak bilgiyi saklar ya da filtreler. Böylelikle sıfırlanan gradyan problemine takılmadan bilgiyi uzun süre saklamak ya da tahminden uzak bilgiyi kaldırmak için eğitilebilmektedir.

5. DERİN ÖĞRENME MODELLERİNİN OLUŞTURULMASI VE EĞİTİLMESİ

Tez çalışması kapsamında metin analizleri gerçekleştirmek için oluşturulan derin sinir ağı modellerinin eğitim ve test aşamalarındaki genel iş akışı aşağıdaki gibidir:

- Verilerin toplanması, eğitim ve test verilerinin belirlenmesi
- Verilerin ön işleme süreçlerinden geçirilmesi
- Verilerin yüklenmesi
- Model seçimi ve model hiper parametrelerinin tanımlanması
- Hiper parametreler ve programlama adımları ile derin sinir ağı modelinin kurulması
- Eğitim verisi üzerinde modelin eğitilmesi
- Doğruluk ve hata sonuçlarının alınarak modelin değerlendirilmesi
- Eğitim ve test hata grafiklerinin yorumlanması, modelin iyileştirilmesi ve tahmin sonuçlarının alınması.

Yukarıdaki adımlar Şekil 5.1'de özetlenmiştir.



Şekil 5. 1. Metin verileri üzerinde derin sinir ağı modellerinin eğitim ve test aşamaları

5.1. Metinlerin Vektör Gösterimi

Geleneksel makine öğrenmesi modellerinde olduğu gibi derin öğrenme modelleri de eğitilmeden önce verilen metinlerin sayısal gösterime dönüştürülmesi gerekmektedir. Metni sayısal temsile dönüştürme işlemi, vektörleştirme (vectorization) olarak adlandırılır ve bu işlem birkaç farklı şekillerde yapılabilir:

- Metni kelimelere dönüştürme ve her kelimeyi bir vektör olarak gösterme
- Metni karakterlere dönüştürme ve her karakteri bir vektör olarak gösterme
- Kelimelerin n-gramını oluşturma ve bunları vektörler ile temsil etme

Verilen bir cümlenin karakterlere ya da kelimelere ayrılmasına birim (token) adı verilir. Metin verileri küçük birimlere ayrıldıktan sonra her bir birim bir vektörle eşleştirilir. Birimlerin vektörler ile eşleşmesi için iki temel yaklaşım mevcuttur: 1-hot kodlama (one hot encoding) ve kelime temsili (word embedding). Bu tez çalışmasında kelime temsilleri oluşturmak için word2vec algoritması tercih edilmiştir. Bu algoritmanın avantajları aşağıda verilmiştir.

- Basit, anlaması kolay ve güçlü bir mimari sunmaktadır. Diğer teknikler ile kıyaslandığında eğitimi hızlıdır.
- Verilerin etiketlenmesine ihtiyaç duymadan eğitimi için minimum düzeyde çaba gerektirir.
- Hem çok küçük veri setlerinde hem de çok büyük veri setlerinde çalışabildiği için ölçeklenebilirdir.
- Anlamsal benzerliği yakalama konusunda çok başarılıdır.
- Gözetimsiz bir yaklaşım olduğundan dolayı insan emeği çok azdır. Bu yüzden zaman kazandıran bir tekniktir.

Word2vec, çok başarılı bir teknik olmasına rağmen aşağıda verilen bazı zorluk noktaları vardır:

- Oluşturulacak yeni bir veri seti için word2vec modelinin geliştirmesi kolay olsa da hata ayıklaması (debug) zorluğu en önemli problemlerden birisidir.
- Bir kelimenin birden fazla anlamı olduğu durumda, kelime temsili vektör uzayında bu anlamların ortalamasını alacaktır. Bu durumda oluşacak olan belirsizlik word2vec modelinin bir diğer zorluğudur.

5.2. Veri Seti ve Ön işleme Adımları

Sinir ağlarını öğrenme işlemi için üç tip veri setine ihtiyaç duyulmaktadır:

1. *Eğitim Seti (Train Set):* Film yorumları ve bu yorumların pozitif/negatif etiketleri gibi girdi verileri ve onların etiketlerini içeren veri setidir. Bu verileri öğrenme algoritması öğrenerek, daha sonradan etiketsiz verileri sınıflandırmak için kullanmaktadır. Genellikle veri setinin %70' ini oluşturmaktadır.
2. *Test Seti (Test Set)* ve *Doğrulama Seti (Validation Set):* Eğitim setinden öğrenen modellerin doğrulunu ya da performansını test ederek en iyi modeli seçmek için doğrulama seti kullanılmaktadır. Modelin öğrendiklerini, etiketlenmemiş metinlere veya genel olarak görülmeyen verilere uygulama yeteneğini test etmek ve iyi çalışıp çalışmadığını görmek için de test verileri kullanılmaktadır. Veri bilimcileri veri setini eğitim, doğruluma ve test seti olarak üçe ayırabildikleri gibi, sadece veri setini eğitim ve test olarak da ikiye ayırabilmektedir.

Bu tez kapsamında oluşturulan haberler ve film yorumları veri setleri üzerinde word2vec algoritmasını ya da modelini uygulamadan önce veri seti temizleme işlemleri için aşağıdaki hususlar dikkate alınmıştır:

1. *Veri türü:* Word2vec modeli tüm metin verilerine uygulanabilmektedir. Bu nedenle kullanılmayacak özel bir metin verisi türü yoktur.
2. *Sık kullanılan kelimelerin kaldırılması:* Google tarafından İngilizce dili için geliştirilen word2vec modelde "a" gibi bazı sık kullanılan kelimeler kaldırılmasına rağmen, "the"

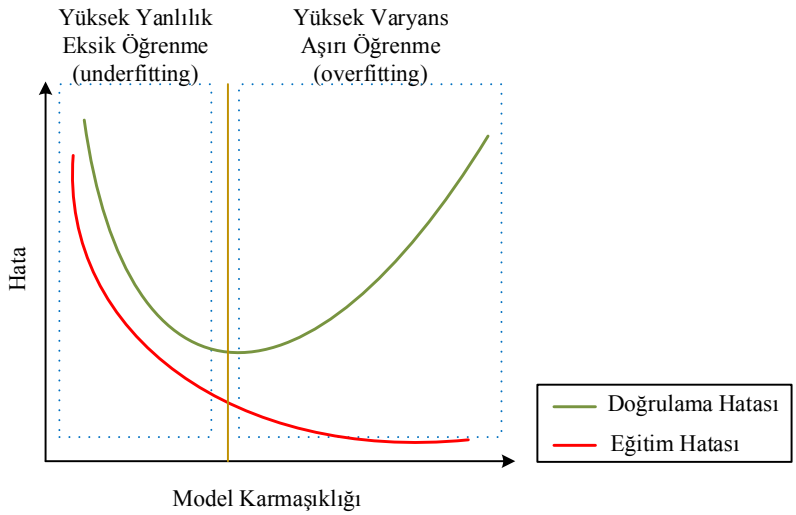
kelimesi gibi çok sık geçen kelimeler kaldırılmamıştır. Bu durum yapılacak NLP uygulamasına bağlı olarak değişkenlik göstermektedir. Örneğin duygu analizi çalışması için sık geçen kelimelerin kaldırılmasında bir sakınca yokken makine çevrimi ve dilin gramer yapısını anlamaya yönelik çalışmalarda kaldırılmaması gerekmektedir. Bu tez çalışması kapsamında hazırlanan Türkçe veri setinde herhangi bir sık geçen kelime kaldırılmamıştır.

3. *Diğer ön işleme adımları:* Veri setindeki tüm sayılar, noktalama işaretleri ve simgeler kaldırılmıştır. Ayrıca tüm harfler küçük harflere dönüştürülmüştür. Birbirinden bağımsız metinler (yorumlar, haberler gibi) farklı satırlara ayrılmıştır.

5.3. Derin Öğrenmede Yanlılık/Varyans İkilemi

Diğer makine öğrenmesi modelleri ile karşılaştırıldığında, derin sinir ağları optimizasyon için gerekli hiper parametrelerin yanı sıra milyonlarca öğrenilebilir parametre ile çok daha karmaşıktır. Bu durum geleneksel makine öğrenmesi modellerinde olduğu gibi yanlılık/varyans ikilemini (bias/variance trade-off) [202] ortaya çıkarabilmektedir. Derin öğrenmede Şekil 5.2'de gösterilen yüksek yanlılık ve varyans hatalarını kontrol etmek için aşağıdaki çözümler uygulanmaktadır [190] :

- *Yanlılık Hatası*: Model ile ilişkili bir hatadır. Doğrusal olmayan bir fonksiyon doğrusal bir model ile modellenirse, model beklenenin altında kalır ve yanlılık hatası yüksek olur. Yanlılık hatası yüksek olduğunda, eğitim verisi eksik öğrenilir. Kısaca bu durum eksik öğrenme (underfitting) olarak adlandırılır. Yüksek yanlılık hatası olan bir ağ, eğitim seti üzerinde tahmin çıktıları üretirken yüksek hata oranı çıkar ve model veriyi iyi bir şekilde öğrenemez. Yanlılık hatasını azaltmak için yeni katmanlar ve nöronlar ekleyerek ağ mimarisinin değiştirilmesi gereklidir. CNN ve RNN yöntemleri bu duruma iyi bir çözüm olabilmektedir.



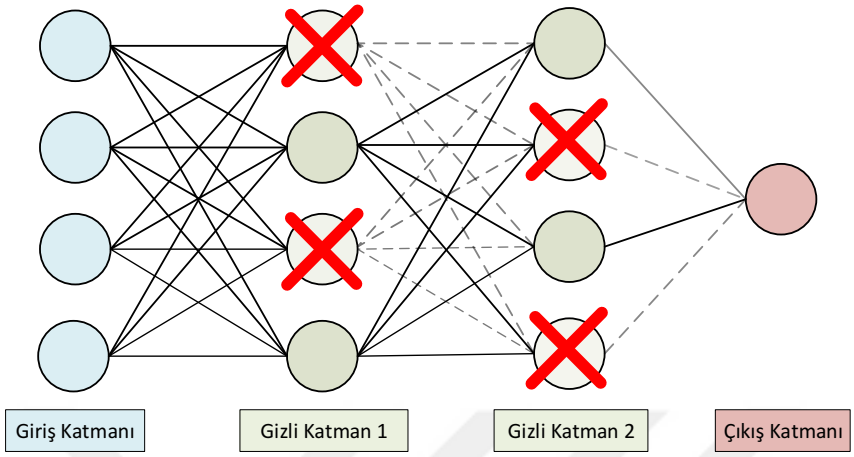
Şekil 5. 2. Yanlılık ve varyans hatalarının gösterimi

- *Varyans Hatası:* Varyans hatası, eğitim verisindeki rassallık sebebiyle ortaya çıkan hatadır. Eğitim dağılımı modelin artık genelleşemeyeceği kadar iyileştirildiğinde, yüksek varyans hatası ortaya çıkmış olur. Bu durum aşırı öğrenme (overfitting) olarak adlandırılır. Varyans hatası yüksek olan bir ağ, eğitim verisini çok iyi öğrenmiştir ve veriyi ezberleme problemi ile karşı karşıya kalmıştır. Doğrulama hatasının eğitim hatasından ayrılarak arttığı noktada, aşırı öğrenme problemi oluşmaya başlar. Varyans hatasını azaltmak için veri sayısının arttırılması, dropout ve düzenlileştirme (regularization) tekniklerinin uygulanması çözüm olabilmektedir.

5.3.4. Dropout ile Aşırı Öğrenme Kontrolü

Derin sinir ağlarında aşırı öğrenmeyi engellemek için kullanılan dropout tekniği, kabaca ağın gizli katmanlarındaki bazı nöronların bırakılma (drop-out) yaklaşımıdır. Her bir mini-batch boyutunda, her bir gizli katmandaki düğümler yani nöronlor rastgele seçilerek pasif hale getirilir. Droput değerinin 0.5 olarak ayarlandığı Şekil 5.3'deki gibi basit bir derin sinir ağında, her bir gizli katmandaki nöronların yaklaşık yarısı bırakılacak yani ağ içinde kullanılmayacaktır. Burada, bir devirde (epoch 1) bir mini-batch için birinci gizli katmanda rastgele seçilen 1. ve 3.

düğüm, ikinci gizli katmanda ise 2. ve 4. düğüm kapatılmıştır. Dolayısıyla tüm sinir ağının ağırlıkları da yaklaşık olarak yarıya inmiştir.



Şekil 5. 3. Dropout ile aşırı öğrenme/varyans kontrolü

Dropout tekniği ile daha az ağırlık ile daha küçük sinir ağları eğitilir ve her bir küçültülmüş ağın veriyi aşırı öğrenme ihtimali azalır. Ayrıca her nöronun diğer tüm nöronlara tam bağlı olma durumu ortadan kalkar.

5.3.5. Düzenlileştirme ile Aşırı Öğrenme Kontrolü

Derin sinir ağlarında aşırı öğrenmeyi kontrol etmek için diğer bir yol olan düzenlileştirme tekniği, model büyüdükçe modeldeki bazı ağırlıkları cezalandırma yaklaşımıdır. Sinir ağlarını düzenlemek için L1 ve L2 olmak üzere iki çeşit düzenlileştirme yöntemi vardır. Fakat L2 düzenlileştirme tekniği, hesaplama verimliliği açısından daha avantajlı olduğu için derin sinir ağlarında daha sık kullanılmaktadır.

Düzenlileştirme yaklaşımı için hata fonksiyonunun düzenlenmesi gerekmektedir. Bunun için uygulanan düzenlileştirme miktarının değişimini arttırabilen ya da azaltabilen λ parametresi kullanılmaktadır. L1 ve L2 düzenlileştirme, sırasıyla Denklem 5.1 ve Denklem 5.2'de verilmiştir:

$$E = -\frac{1}{m}\sum_{i=1}^{m}(y_i \times \ln(\hat{y}_i)) + ((1-y_i) \times \ln(1-\hat{y}_i)) + \boldsymbol{\lambda}(|\boldsymbol{w_1}| + \cdots + |\boldsymbol{w_2}|) \quad (5.1)$$

$$E = -\frac{1}{m}\sum_{i=1}^{m}(y_i \times \ln(\hat{y}_i)) + ((1-y_i) \times \ln(1-\hat{y}_i)) + \boldsymbol{\lambda}(\boldsymbol{w_1^2} + \cdots + \boldsymbol{w_2^2}) \quad (5.2)$$

Daha önceki bölümde anlatılan hata fonksiyonuna **λ** parametresi eklenen her iki denklemden de görüleceği gibi, ağırlık değerleri büyükse hata değeri artacaktır. Burada **λ** parametresi, ağırlık katsayılarının ne kadar cezalandırılacağını belirlemektedir. L1 düzenlileştirme, yüzlerce özellik arasından en iyi özelliklerin seçiminde iyi bir performans sergilemektedir. Diğer yandan L2 düzenlileştirme sinir ağı modellerinin eğitiminde daha iyi sonuçlar vermektedir. Bu nedenle derin sinir ağı modellerinde daha sık kullanılmaktadır.

5.4. Derin Öğrenme Modelleri için Hiper Parametre Ayarlamaları ve Seçimleri

Hiper parametreler ile ilgili bilinen en önemli zorluk veri setine ve problem ihtiyacına bağlı olarak değişmesidir, her modelde iyi bir sonuç verecek sihirli değerlerin olmamasıdır. Hiper parametreler genel olarak iki kategoriye ayrılır. İlki model hiper parametreleridir. Bu parametreler, modelin yapısında yer alan değişkenlerdir. Aktivasyon fonksiyonları, dropout, gizli katmanların sayısı ve modele özel hiper parametreler bu kategoridedir. İkincisi iyileştirici (optimizer) hiper parametrelerdir. Modelin yapısından ziyade eğitim ve en iyileme anlamına gelen optimizasyon süreçlerinde yer alan değişkenlerdir. Bu kategori de öğrenme katsayısı (learning rate), mini-batch boyutu ve tüm eğitim seti üzerindeki devir sayısını (epoch) içermektedir.

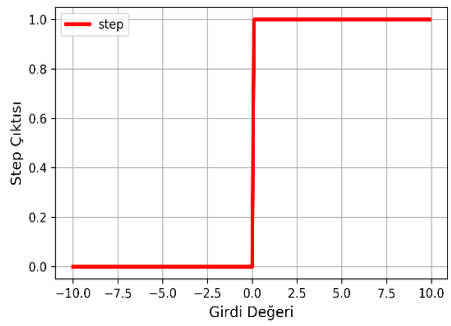
5.4.1. Aktivasyon Fonksiyonları

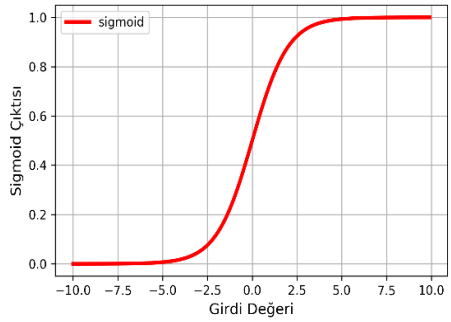
Bir sinir ağına doğrusal olmayan özellikler kazandıran aktivasyon fonksiyonlarının temel amacı, bir ağ içindeki düğümün girdi sinyalini bir çıktı sinyaline dönüştürmektir. Bir yapay sinir

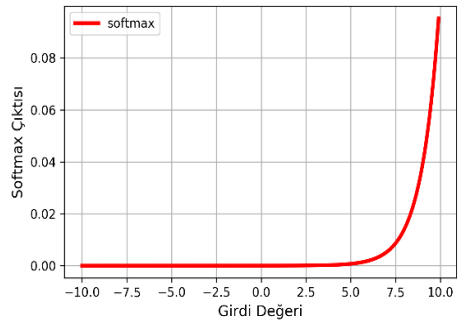
ağında x girdileri ve bu girdilerin ilişkili olduğu w ağırlıkları toplanır ve bu sonuca $f(x)$ aktivasyon fonksiyonu uygulanarak ilgili katmanın çıktısı alınır ve bir sonraki katmana girdi olarak verilir. Aktivasyon fonksiyonu olmadan metin verilerinin öğrenilmesi ve modellenmesi mümkün değildir. Çünkü türevlenebilir doğrusal bir fonksiyon ile ancak geriye yayılım gerçekleşerek ağırlık güncellemeleri yapılmaktadır. Bu nedenle kullanılacak olan aktivasyon fonksiyonunun seçimi önemlidir. Daha önce geleneksel sinir ağlarında anlatılan aktivasyon fonksiyonları ve popüler derin sinir ağlarında kullanılan en yeni aktivasyon fonksiyonları denklemleri ve Python uygulama kodları ile birlikte Tablo 5.1'de sunulmuştur. Açıklanan tüm aktivasyon fonksiyonlarının grafiksel gösterimi Şekil 5.4'te verilmiştir.

Tablo 5. 1. Aktivasyon fonksiyonları

Fonksiyon Grafiği	**Fonksiyon Açıklamaları**

	• Fonksiyon İsmi: Step
	• Fonksiyonu denklemi:
	$f(x) = \begin{cases} 1 \text{ eğer } x \geq 0 \\ 0 \text{ eğer } x < 0 \end{cases}$
	• Python dilinde fonksiyon kodu:
	`def step(x):`
	`  return np.array(x>0, dtype=np.int)`

	• Fonksiyon İsmi: Sigmoid
	• Fonksiyonu denklemi:
	$f(x) = \frac{1}{1 + e^{-x}}$
	• Python dilinde fonksiyon kodu:
	`def sigmoid(x):`
	`  return 1 / (1 + np.exp(-x))`

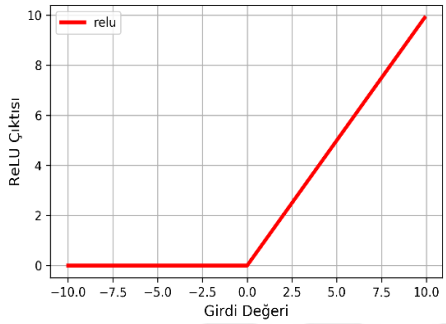


- Fonksiyon İsmi: Softmax
- Fonksiyonu denklemi:

$$f(x) = \frac{e_i^x}{\sum_j e_j^x}$$

- Python dilinde fonksiyon kodu:

```
def softmax(z):
    return np.exp(z) /
           np.sum(np.exp(z))
```

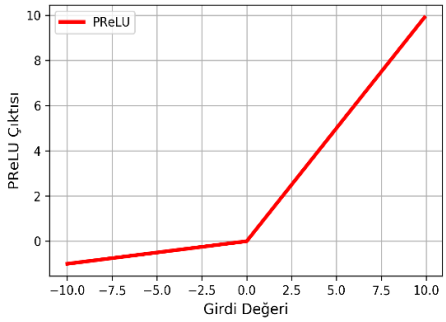


- Fonksiyon İsmi: ReLU
- Fonksiyonu denklemi:

$$f(x) = \begin{cases} x \text{ eğer } x \geq 0 \\ 0 \text{ eğer } x < 0 \end{cases}$$

- Python dilinde fonksiyon kodu:

```
def ReLU(x):
    return np.maximum(0, x)
```

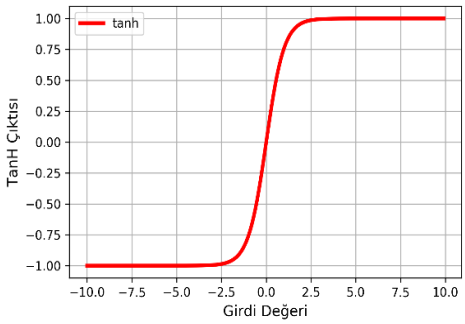


- Fonksiyon İsmi: Parametrik ReLU (PReLU) [204]
- Fonksiyonu denklemi:

$$f(x) = \begin{cases} x \text{ eğer } x \geq 0 \\ \alpha x \text{ eğer } x < 0 \end{cases}$$

- Python dilinde fonksiyon kodu:

```
def PReLu(x,alpha):
    if x <0:
        return alpha*x
    else:
        return x
```

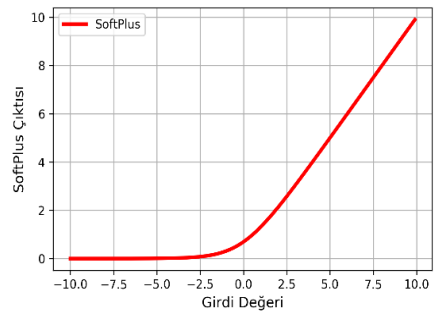


- Fonksiyon İsmi: TanH
- Fonksiyonu denklemi:

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

- Python dilinde fonksiyon kodu:

```
def tanh(x):
    return np.tanh(x)
```

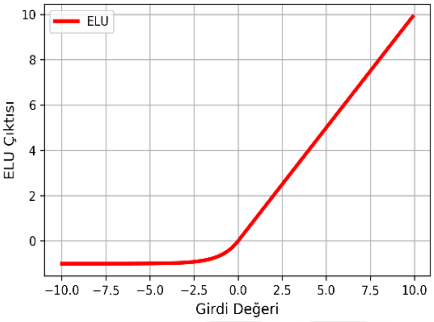


- Fonksiyon İsmi: SoftPlus
- Fonksiyonu denklemi:

$f(x) = log_e(1+e^x)$

- Python dilinde fonksiyon kodu:

```
def softplus(x):
    return np.log(1.0 + np.exp(x))
```

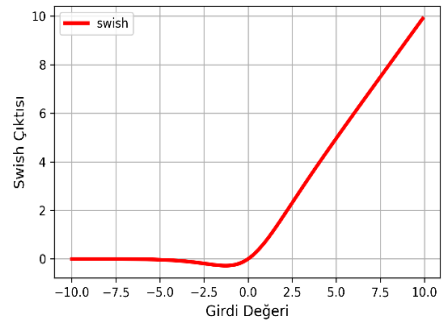


- Fonksiyon İsmi: ELU [203]
- Fonksiyonu denklemi:

$$f(x) = \begin{cases} x & eğer\ x \geq 0 \\ \alpha(e^x - 1) & eğer\ x < 0 \end{cases}$$

- Python dilinde fonksiyon kodu:

```
def elu(x,alpha):
  if x <0:
      return alpha*(np.exp(x)-1)
    else:
      return x
```

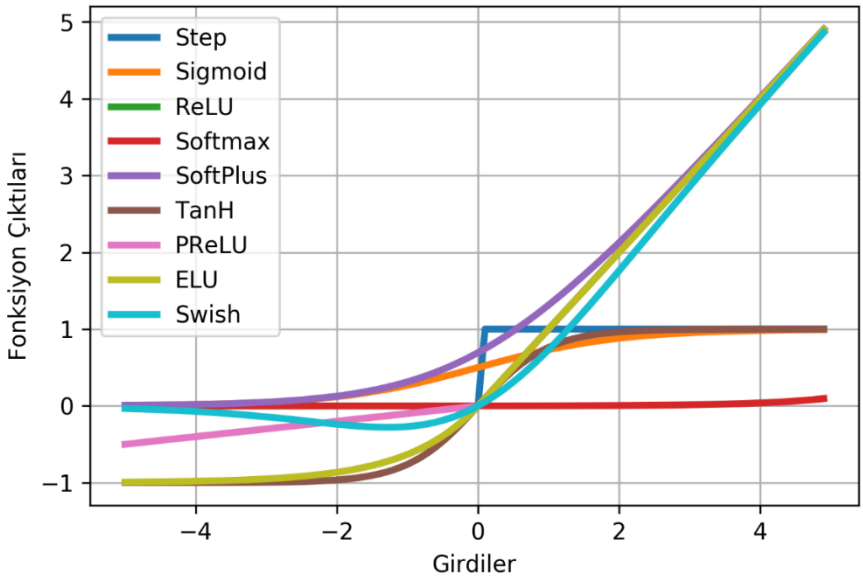


- Fonksiyon İsmi: Swish
- Fonksiyonu denklemi:

$f(x) = x \times sigmoid(x)$

- Python dilinde fonksiyon kodu:

```
def swish(x):
    return x*sigmoid(x)
```

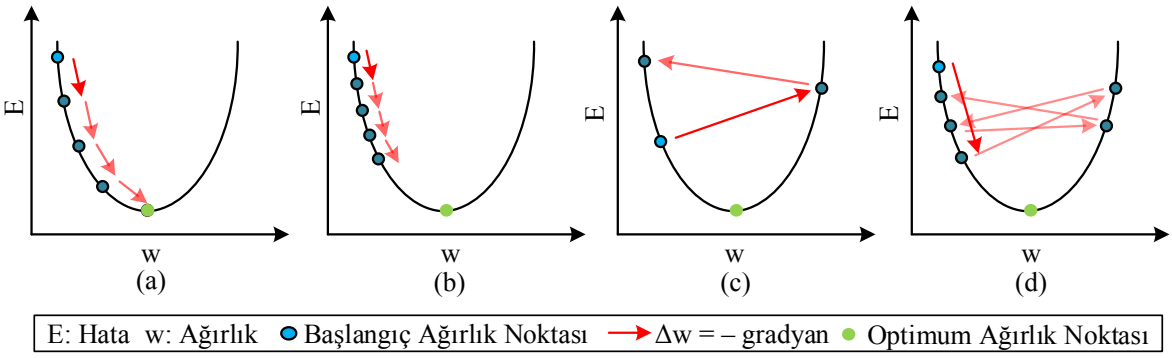


Şekil 5. 4. Sinir ağları için kullanılan aktivasyon fonksiyonları

5.4.2. Öğrenme Katsayısı

Yoshua Bengio [80], öğrenme katsayısının (learning rate) en önemli hiper parametre olduğunu ve her zaman değerinin iyi ayarlanmış olduğundan emin olunması gerektiğini belirtmiştir. Çok katmanlı sinir ağları için öğrenme katsayısı değerleri (0,1) aralığında 1'den küçük ve 10^{-6}'dan büyük alınmalıdır. Genellikle 0.01 öğrenme katsayısı değeri iyi bir başlangıç noktası sayılmaktadır. Bununla birlikte, eğitim hatasının davranışına bağlı olarak 0.001, 0.0001, 0.00001 gibi değerler de yaygın olarak kullanılmaktadır.

Öğrenme katsayısının belirlenmesinde kesin bir kural ve kısıtlamanın olmamasıyla birlikte, en iyi öğrenme katsayısı genellikle en az hata üreten en iyi değere sahiptir. Bu nedenle, modelin eğitim sürecindeki davranışını izleyerek en iyi değere ulaşmak mümkündür. En iyi değerlere ulaşmada yardımcı olacak bazı durumlar aşağıda verilmiştir ve bu maddelere karşılık gelen görsel anlatımı Şekil 5.5'te sunulmuştur:



Şekil 5. 5. Derin sinir ağlarında öğrenme katsayısının davranışları

- Eğitim sırasında modelin doğrulama hatası azalıyorsa (a), öğrenme katsayısı iyi seçilmiştir.
- Eğitim sırasında modelin doğrulama hatası çok yavaş azalıyorsa (b), öğrenme katsayısı kötüdür ve arttırılmalıdır.
- Eğitim sırasında modelin doğrulama hatası artıyorsa (c), öğrenme katsayısı kötüdür ve azaltılmalıdır.
- Eğitim sırasında modelin doğrulama hatası önce azalıyor sonra artıyor ve hata-ağırlık grafiğinde sürekli salınım gerçekleştirerek optimum ağırlık noktasına ulaşamıyorsa (d), eğitim sırasında öğrenme katsayısının zamanla yavaşça azaltılması gerekmektedir. Bu tekniğe öğrenme katsayısı azaltma (decay) denilmektedir. Bu teknikte öğrenme katsayısı ya doğrusal ya da üstel olarak azaltılır.

5.4.2. Mini-batch Boyutu

Modelin eğitiminde kullanılan hesaplama kaynaklarını, eğitim hızını, iterasyon sayısını ve hata hesaplamalarındaki gürültüleri etkileyen bir hiper parametredir. Mini-batch, tek seferde tüm verilerin eğitilmesi yerine veri setinin alt kümeleri üzerinde eğitim yapmak için kullanılan bir tekniktir. Bu yöntem, bir bilgisayarın, tüm veri setini depolamak için yeterli belleğe sahip olmaması durumunda bir model eğitme olanağı sağlamaktadır. Batch eğitimi için her devir başlangıcında veriler rastgele karıştırılır ve eğitim verileri daha küçük mini-batch' lere ayrılır.

Model her bir mini-batch üzerinde ileri besleme işlemi gerçekleştirir ve hata değerini hesaplar. Daha sonra her bir mini-batch üzerinde geriyayılım kullanılarak gradyan hesaplamaları gerçekleştirilir ve ağın ağırlıkları güncellenir. Ancak tüm örneklerde aynı anda hata hesaplanamayacağı için bu işlem hesapsal olarak verimsiz bir yöntemdir. Bu nedenle ağırlık güncellemeleri için SGD ile birleştiğinde daha verimli ve kullanışlı olmaktadır.

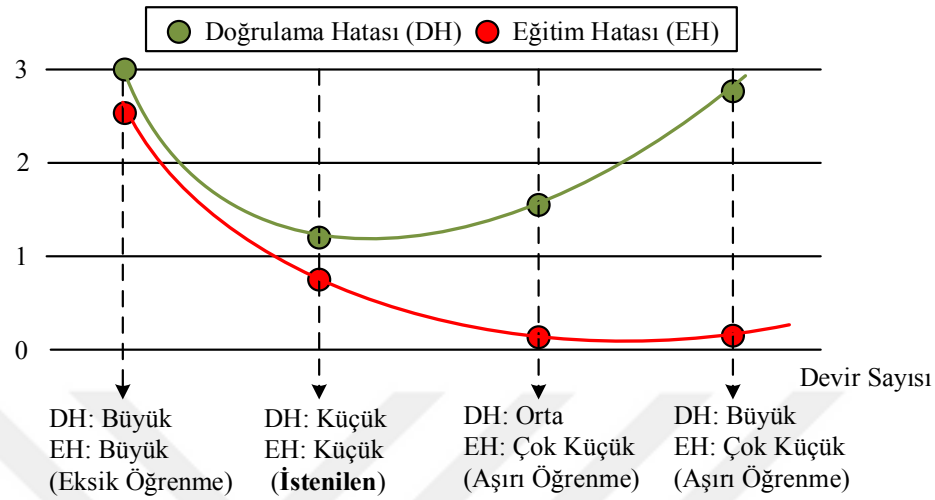
Mini-batch boyutunun belirlenmesi eğitim setine ve kullanılacak ağ modeline göre değişim gösterse de literatürde yapılan deneysel çalışmalar sonucu [205,206] aşağıdaki sonuçlar elde edilmiştir:

- 1, 2, 4, 8, 16 gibi çok küçük mini-batch boyutları ağ hesaplamalarında daha fazla gürültüye sebep olmaktadır.
- 512, 1024, 2048 gibi çok büyük mini-batch boyutları hesaplama maliyetinin yüksek ve modelin doğruluğunun ya da performansının düşük olmasına sebep olmaktadır.
- Veri boyutu, türü ve yapılacak işleme bağlı olarak değişen 32, 64, 128 ve 256 mini-batch boyutları seçilebilecek en iyi değerler olarak kabul edilmektedir.

5.4.3. Devir Sayısı

Kısaca devir (epoch) sayısı olarak isimlendirilen eğitim iterasyonlarının sayısında her bir devir, tüm veri setinin ileri ve geriye doğru (forward and backward) tek bir geçiş adımıdır. Fazla veri gerektirmeden oluşturulan modelin doğruluğunu arttırmak için kullanılmaktadır. Bu nedenle doğru sayıda devir seçimi önemli bir işlemdir. Şekil 5.6'da sunulan model karmaşıklık grafiğinde örnek bir modelin devir sayısına göre değişen eğitim sonuçları gösterilmiştir. Devir sayısı istenilen değerden büyük olduğunda model aşırı öğrenme problemiyle ve istenilen değerden küçük olduğunda ise eksik öğrenme problemiyle karşı karşıya kalmaktadır. İstenilen devir değerini belirlemek için iki çözüm söz konusudur. İlki sezgisel yoldur; model eğitimi süresince doğrulama ve eğitim hataları grafiksel ya da sayısal değerler üzerinden gözlemlenerek doğrulama hatasının yükselmeye başladığı noktada eğitim durdurulmaktadır. İkincisi ise bu işlemin otomatik bir şekilde yapılmasıdır. Bu tez çalışmasında, oluşturulan derin sinir ağı

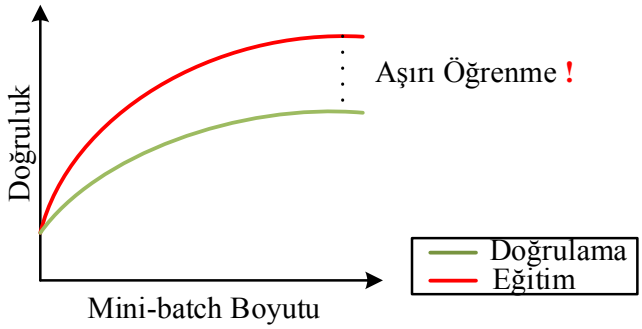
modellerinde kullanılan devir sayısı parametresi, TensorFlow tarafından desteklenen erkenden durdurma (early stopping) tekniği kullanılarak otomatik bir şekilde optimize edilmiştir.



Şekil 5. 6. Model karmaşıklık grafiğine göre devir sayısının belirlenmesi

5.4.4. Gizli Katmanların Sayısı

Bir modelin kapasitesi, farklı problem ve görevleri öğrenme yeteneği olarak ifade edilmektedir. Modelin kapasitesi değiştirilerek aşırı öğrenme ya da eksik öğrenme durumları kontrol edilir. Düşük kapasiteli modeller eğitim setini eksik öğrenirken, yüksek kapasiteli modeller eğitim setini aşırı öğrenerek ezberlemektedir [207]. Gizli katmanların/birimlerin sayısı bir modelin öğrenme kapasitesinin ölçülmesi için önemli bir parametredir. Bu parametre değerinin seçiminde eğitim ve doğrulama değerlerinin doğruluk (accuracy) grafiğine bakılmaktadır. Şekil 5.7'de örnek bir modelin eğitim ve doğrulama veri seti üzerindeki mini-batch boyutuna göre doğruluk oranı sonuçları verilmiştir. Bu sonuçlara göre, eğitim değerleri doğrulama değerlerinden daha iyi çıktığı durumda model aşırı öğrenme problemiyle karşı karşıya kalmaktadır. Gizli katman sayısının belirlenmesi ve aşırı öğrenme probleminin çözümü için aşağıdaki öneriler sunulmaktadır:



Şekil 5. 7. Model doğruluk grafiğine göre gizli katman sayısının belirlenmesi

- Modelin eğitimi sırasında doğruluk oranı sonuçlarına bakıldığında, eğitim değerleri doğrulama değerlerinden daha yüksek çıkıyorsa model veri setini aşırı öğrenmiştir ve gizli katman sayısının azaltılması gerekmektedir.
- Aşırı öğrenme probleminin çözümü için ayrıca dropout ve L2 düzenlileştirme tekniklerini kullanılmaktadır.
- Genellikle gizli katman sayısı ideal değerden küçük alındığında bir probleme yol açmazken, çok büyük gizli katman sayısı değerleri modelin aşırı öğrenmesine yol açmaktadır.
- Model eğitimi istenilen doğruluk sonuçlarını sağlamıyorsa, daha fazla gizli katman eklemek ve model hatalarını takip etmek gerekmektedir.

Derin sinir ağının eğitiminde kullanılacak olan diğer parametreler ve terimler aşağıda listelenmiştir:

Kelime Temsil Boyutu: Kelime temsilleri, metinlerin sayısal vektör gösterimleri için derin öğrenme modellerinde kullanılan popüler bir çözümdür. Kelime temsili boyutu ise eğitim aşamasında test edilmesi gereken bir diğer hiper parametredir. Bu vektör boyutu eğitim için kullanılan kelime listesinin boyutuna bağlı olarak değişmesi ile birlikte genellikle 50, 100, 300 ve bazen 1000 değerini alabilmektedir.

Hata Fonksiyonu: Rastgele değerler ile başlatılan w ve b değerleri arasında matris çarpımı gerçekleştirilerek elde eden tahmini y çıktısı gerçek değere yakın bir sonuç vermeyecektir. Oluşturulan modelin tahmin ettiği çıktı değerlerin, gerçek değerlere ne kadar yakın olduğunu söyleyen bir fonksiyona ihtiyaç vardır. Bu bir regresyon problemidir ve toplam karesel hata (sum of squared error - SSE) kullanılır. Tahmin edilen y ve gerçek y değeri arasındaki farkın karesi alınarak modelin tahmin edilen değer ile gerçek değere ne kadar yakın olduğunu anlaması sağlanır.

Gradyan Azaltma Algoritması: Hata fonksiyonunun (error function) değeri ne kadar küçük olursa model gerçek çıktıya o kadar yakın bir tahmin gerçekleştirir. Bu nedenle, matematiksel olarak yapılmak istenen hata fonksiyonunun değerini minimize etmektir. Bu işlem gradyan azaltma (gradient descent) olarak adlandırılır. Gradyan, ağdaki her ağırlığa göre hata fonksiyonunun kısmi türevidir. Zincir kuralı kullanılarak her katmanda, her ağırlığın gradyanı hesaplanır. Her katmanın gradyanları bilindiğinde, hata fonksiyonunu minimize etmek için gradyan azaltma algoritması kullanılır. Bu işlem ağın hatası en aza inene kadar tekrarlanır.

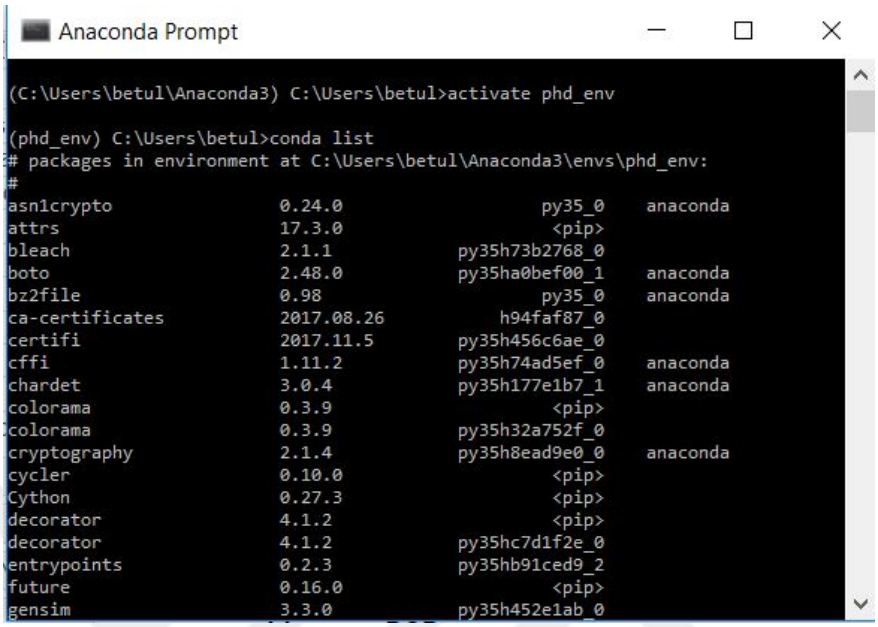
5.5. Geliştirme Ortamları ve Platformları

Bu tez çalışmasında metin analizi mimarisi üzerinde çalıştırılacak derin öğrenme modelleri için seçilen derin öğrenme platformları ve yazılım kütüphaneleri aşağıda verilmiştir. Derin öğrenme mimarisi ile büyük metin verileri üzerinde otomatik duygu analizi ve metin sınıflandırma çalışmaları gerçekleştirilerek, gerçek dünya uygulamalarında performans karşılığı alabilecek pratik ve teorik uygulamalara katkı sağlanmıştır.

5.5.1. Anaconda

Özellikle yapay zeka çalışmalarında yaygın olarak kullanılan Anaconda (https://anaconda.org/), Python programlama dili, yüzün üzerinde veri bilimi paketleri ve

bağımlılıkları ile gelen bir dağıtımdır. Çeşitli kütüphaneler ile ilgili sorunları en aza indirgeyen Conda uygulaması bir paket ve ortam yöneticisidir.



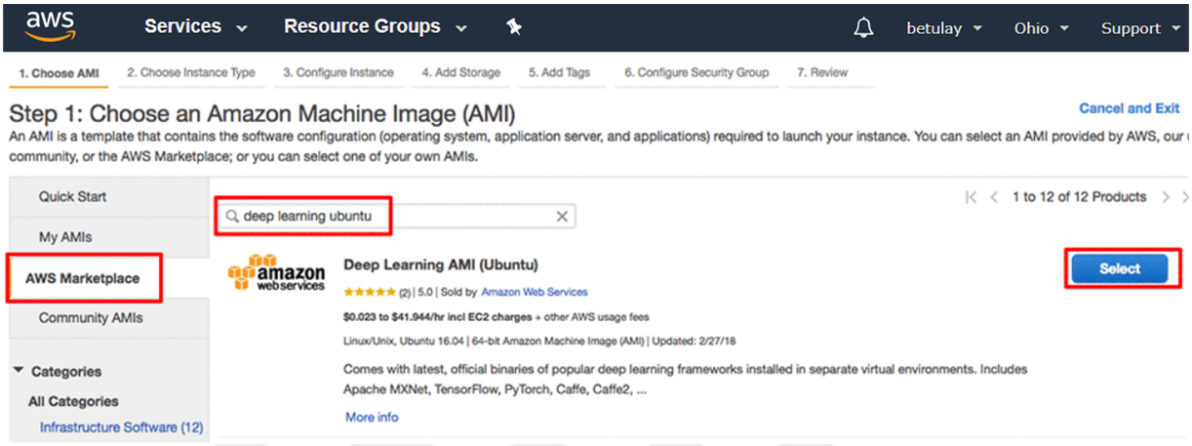
Şekil 5. 8. Anaconda ortamı

Tez çalışması için kurulan Şekil 5.8'deki bu paket yöneticisi kütüphaneleri ve diğer yazılımları bilgisayara yüklemek için kullanılmaktadır. Python kütüphanesinin paket yöneticisi olan pip ile benzerdir. Ancak pip genel amaçlı python kullanıma yönelikken, conda veri bilimine odaklı erişilebilir python ya da python olmayan paketlere odaklanmıştır. Anaconda dağıtımı önceden derlenmiş paketleri kurar, örneğin çeşitli matematik işlemlerdeki performansı hızlandıran MKL kütüphanesi ile derlenmiş Numpy, Scipy ve Scikit-learn ile birlikte gelir.

5.5.2. Amazon Web Servisleri

Zaman maliyeti açısından probleme ve veri setinin boyutuna göre kişisel bilgisayarlardaki CPU üzerinde derin sinir ağlarının eğitilmesi zor olmaktadır. Eğitim, devir sayısına ve modelin büyüklüğüne göre saatleri bazen günleri alabilmektedir. Bunun için hızlı bir alternatif GPU

kullanımıdır. Bu tez çalışmasında geliştirilen modellerin eğitimi için NVIDIA GTX 1080 grafik kartı kullanılmıştır. Fakat bunun yanında bazı modellerin eğitimi için, Esnek Hesaplama Bulutu (EC2) olarak adlandırılan bir servis sunan ve GPU eklentili sanal sunucu üzerinde çalışılmasına olanak sağlayan Amazon web servisleri (Şekil 5.9) kullanılmıştır.



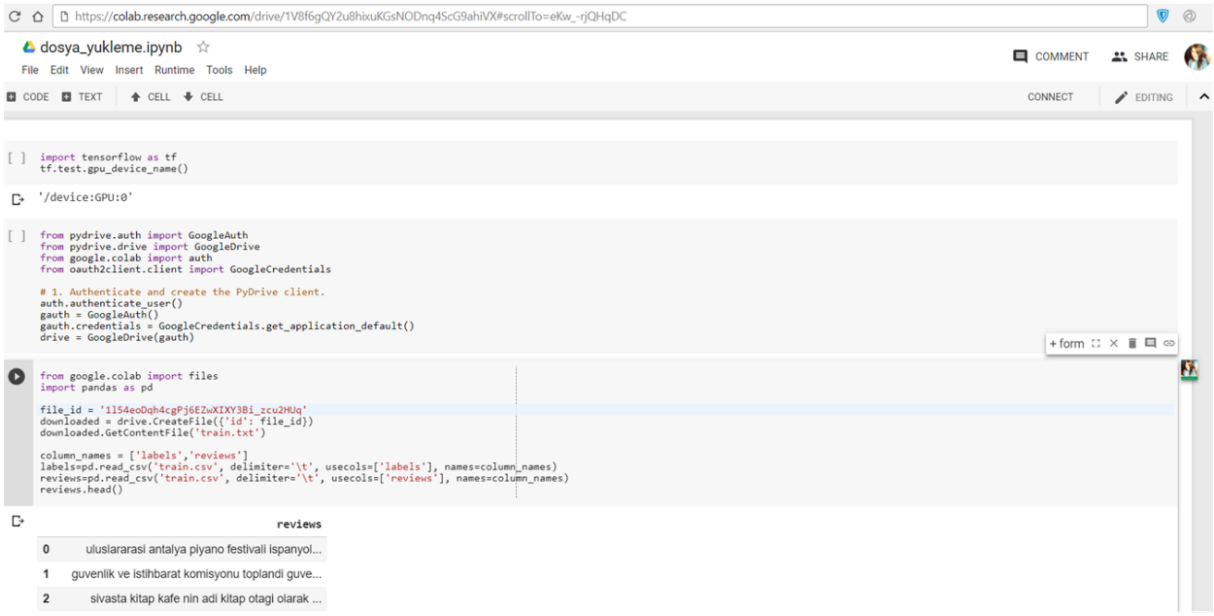
Şekil 5. 9. Amazon web servisinden seçilen Amazon Makine Görüntüsü (AMI)

```
=============================================================================
      __|  __|_  )
      _|  (     /   Deep Learning AMI  (Ubuntu)
     ___|\___|___|
=============================================================================

Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1039-aws x86_64v)

Please use one of the following commands to start the required environment with the framework of your choice:
for MXNet(+Keras1) with Python3 (CUDA 9) _____________________ source activate mxnet_p36
for MXNet(+Keras1) with Python2 (CUDA 9) _____________________ source activate mxnet_p27
for TensorFlow(+Keras2) with Python3 (CUDA 8) ________________ source activate tensorflow_p36
for TensorFlow(+Keras2) with Python2 (CUDA 8) ________________ source activate tensorflow_p27
for Theano(+Keras2) with Python3 (CUDA 9) ____________________ source activate theano_p36
for Theano(+Keras2) with Python2 (CUDA 9) ____________________ source activate theano_p27
for PyTorch with Python3 (CUDA 8) ____________________________ source activate pytorch_p36
for PyTorch with Python2 (CUDA 8) ____________________________ source activate pytorch_p27
for CNTK(+Keras2) with Python3 (CUDA 8) ______________________ source activate cntk_p36
for CNTK(+Keras2) with Python2 (CUDA 8) ______________________ source activate cntk_p27
for Caffe2 with Python2 (CUDA 9) _____________________________ source activate caffe2_p27
for base Python2 (CUDA 9) ____________________________________ source activate python2
for base Python3 (CUDA 9) ____________________________________ source activate python3
```

Şekil 5. 10. Derin öğrenme modellerinin çalıştırılması için önceden yüklenmiş yazılımları ve işletim sistemini sağlayan derin öğrenme bulut ortamı-Deep Learning AMI (https://aws.amazon.com/)

Amazon EC2 üzerinde çalışan Deep Learning AMI, Şekil 5.10'da gösterildiği gibi derin öğrenme modellerinin geliştirilmesi ve eğitilmesi için Apache MXNet, TensorFlow, PyTorch, Caffe, Caffe2, Keras, Chainer, CNTK ve Theano gibi derin öğrenme platformlarının kullanılmasına olanak sağlar.

5.5.3. Google Colaboratory



Şekil 5. 11. Google Colaboratory geliştirme ortamı

Tez çalışması kapsamında derin sinir ağı modellerinin eğitiminde kullanılan bir diğer geliştirme ortamı olan Google Colaboratory kısaca Colab, makine öğrenimi araştırmalarına ve çalışmalarına yardımcı olmak için oluşturulan bir projedir. Ayrıca kurulum gerektirmeyen ve tamamen bulutta çalışan bir Jupyter notebook ortamıdır. Şekil 5.11'de ekran görüntüsü gösterilen Colab, derin öğrenme çalışmalarında yardımcı olacak ücretsiz GPU kullanımı gibi pek çok seçenek ve alternatif sunmaktadır [208].

5.5.4. TensorFlow

2015 yılında Google tarafından açık kaynak bir proje olarak piyasaya sürülen TensorFlow (https://www.tensorflow.org/), derin sinir ağlarını oluşturmak için kullanılan bir platformdur. Java, C++, Go ve Python dilleri için destek sunmaktadır. Bu tez çalışmasında, yeni modellerin geliştirilmesi için en çok tercih edilen ve kullanılan Python dili tercih edilmiştir. Tüm matematiksel işlemler düğüm (node) olarak ifade edilir ve bu düğümler arasındaki kenarlar (edge) tensorlerdir. Tensorflow' da veri modelleri integer, float ya da string tipi belirtilerek depolanmaz. Bu değerler tensor adı verilen bir nesnede tutulmaktadır. Tensorler, çok boyutlu veri dizileridir. Derin öğrenmede yaygın olarak kullanılan tensorler şunlardır: Skalar (0B tensor), Vektör (1B tensor), Matris (2B tensor), 3B tensor, 4B tensor ve 5B tensor. Farklı boyut ve çeşitlerdeki bu tensorler aşağıdaki kod bloğunda verilmiştir:

```
#Tensorflow kütphanesinin yüklenmesi
>>import tensorflow as tf

#a, 0-boyutlu bir int32 tensor
>> a = tf.constant(123)

#b, 0-boyutlu bir string tensor
>> b = tf.constant("derin ogrenme")

#c, 1-boyutlu bir int32 tensor
>> c = tf.constant([123, 456, 789])

#d, 2-boyutlu bir int32 tensor
>> d = tf.constant([[123, 456, 789],
                   [111, 222, 333]])

#e, 3-boyutlu bir int32 tensor
>> e = tf.constant([[[123, 456, 789],
                    [111, 222, 333]],
                    [[123, 456, 789],
                    [111, 222, 333]]])
```

Tensorflow ağdaki gradyanları otomatik hesaplar ve hata fonksiyonunu en aza indirgemek için optimize eder. Bu hesaplamalar bir ya da daha fazla Grafik İşlemci Birimleri (GPU) üzerinde gerçekleştirilerek büyük ölçüde hızlandırılabilmektedir. Bu nedenle, GPU hesaplaması derin öğrenme uygulamalarında bir zorunluluk haline gelmiştir.

TensorFlow, Windows, Linux, Mac işletim sistemli makinelerde ve Android gibi mobil bilgisayar platformlarında çalışabilme özelliğe ile taşınabilirdir. TensorFlow kaynak koduna etkin bir şekilde katkıda bulunan ve hata geri bildirimleri ile daha iyi bir noktaya çıkmasına sağlayan bir geliştirici topluluğu ve kullanıcısı vardır. Bu özelliği ile araştırmacılara kullanım kolaylığı ve destek sağlamaktadır. Derin sinir ağı modellerini geliştirmek, analiz etmek ve iyileştirmek için sunduğu TensorBoard aracı ile kullanışlı bir platformdur.

5.5.5. Keras

Keras (https://keras.io), derin öğrenme ağlarını hızlı bir şekilde oluşturmak için TensorFlow kullanımına izin veren yüksek seviyeli bir Uygulama Programı Arayüzüdür (UPA). Keras ile sıralı katmanlardan oluşacak sinir ağı modeli oluşturulurken Sıralı katmanlardan oluşacak sinir ağ modeli için *keras.models.Sequential* sarmalayıcı sınıfı (wrapper class) kullanılmaktadır. Bu sınıf modeli eğitmek ve çalıştırmak için kullanılan compile(), fit() ve evaluate() gibi yaygın methodlar ile Keras model arayüzünü implemente etmektedir.

```
from keras.models import Sequential
 model.Sequential()
```

Keras Katman sınıfı, tam bağlı katmanlar, maksimum havuz katmanları ve aktivasyon katmanları gibi çeşitli sinir ağı katmanları için ortak bir arayüz sağlamaktadır. Bir modele yeni bir katman eklemek için modelin add() methodu kullanılmaktadır. Tek bir gizli katmanı olan basit bir model şu şekilde oluşturulmaktadır:

```
Import numpy as np
from keras.models import Sequential
```

```
from keras.layers.core import Dense, Activation
 X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]],
              dtype=np.float32)
 y = np.array([[0], [1], [1], [0]], dtype=np.float32)
 model=Sequential()
 model.add(Dense(32,input_dim=X.shape[1]))
 model.add(Activation('softmax'))
 model.add(Dense(1))
 model.add(Activation('sigmoid'))
```

İlk gizli katman, 32 düğüm oluşturur ve her bir düğüm girdi olarak iki elemanlı vektör alır. Her katman, bir önceki katmanın çıktısını girdi olarak almaktadır. Bu zincir modelin çıktısı olan son katmana kadar devam etmektedir. Modeli oluşturduktan sonra çalıştırmadan önce derlenmesi (compile) gerekmektedir. Keras model, derleme aşamasında TensorFlow kütüphanesini çağırmaktadır. İstenilen girdi verisi üzerinde modeli çalıştırmadan önce, optimizer, hata fonksiyonu (loss function) ve diğer parametrelerin tanımlanmalıdır. Tasarlanan modelin derlenmesi için seçilen örnek parametreler şu şekildedir:

```
model.compile(loss="categorical_crossentropy",
                optimizer="adam", metrics = ["accuracy"])
```

Yalnızca iki sınıf kullanıldığı için hata fonksiyonu için çapraz entropi (crossentropy) seçilmiştir. Optimizasyon için SGD yerine Adam algoritması [209] seçilmiştir. Modeli değerlendirmek için doğruluk (accuracy) metriği belirlenmiştir. Model mimarisinin sonuçları `model.summary()` komutu ile görülebilmektedir. Son olarak model, epoch sayısını içeren fit() metodu ile eğitilir ve modeli değerlendirmek için evaluate() metodu kullanılır.

TensorFlow ve Keras kütüphanelerine alternatif olarak pek çok derin öğrenme platformu vardır. Bu popüler platformların karşılaştırması Tablo 5.2'de sunulmuştur. Bu tez çalışmasında bu iki platformun kullanılma sebebi popüler olması, kullanım kolaylığı, geliştiricilerin destek sağlaması ve dokümantasyonlara erişim kolaylığıdır. Shi ve arkadaşları [210] , yayınladıkları makalede dört popüler platformun CNN, RNN ve geleneksel ileri beslemeli sinir ağları üzerinde

birden fazla GPU ve CPU kullanarak karşılaştırmasını yapmıştır. Karşılaştırılması yapılan Caffe (http://caffe.berkeleyvision.org/), CNTK (https://cntk.ai) , TensorFlow ve Torch (http://torch.ch/) kütüphanelerinin çok verimli bir şekilde GPU kullanabildiklerini göstermiştir. Fakat GPU üzerindeki performans sonuçlarına bakıldığında, öne çıkan bir platform olmadığını ve bu platformların hala iyileştirilmeye ihtiyaç duyduğunu belirtmiştir.

Tablo 5. 2. Derin öğrenme platformlarının karşılaştırılması

Platform	**Geliştirilen Dil**	**Topluluk Desteği**	**Model Esnekliği**	**Kolay Konfigürasyon**	**Hız**	**GPU Paralelleştirme**
TensorFlow	Python	Çok Güçlü	Çok iyi	Çok iyi	İyi	Güçlü
Keras	Python	Çok Güçlü	Çok iyi	Çok iyi	İyi	Güçlü
Caffe	C++	Çok Güçlü	İyi	İyi	İyi	Orta
MXNet	R, Python, Julia, Scala	Çok Güçlü	İyi	İyi	İyi	Çok Güçlü
Torch	Lua, Python	Çok Güçlü	Çok iyi	Çok iyi	Çok iyi	Güçlü
Theano	Python, C++	Güçlü	İyi	Orta	İyi	Orta
CNTK	C++	Güçlü	İyi	Orta	İyi	Orta

5.5.6. Tensorboard

TensorBoard, TensorFlow'da oluşturulan derin sinir ağının yapısını, parametrelerini ve metriklerini görselleştirmemizi sağlayan web tabanlı bir uygulamadır. Derin sinir ağının daha kolay ve hızlı bir şekilde optimize edilmesine, hataların gözlemlenerek bulunmasına ve düzeltilmesine yardımcı olmaktadır.

Bu tez çalışmasında kullanılan modellerin karşılaştırılmasını yapmak, eğitim sonuçlarını gözlemleyerek farklı derin sinir ağı mimarilerini ve hiper parametrelerini değiştirmek ve optimize etmek için TensorBoard kullanılmıştır. Bu amaç doğrultusunda, TensorFlow ve Keras platformları üzerinde oluşturulan ağları görselleştirmek için TensorBoard kurulumu gerçekleştirilmiştir. TensorFlow ve Keras'ta ağ eğitimi gerçekleştirirken özel bir log yolu tanımlayarak aktivasyon metrikleri içine TensorBoard eklenmiştir. Bu işlemin

gerçekleştirilmesi için “Keras callbacks” olarak adlandırılan eğitim süresince çalışacak olan özel bir fonksiyon sınıfı kullanılmıştır. Her bir devir sayısından (epoch) sonra kaydedilen ağ ağırlıkları ile birlikte log dosyaları tutularak ağın eğitimi sonucu doğruluk (accuracy) ve hata (loss) değerlerinin görselleştirilmesi sağlanmıştır. Bu işlemler için öncellikle TensorBoard callback sınıfı eklenir. Daha sonra derin sinir ağının eğitimi için kullanılacak olan callback’ler için bir liste oluşturulur ve bu liste modelin “.fit()” metoduna bir argüman olarak iletilir.

5.5.7. FastText Kütüphanesi

Bu tez çalışmasında haberler ve beyazperde film yorumlarından önceden eğitilmiş kelime temsillerinin oluşturulmasında word2vec yanı sıra FastText Kütüphanesi (https://fasttext.cc/) de kullanılmıştır. FastText temsillerinin word2vec temsillerine göre ön plana çıkan avantajı kelime vektörlerini öğrenirken kelimelerin içyapısını dikkate almasıdır [211]. Türkçe gibi morfolojik olarak zengin diller için bu özellik yararlı olmaktadır. Word2vec algoritmasında eğitim yapılırken her kelime atomik yani benzersiz düşünülür fakat FastText ile her kelime n-gram karakterlerine göre guruplanır. Örneğin “film” kelimesi, “filmi”, “filmin”, “filmde”, “filmler” gibi n-gram özelliklere ayrıştırılarak oluşturulmaktadır.

6. DERİN ÖĞRENME MODELLERİ İLE METİN SINIFLANDIRMASI

6.1. Problem Tanımı

Rosenblatt, 1957 yılında ilk algılayıcıyı keşfettikten sonra, sinir ağları gittikçe dikkat çekmeye başladı. Fakat bellek ve işlemci hızı kısıtlamaları sebebiyle potansiyel uygulamalar ve araştırmalar büyük bir sorunla karşı karşıya geldi. Son on yılda, daha güçlü ve daha ucuz bilgisayarlar sayesinde, makine öğrenimi araştırmacıları, metinler, resimler ve sesler içeren çok geniş veri kümeleri kullanan birkaç derin katmana sahip karmaşık ve büyük modeller inşa ettiler. Derin öğrenme yöntemleri, NLP, görüntü ve konuşma tanıma gibi pek çok uygulamada son yıllarda başarılı sonuçlar alınan makine öğrenimi araştırmacılarının attığı önemli bir adımdır. Derin öğrenmenin makine öğrenimi alanında dikkat çekmesinin pek çok sebebi vardır. Bu sebeplerden birisi, GPU (Graphics Processing Unit) olarak adlandırılan yeni ekran kartı işlemcilerinin geliştirilmesi ile birlikte ağların eğitimi için gerekli süre önemli ölçüde azaltılmıştır. Bir diğer sebep, derin mimarları eğitmek için kullanılacak çeşitli veri setlerinin artması ve bu veri setlerine erişimin kolaylaşmasıdır.

Diğer makine öğrenmesi algoritmaları ile kıyaslandığında derin öğrenme, özellikleri otomatik öğrenme gücüyle ön plana çıkmaktadır. Sinir ağlarının her katmanı bir önceki katmanlardan farklı bir soyutlama öğrenmektedir. Örneğin, bir nesne tanımada birinci katman farklı kenarları öğrenir, bir sonraki katman kenarları bir araya getirerek motifleri öğrenir, sonraki katman motifleri birleştirerek parçaları oluşturur ve sonraki katmanda nesnenin yüksek seviyeli gösterimi vardır. Bu hiyerarşik özelliklerin gösterim şu şekildedir:

[piksel -> kenar -> doku -> motif -> parça -> nesne].

Benzer şekilde bir metin tanımada işleminde birinci katman karakterleri öğrenir, bir sonraki katman karakterleri bir araya getirerek kelimeleri öğrenir, sonraki katman kelime gruplarını oluşturur ve en son katmanda tüm metnin tanıma işlemi gerçekleştirilir. Bu hiyerarşik özelliklerin gösterim şu şekilde yapılandırılabilir*:*

[karakter -> kelime -> kelime grubu -> cümle -> öykü].

Derin öğrenme, metinlerin sınıflandırılması ve kümelenmesi, doküman özetleme, müşteri ilişkileri yönetimi, web madenciliği, duygu analizi ve diğer bilgiye erişimi içeren metin analitiği problemlerine etkili bir çözüm olarak ortaya çıkmıştır. Bu tez bölümünün amacı, Türkçe metinler üzerinde ikili duygu sınıflandırma ve çoklu metin sınıflandırma problemlerinin çözümü için çeşitli derin öğrenme modellerinin araştırılması ve uygulanmasıdır. Ayrıca kullanılan derin öğrenme modellerinin önceden eğitilmiş Türkçe kelime temsilleri ile geliştirilmesi ve performans sonuçlarının karşılaştırılmasıdır. Bu temel kapsamlar ile birlikte bu bölümde şu soruların cevaplarını bularak derin öğrenme yaklaşımları ile metin sınıflandırma problemlerine katkı sunmak amaçlanmıştır:

- Derin öğrenme yöntemleri genellikle İngilizce üzerinde geliştirilmiş ve metin sınıflandırma çalışmalarında yüksek başarı elde etmiştir. Web üzerinde ve sosyal medyada erişilebilir, çok büyük Türkçe içeriğin oluşmasıyla birlikte Türk dilinde yazılmış metinlerin sınıflandırma ihtiyacı doğmuştur. Bu ihtiyaçtan yola çıkılarak sunulan bu çalışmada, derin öğrenme yaklaşımları ne kadar başarı elde etmiştir?
- Türkçe için çok büyük bir veri seti üzerinde önceden eğitilmiş kelime temsilleri oluşturularak kullanılan bu derin öğrenme yaklaşımlarının performansını arttırmak mümkün mü?
- Farklı mimariler, yöntemler, katmanlar ve hiper parametre optimizasyonları ile hangi derin öğrenme modellerinin metin sınıflandırma problemlerindeki başarısı yüksektir?
- İkili duygu sınıflandırması ve haber metinleri gibi çoklu metin sınıflandırması problemlerinde kullanılan derin öğrenme yaklaşımlarının performansı nedir?
- Aşırı ya da eksik öğrenme problemlerinin çözümü için hangi hiper parametreler ve modeller önerilmektedir?

Sonuç olarak bu tez çalışmasında kullanılan farklı derin öğrenme mimarileri, Türkçe metin sınıflandırma problemlerinde başarılı sonuçlar ile yüksek performans sergilemiştir. Bu bölümde duygu sınıflandırması ve haber metinlerin sınıflandırması olmak üzere iki uygulama

gerçekleştirilmiştir. Ayrıntılı olarak anlatılacak olan duygu sınıflandırma bölümünde kullanılan mimariler, haber sınıflandırmada da doğruluk ve güvenilirlik açısından test edilerek performans sonuçları görsel ve istatiksel olarak ortaya koyulmuştur.

6.2. Motivasyon

Doküman sınıflandırma akademik alanda önemli olmakla beraber olağan üstü ekonomik değeri olan bir alandır. Kullanıcı davranışları web tarama geçmişindeki metinlerin analizi ile çıkarılabilir ve bu davranışa göre reklam tavsiyeleri gerçekleştirilebilir. Bir ürün hakkında yapılan yorumlar otomatik analiz edilebilir ve bu analiz sonucu ürün sağlayıcılar doğru iş planları ve kararları alabilir. Akademik dünyadaki makaleler konu içeriğine göre sınıflandırılabilir ve en doğru kategoride olması sağlanabilir. Özet olarak bir doküman içeriği anlaşılarak sınıflandırılabilmesi önem arz etmektedir. Bu problemin çözümü için en temel yaklaşım dokümanların ya da kelimelerin sayısal vektörlere çevrilmesi ve vektör uzayında her bir metnin karşılaştırılması ve sınıflandırılmasıdır.

Görüş madenciliği olarak da adlandırılan duygu analizi, NLP ve metin madenciliği problemlerinden birisidir [212]. Hem akademik araştırmalarda hem de endüstrideki uygulamalarda hızla gelişen önemli çalışma alanlarındandır. Sınıflandırma problemlerinden olan duygu analizi, bir kullanıcının yorumlarından oluşan metnin genel duygusunu tahmin etmeyi hedeflemektedir. Örneğin, gösterimde olan bir film hakkında izleyicinin görüşünün pozitif veya negatif olduğunun tespit edilmesi vb. Duygu sınıflandırması, metinde ifade edilen görüşün pozitif ya da negatif olduğunu etiket veya puan ile doğru bir şekilde bulmayı amaçlamaktadır.

Çevrimiçi erişilebilir çok sayıda etiketli veri ve derin öğrenme yöntemlerinin artması ile birlikte metin verilerinden duygu çıkarımı, NLP çalışmalarında yeniden popülerlik kazanmıştır. Kişilerin belirli bir olaya ya da ürüne yönelik duygularının tespit edilmesi, daha doğru planlamalar ve iş kararları alınmasında etkili olmaktadır. Bilgi ya da veri toplamak için e-postalar, çevrimiçi anketler, sosyal medya gönderileri, blog yayınları ve benzeri kişilerin görüşlerini içeren çok farklı kaynaklar vardır. Bu kaynaklardan toplanan veri miktarı, insanlar

tarafından yapılan manuel analizler için genellikle çok büyüktür. Öte yandan, göz ardı edilemeyecek kadar çok değerlidir. Duygu analizi bu problemin çözümü için, önceden tanımlanmış ya da etiketlenmiş veriler üzerinde model eğitir ve etiketsiz ya da tanımsız duyguları içeren veriler üzerinde bu modeli kullanır.

Cui ve arkadaşları [213] çevrimiçi (online) marka yorumları üzerinde duygu analizi gerçekleştirmiştir. Pasif-Agresif Algoritma Tabanlı Sınıflandırıcı, Dil Modelleme (DM) Tabanlı Sınıflandırıcı ve Winnow Sınıflandırıcısı [214] olmak üzere üç teknik üzerinde duygu analizini tartışmışlardır. Socher ve arkaşları [215], cümleleri tam etiketli ayrıştırma ağaçlarını kullanarak pozitif veya negatif olarak tanımlamak için Özyinelemeli Sinir Tensor Ağı (Recursive Neural Tensor Network) adlı yeni bir model önermektedir.

Destek Vektör Makineleri, Naive Bayes, K En Yakın Komşu ve Maksimum Entropi gibi geleneksel gözetimli ve gözetimsiz ML algoritmaları, duygu analizinde yaygın olarak kullanılmaktadır. Ancak, yakın zamanda CNN, LSTM ve RNN ağları gibi derin öğrenme mimarileri yüksek başarı sonuçlarından dolayı daha çok tercih edilmektedir. Fakat her türlü problem için evrensel derin öğrenme modeli bulunmadığından, problem alanlarına göre uygun model seçilmelidir. Farklı ağ modelleri, bilgisayar görmesi ve NLP gibi farklı uygulama alanlarında farklı performans sonuçları vermektedir [17].

Sarkar ve arkadaşları [216], duygu analizi için son zamanlardaki yaklaşımların GloVe [217] ve word2vec [16] modeller gibi kelime temsillerine dayalı metin özelliklerini kullandıklarını belirtmektedir. Önceki çalışmalar, kelime torbaları veya TF-IDF gibi diğer önemli metin gösterimi modellerini yaygın olarak kullanmıştır. Bu tez çalışmasında da word2vec ile yaklaşık 80 bin benzersiz kelime içeren BEYAZPERDE Türkçe film yorumları veri seti kullanılarak kelime temsilleri oluşturulmuştur. Benzer şekilde aynı algoritma ile yaklaşık 1,3 milyon benzersiz kelime içeren HABERLER Türkçe haber veri seti kullanılarak kelime temsilleri oluşturulmuştur.

Metin sınıflandırması alanındaki çalışmalar genellikle İngilizce için yoğunlaşmaktadır. Son yıllarda, web üzerinde erişilebilir, önemli büyüklüklerde Türkçe içeriğin oluşmasıyla birlikte Türk dilinde metin madenciliği geliştirme ihtiyacı doğmuştur. Metin madenciliği yardımıyla,

kullanıcı yorumları ve farklı ana başlıklardaki dokümanlar kolay bir şekilde sınıflandırılarak ve daha etkin iş kararları alınabilmektedir.

Bu çalışmanın temel bölümünde, popüler derin öğrenme ağları ile Türkçe kelime temsilleri kullanılarak film yorumları üzerinde duygu sınıflandırması gerçekleştirilmiştir. Türkçe dilinde yapılacak duygu sınıflandırması çalışmalarında standart bir veri seti olmadığı için www.beyazperde.com sitesinden Türkçe film yorumları toplanmıştır. Oluşturulan veri seti ve kelime temsilleri kullanılarak popüler derin öğrenme modelleri test edilip duygu sınıflandırması doğruluğuna etkisi detaylı olarak sunulmuştur.

Çalışmanın ikinci bölümünde, duygu sınıflandırması performans sonuçlarına göre seçilen en iyi model ve parametre seçimleri ile 5 kategorili haber sınıflandırması gerçekleştirilmiştir. Türkçe haber sınıflandırması çalışmalarında standart bir veri seti olmadığı için çeşitli haber sitelerinden yaklaşık 10 milyon haber toplanarak kelime temsilleri eğitilmiştir. Bu veri setinden ekonomi, politika, kültür, dünya ve spor konusu içeren 500 bin haber seçilerek, oluşturulan eğitim verisi üzerinde derin öğrenme modelleri eğitilip test edilmiştir. Performans sonuçları değerlendirilip tartışılmıştır.

6.3. İlgili Çalışmalar

Kullanıcı yorumları gibi çevrim içi ortamda artan sosyal içeriklerle birlikte, otomatik duygu sınıflandırması talebi büyük bir artış göstermiştir. Bu amaç doğrultusunda, yakın zamana kadar yapılan çalışmalar [218][219][220]'de özetlenmiştir. Film eleştirileri üzerine yapılan duygu analizi çalışmalarında geleneksel makine öğrenmesi yöntemlerinden Destek Vektör Makineleri, Naive Bayes, K En Yakın Komşu ve Maksimum Entropi algoritmaları yaygın olarak kullanılmıştır. Kalaivani ve Lillian [218], 1000 pozitif ve 1000 negatif yorum içeren veri seti üzerinde gözetimsiz makine öğrenmesi algoritmalarını uygulayarak duygu sınıflandırması gerçekleştirmiştir. Bu çalışmada Destek Vektör Makinelerinin, Naive Bayes ve K En Yakın Komşu algoritmalarına göre %80 doğruluk oranı ile daha iyi sınıflandırma işlemi yaptığı gözlemlenmiştir. Bir diğer çalışma [219], IMDB (imdb.com) film eleştirileri veri tabanından alınan 1400 adet pozitif ve negatif yorum üzerinde, Naive Bayes Algoritmasının %65 doğruluk

oranı (accuracy) ile Destek Vektör Makineleri' ne göre daha iyi bir sınıflandırma sonucu elde ettiği belirtilmiştir. Bhadane [220], verilen bir metni pozitif ya da negatif olarak sınıflandırmak için belirli statik özellikler ekleyerek Destek Vektör Makineleri'ni kullanmıştır. Kelime köklerinin bulunması, metin içerisinde 30 defadan fazla ve 4 defadan az geçen kelimelerin çıkarılması, kelimeler arasındaki eş anlamlılık ve zıt anlamlılık gibi ilişkileri içeren WordNet kullanımı, sık kullanılan kelimelerin (stopwords) kaldırılması gibi pek çok ön işlemeden geçirilen yorumlar üzerinde %78 doğruluk oranına ulaşılmıştır. Geleneksel makine öğrenmesi yöntemleri ile yapılan duygu analizi çalışmalarında boyutsallık sorunu [221] ve özellik seçimlerine bağlı olarak doğruluk oranlarının değişmesi önemli bir sınıflandırma problemidir. Yüksek boyutlu veri setleri kullanıldığında, belirli öğrenme algoritmaları kötü davranış sergilemekte ve doğru olmayan sonuçlar elde etmektedir.

Makine öğrenmesi tekniklerinin gelişmesi ile büyük verilerin işlenmesi ve analizinde daha karmaşık modellerin oluşturulması mümkün hale gelmiştir. Metin analizinde geliştirilen bu modeller n-gram, kelime torbası (bag of words) ve kelime frekansları (TF-IDF) gibi yaygın yöntemlerde kullanılan basit modellere göre üstünlük sağlamıştır. Son günlerde geliştirilen en başarılı model ise kelimelerin dağıtık gösterimleridir. Kelimelerin dağıtık gösterimleri [13], çeşitli yapay sinir ağları tabanlı dil modellerinden elde edilmektedir ve n-gram modellere göre üstünlük sağlamaktadır [14]. Mikolov ve arkadaşları farklı sinir ağ tabanlı dil modelleri sunmuştur [15]. En son geliştirdikleri skip-gram modellerini içeren word2vec yazılımı ile de basit bir logaritmik doğrusal sınıflandırma ağı oluşturmuştur [16]. Skip-gram modeli, çok büyük miktarda yapılandırılmamış metin verisinden yüksek doğrulukta kelime vektörünün gösterimlerini öğrenmek için etkili bir yöntemdir. NLP çalışmalarında kullanılan derin öğrenme yöntemlerinin çoğu, sinir dil modellerinden elde edilen kelime vektörlerinin gösterimlerini öğrenmektedir.

Derin öğrenme yeni bir kavram olmasa da, geçtiğimiz son on yıl içerisinde, gittikçe artan veri miktarı ile birlikte daha etkin eğitim modellerinin geliştirilmesi ve hesaplama kapasitesindeki ilerlemeler nedeniyle popülerlik kazanmıştır. Görüntü ve ses tanıma çalışmalarında başarılı sonuçlar elde eden derin öğrenmenin son durağı NLP'dir. Ouyang [222], 11855 yorum ve beş farklı sınıf içeren (pozitif, negatif, biraz pozitif, biraz negatif, nötr) bir film

eleştirileri veri seti üzerinde word2vec ve CNN ile oluşturduğu derin sinir ağı mimarisini uygulamıştır. Oluşturduğu modelin, Öz yinelemeli Sinir Ağları (RNN) ve Matris-Vektör Öz yinelemeli Sinir Ağları (MV-RNN) ile karşılaştırdığında %45.4 doğruluk oranı ile daha iyi sınıflandırdığını gözlemlemiştir. Bir diğer çalışmada [223], LSTM ve CNN sinir ağı modellerini kullanarak IMDB ve Yelp (yelp.com/dataset_challenge) veri setlerinden alınan kullanıcı yorumları üzerinde duygu analizi çalışmaları gerçekleştirmiştir. Deneysel sonuçlara göre LSTM, çoklu filtre kullanan CNN modele göre daha iyi performans göstermiştir.

Zhou [224], Stanford Sentiment Treebank (SST) [215] veri setini kullanarak film eleştirileri üzerinde duygu sınıflandırması gerçekleştirmiştir. CNN ve LSTM modellerinin birleşiminden oluşan mimarileri, çok katmanlı CNN ve RNN modellere göre daha iyi sonuçlar elde etmiştir. 2-sınıf için (pozitif ve negatif) %87.7 doğruluk oranına ulaşırken, 5-sınıf için (çok pozitif, pozitif, nötr, çok negatif, negatif) %49.2 doğruluk oranı elde etmiştir. Yoon Kim [225], word2vec üzerinde eğitilen basit bir CNN modelini kullanarak çeşitli veri setlerinde duygu sınıflandırması gerçekleştirmiştir. Bir cümlenin nesnel mi öznel mi olduğunu sınıflandırmak için kullanılan Subjectivity veri seti [226] üzerinde %93.2, pozitif ve negatif müşteri yorumlarının bulunduğu CR veri setinde [227] %85, pozitif ve negatif film yorumlarını içeren SST veri setinde ise %88.1 doğruluk oranı elde etmiştir. Collobert ve arkadaşları [162] tarafından yapılan çalışmada, İngilizce Wikipedia üzerinde eğitilen word2vec kelime temsillerinin kullanılmasının, performansı önemli ölçüde arttırdığı görülmüştür. Bu durumun Mikolov ve arkadaşlarının [228] önerdiği mimariden mi yoksa Google haberlerden alınan 100 milyon kelime üzerinde eğitilmiş vektörlerden mi kaynaklandığının açık olmadığı belirtilmiştir.

Kato ve Goto [229], word2vec algoritmasını kullanarak Japon dilinde yayınlanmış web haberleri üzerinde kelime temsilleri eğitmiştir. Bu kelime temsillerini kullanarak ekonomi, eğlence, bilim içeren 6 kategoride haber verisini üzerinde sınıflandırma gerçekleştirmiştir. 800 etiketli veri setinin, 600 örneğini eğitim için ayırmıştır. Word2vec kullanımının metin sınıflandırmada pratik olduğunu söyleyen çalışmanın sınıflandırma başarısı %78 olarak kaydedilmiştir. Zhang ve arkadaşları [10], spor, finans, eğlence, teknoloji ve otomobil içeren 5 kategori ve her bir kategoride seçilen 90,000 eğitim örneği ile Sogou haber veri seti üzerinde metin sınıflandırma gerçekleştirmiştir. Word2vec kelime temsilleri ile CNN model kullanarak

gerçekleştirilen sınıflandırmanın test seti üzerindeki hata %4,54 olarak verilmiştir. Word2vec kelime temsilleri kullanmadan ise daha yüksek hata alınmıştır. Aynı çalışmada 496,835 etiketli haber makalesi içeren AG's haber veri seti üzerinde %9,92 sınıflandırma hatası alınmıştır. Veri setinin boyutuna ve içeriğine göre aynı modellerin farklı sonuçlar verdiğini gösteren çalışmada, tüm veri setleri için çalışan tek bir makine öğrenmesi modelinin olmadığını belirtmiştir. Geliştirdiği derin öğrenme yöntemi ile ikili duygu sınıflandırması üzerine de odaklan çalışmada Yelp film yorumları veri setinden 130 bin eğitim örneği alarak 10 bin test verisi üzerinde %4,63 hata oranı ile yüksek başarı elde etmiştir. Karakter Seviyeli Konvolüsyon Ağları olarak geçen bu model başka bir çalışmada [230] Çin veri setleri üzerinde test edilmiştir ve en düşük %4,66 hata oranı ile metin sınıflandırılması gerçekleştirilmiştir. Ayrıca yapılan çalışmada veri seti ne kadar büyükse CNN mimarisinin metin sınıflandırmada o kadar iyi çalıştığı belirtilmiştir.

6.4. Derin Öğrenme Yaklaşımı

İnsan beyninin yapısını taklit eden derin öğrenme, sinir ağlarının bir uyarlaması ve makine öğrenmesinin alt alanlarından birisidir. Fakat sinir ağlarını içeren derin öğrenme, herhangi bir makine öğrenmesi algoritmasından çok daha fazla önem arz etmektedir. Çünkü makine öğrenmesi ve insan öğrenmesi arasında öğrenme süreci bakımından büyük bir fark vardır.

Makine öğrenmesi ile bir makineye sınıflandırılacak nesnelerin özelliklerini söyleme ihtiyacında bulunuruz. Makine öğrenmesinin bu şekilde özellik mühendisliği gerçekleştirememesi en zayıf noktasıdır ve bu zayıf nokta, yapay zekâya erişmede büyük bir engel olarak görülmektedir [211]. Derin öğrenme ile birlikte bu engel aşılarak bir makine, kendi başına nesnelerin özelliklerini öğrenebilmektedir ve böylece derin öğrenme yenilikçi bir teknik olarak kabul edilmektedir. Bu yenilikçi yaklaşımın son yıllarda popülerlik kazanma sebepleri şu maddelerle özetlenebilir:

- Son yıllarda, Grafik İşleme Birimleri (GPU) güçlü ve verimli hesaplama, ucuz maliyette daha hızlı işlemler gerçekleştirme olanağı sağlamıştır.

- Eğitim amaçlı kullanılan veri setlerinin boyutları önemli derecede artmıştır. Bu artış ile birlikte erişilebilir büyük veri tabanları sunulmuştur.
- Makine öğrenimi, veri bilimi ve bilgi işlemedeki son araştırmalar önemli ilerlemeler göstermiştir.

Derin öğrenme, görüntü ve metin gibi giriş verileri ve sonuç bilgileri arasındaki ilişkileri değerlendirmek için çok katmanlı-derin bir mimari sunmaktadır. Bu derin mimari, ayrı bir adımda özellik çıkarımı gerçekleştirmek yerine tek bir öğrenme algoritması ile çok sayıda değişkeni işleyebilir ve özellik çıkarımı yapabilir. Her biri daha az nörona sahip daha fazla katman daha güvenilir ve doğru gösterimler elde etmektedir.

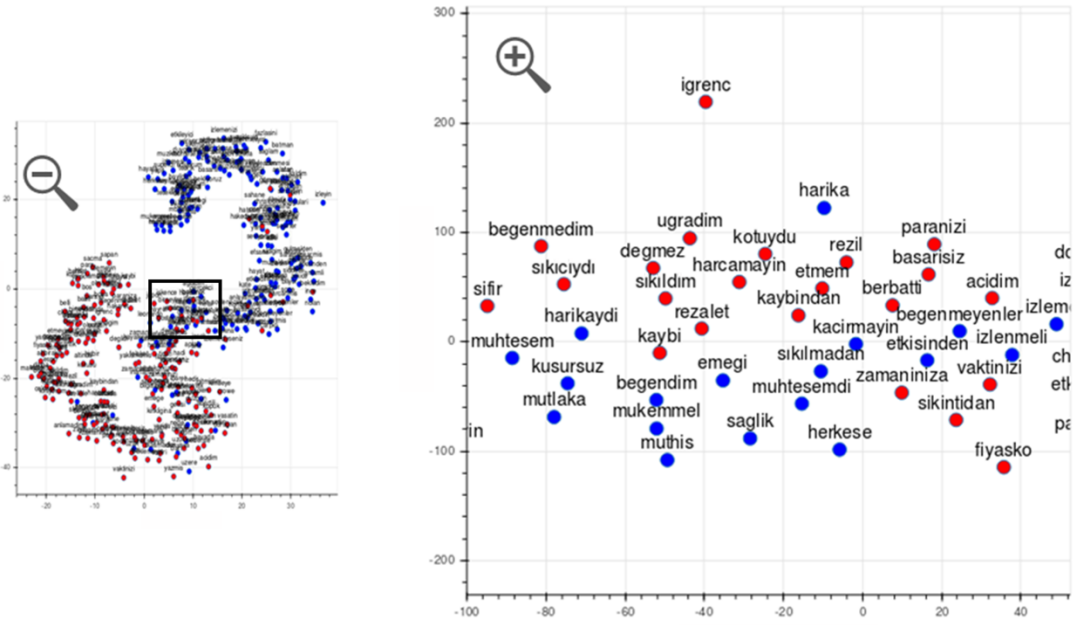
Bu bölüm kapsamında metin sınıflandırma uygulamaları için kullanılacak olan derin öğrenme mimarileri anlatılacak olup, bu mimariler ile oluşturulan modellerin testleri performans sonuçları ve tartışma bölümünde ayrıntılı olarak verilecektir.

6.5. Veri Seti ve Geliştirme Ortamları

Bu çalışmada yapılan son eğitim ve testler önceki çalışmalarla süre performansı açısından doğru kıyaslanabilmesi ve veri güvenliği açısından Ubuntu 14.04 işletim sistemi yüklü, Intel Xeon X5570 2.93-GHz işlemciye ve 85 GB hafızaya sahip bir web sunucu üzerinde gerçekleştirilmiştir. Modellerin oluşturulması ve kodlanması aşamalarında ise Amazon EC2 üzerinde çalışan Deep Learning AMI, Anaconda dağıtımı ve Colab ortamları gibi çeşitli derin öğrenme uygulama platformları kullanılmıştır.

Kelime vektörlerinin eğitimi için 4.Bölümde anlatılan ve bu tez çalışması için oluşturulan BEYAZPERDE veri seti kullanılmıştır. Önceden eğitilmiş kelime temsillerini oluşturmak için bu veri setinin tamamı kullanılırken, derin öğrenme modellerinin eğitimi için 40,617 film yorumu alınmıştır. Daha sonra bu eğitim seti rastgele olarak 32,493 eğitim ve 8,124 doğrulama setine ayrılmıştır. Veri setinde 0 ve 1 puanlı yorumlar negatif ve 5 puanlı yorumlar pozitif olarak etiketlenmiştir. Eğitilen pozitif ve negatif kelime vektörlerinin iki boyutlu t-SNE görselleştirmesi Şekil 6.1'de sunulmuştur. Kırmızı renkli noktalar pozitif kelimeleri ve mavi

noktalar negatif kelimeleri göstermektedir. *t-SNE* [231], derin sinir ağı ile eğitilen bir temsil uzayındaki kelimelerin birbirine yakınlıklarını ya da komşuluklarını, daha açık ifade ile birbirinden ne kadar uzak olduğunu iki boyutta görselleştirmek için kullanılan etkili bir tekniktir



Şekil 6. 1. Veri setindeki pozitif ve negatif kelime vektörlerinin t-SNE görseli

Veri seti içerisindeki pozitif ve negatif kelimelerin frekans dağılımlarına bakmak amacıyla öncelikle pozitif, negatif ve toplam olmak üzere üç sayaç tutulmuştur. Pozitif etiketli yorumlardaki kelimeler pozitif sayaçta, negatif kelimeli etiketler negatif sayaçta ve tüm kelimeler toplam sayaçta tutulmuştur:

```
from collections import Counter
import numpy as np
pozitif_sayac = Counter()
negatif_sayac = Counter()
toplam_sayac = Counter()
for i in range(len(yorumlar)):
    if(etiketler[i] == 'pozitif'):
```

```
            for kelime in yorumlar[i].split(" "):
                pozitif_sayac[kelime] += 1
                toplam_sayac[kelime] += 1
        else:
            for kelime in yorumlar[i].split(" "):
                negatif_sayac[kelime] += 1
                toplam_sayac[kelime] += 1

print(pozitif_sayac.most_common())
```

çıktı:
```
[('bir',25774), ('film',18244), ('cok',12432), ('filmi',7662),
('iyi',6147), ('guzel',5655), ('kadar',4801), ('bi',4096),
('bence',3993), ('filmin',3781), ('icin',3364),
('gercekten',3058), ('harika',2808), ('kesinlikle',2694),
('mukemmel',2594),…]
```

Yukarıdaki kod parçasında çıktısı alınan pozitif sayaç içeriğine bakıldığında *mükemmel, güzel, harika* gibi istenen pozitif kelimelerin yanında *bir, film, bence, kadar* gibi pozitif yorumlarda geçen ama pozitif kelime listesinde istenmeyecek kelimeler mevcuttur. Bu problemi çözmek amacıyla yani pozitif sayacımızda yalnızca pozitif kelimeleri tutmak amacıyla her kelimenin pozitif sayaçta bulunma sayısı negatif sayaçta bulunma sayısına oranlanır ve pozitif oran bulunur. Bunun sonucunda belirlenen eşik pozitif oran değerinin üstündekiler daha pozitif kelimeleri gösterir. Örneğin kullanılan veri setinde *mükemmel* kelimesinin pozitif etiketli yorumlarda geçme sayısı 2544, negatif etiketli yorumlarda geçme sayısı 208'dir. Aşağıdaki kod parçasında görüleceği gibi pozitif-negatif oranı 12,4… olarak hesaplanır ve bu oran tüm kelimelerde ne kadar yüksek ise o kadar pozitif kelimeye daha yakındır denilmektedir.

```
pozitif_sayac["mukemmel"]
negatif_sayac["mukemmel"]
pozitif_oran=pozitif_sayac["mukemmel"]/
             float(negatif_sayac ["mukemmel"]+1)
print(pozitif_oranı)
çıktı: 12.411483253588516
```

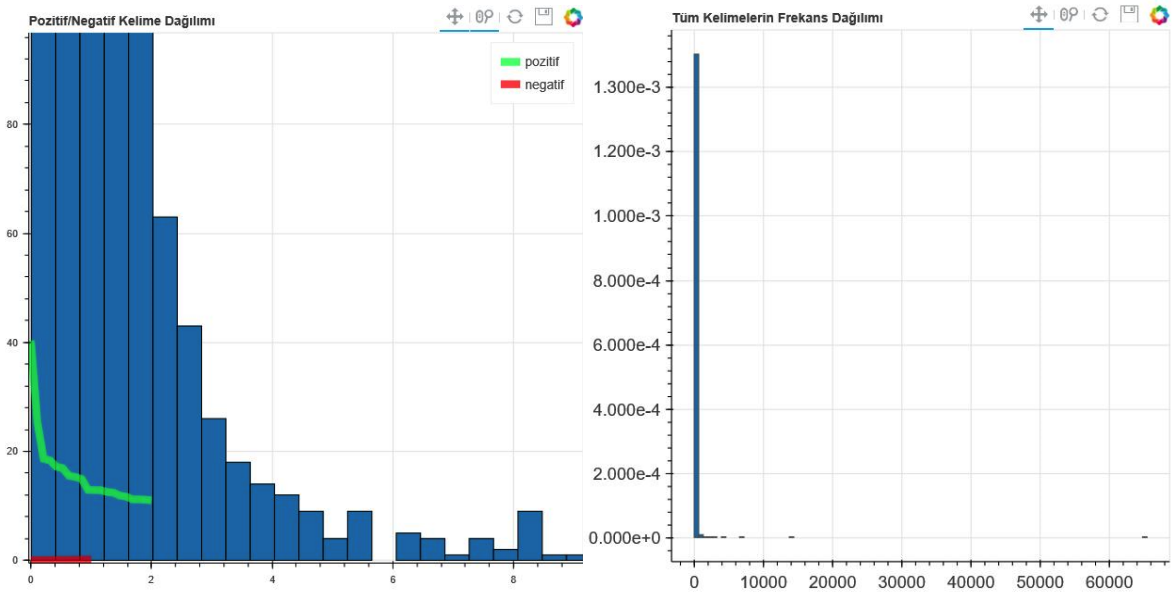
Tüm pozitif kelimeler için aşağıdaki hesaplamalar gerçekleştikten sonra pozitif sayacın çıktısı şu şekilde sonuçlanır:

```
print(pozitif_sayac.most_common())
[('izlemelisiniz',40.333),('izlenmeli',25.4375),('bayildim',18.642
857142857142),('harikaydi',17.2),('harika',16.91566265060241),('mu
tlaka',15.527472527472527),('kusursuz',15.31578947368421),('kacirm
ayin',14.91304347826087),('dortluk',12.888),('favori',12.875),('mu
kemmel',12.411483253588516),('muhtesemdi',11.88888888888889),('sag
lik',11.0),('muthis',10.157407),('begendim',9.837209302325581),('e
tkisinden',9.833333333333334),('muhtesem',9.691358024691358),('sik
ilmadan',9.615384615384615),('etkilendim',9.5625),…]
```

Görüldüğü gibi pozitif kelimelerin tutulduğu sayaç gürültüden temizlenmiştir. Benzer şekilde negatif kelimeler de tersi hesaplama ile bulunmaktadır. Negatif sayaç çıktısı da şu şekildedir:

```
[('kirikligi',0.089595),('kotusu',0.0901),('vasat',0.09961),('zama
ninizi',0.10526315789473684),('kalitesiz',0.1067961165048),'gereks
iz',0.1145124716553288),('yapmacik',0.116),('kopuk',0.116822429906
54206),('ucuz',0.11695906432748537,('amator',0.135416),('izlemeyin
',0.1396761133603239),('etmez',0.14093959731543623),('yakismamis',
0.14285714285714285),('bosuna',0.14516129032258066),('hatrina',0.1
5306122448979592),('abartilmis',0.16494845360824742),('sacmalik',0
.165),('para',0.1673819742489),('goremedim',0.17592592593),('harca
nan',0.17647058823529),('desen',0.1846153846),('basit',0.209178228
38847386),('sacma',0.21299),
```

Şekil 6.2'de pozitif ve negatif kelimelerin dağılımları (soldaki) ve veri setindeki tüm kelimelerin frekans dağılımı (sağdaki) Bokeh görselleştirme kütüphanesi [232] kullanılarak gerçekleştirilmiştir.



Şekil 6. 2. Veri setindeki kelime dağılım görselleştirmesi

Bu tez bölümünün ikinci aşamasında gerçekleştirilen haber sınıflandırma çalışması için HABERLER veri seti (indirme linki: Fırat Üniversitesi Büyük Veri ve Yapay Zeka Laboratuvarı, http://buyukveri.firat.edu.tr/) kullanılmıştır. Bu tez çalışması kapsamında literatüre katkı olarak sunulan haber veri seti, Türkçe çoklu metin sınıflandırma çalışmalarında doğru ve güvenilir karşılaştırmalar yapmak için oluşturulmuştur.

Çeşitli haber sitelerinden (www/log.com.tr, webtekno.com, cumhuriyet.com.tr, diken.com.tr, evrensel.net, haberler.com, internethaber.com, sabah.com.tr, turkiyegazetesi.com.tr, aa.com.tr, star.com.tr, dha.com.tr, shiftdelete.net) Java tabanlı web kazıma aracı ile toplanan yaklaşık 10 milyon haber ve 1,129,196 benzersiz kelime içeren veri seti ile skip-gram word2vec algoritması kullanılarak kelime temsilleri eğitilmiştir. Google haber veri setinden, Google tarafından eğitilen ve yaklaşık 3 milyon benzersiz kelime içeren kelime vektörleri İngilizce dili için geliştirilmiştir. Türkçe metin sınıflandırma için de önceden eğitilmiş kelime vektörlerine ihtiyaç duyulduğundan dolayı, bu tez çalışmasında çok büyük haber verileri üzerinde eğitim gerçekleştirilmiştir. Kullanılan derin öğrenme modellerinin eğitim ve testi için ise bu veri setinden ekonomi, politika, kültür, dünya ve spor etiketli 500.000 eğitim seti

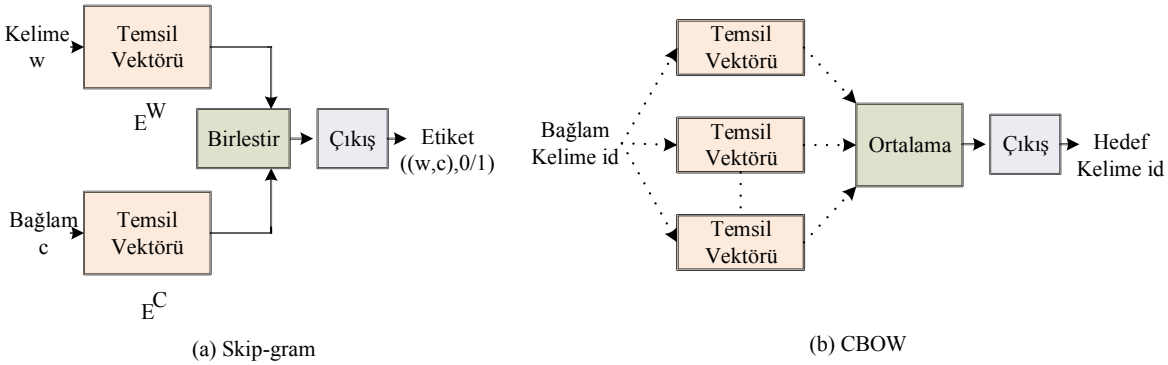
oluşturulmuştur. Haber veri seti üzerinde yapılan çoklu metin sınıflandırma çalışmasının performans sonuçları bu bölümün sonunda verilmiştir.

6.6. Derin Öğrenme Modelleri

Derin öğrenme modellerinin oluşturulması ve test edilmesi için Python dilinde geliştirilen Tensorflow ve Keras kütüphaneleri tercih edilmiştir. Hesaplama karmaşıklığını azaltma ve kullanım kolaylığı sağlama gibi özellikler ile ön plana çıkan bu açık kaynak kütüphane ve platformlar başarılı sonuçlar vermiştir. Bu çalışmada aşağıdaki algoritma ve modellerin karşılaştırılması ve değerlendirilmesi yapılmıştır:

6.6.1. Önceden Eğitilmiş Kelime Temsilleri (PWE)

Kelime temsilleri metin sınıflandırmasında, belge kümelemede, konuşma etiketlemede, varlık tanımlamada, duygu analizinde ve diğer NLP görevlerinde metinleri bir vektöre dönüştürmek için tercih edilen bir tekniktir. Bu dağıtık gösterimler, sözcükler arasındaki anlamsal benzerliği vektörler arasındaki benzerlikle ilişkilendirmek için kelimeleri vektörlere dönüştürmeyi amaçlamaktadır. Word2vec ve GloVe en iyi bilinen kelime temsilleridir.



Şekil 6. 3. Word2vec modelleri

Duygu analizi ve haber sınıflandırma gibi metin sınıflandırma görevlerinde word2vec ile oluşturulmuş kelime temsilleri model performansını arttırmak için kullanılmaktadır. Word2vec modelinin, Şekil 6.3'te gösterildiği gibi skip-gram (a) ve CBOW (b) olmak üzere iki mimarisi vardır. Daha önceki bölümde anlatıldığı gibi skip-gram modeli sık geçmeyen kelimelerin tahmininde daha iyi çalıştığı için bu tez bölümünde kelime temsillerinin eğitiminde tercih edilmiştir.

Skip-gram modeli, verilen merkezi bir kelimenin yakınındaki kelimeleri, yani bağlam kelimelerini tahmin etmeye çalışmaktadır. Şekil 1'de bir kelime vektörü ve bir bağlam vektörü alan ve negatif ya da pozitif örneğe bağlı olarak 1 ya da 0 sayılarını tahmin etmek için öğrenen sınıflandırıcı eğitilmiştir [233]. Gösterilen skip-gram word2vec modelde, her kelime $w \in W$ olmak üzere bir kelime vektörü (word embedding) $E^W \in R^D$ ile gösterilir ve benzer şekilde her bağlam $c \in C$ olmak üzere bir bağlam vektörü (context embedding) $E^C \in R^D$ olarak temsil edilmektedir. Burada, D vektör boyutu (embedding size), W kelimeler ve C bağlamlar sözlüğüdür. Kelime ve bağlamların vektör gösterimleri arasında iç çarpım işlemi gerçekleştirilmektedir. Bu eğitilen ağın çıktıları, görüldüğü gibi kelime temsili katmanının ağırlıklarıdır. Skip-gram modeli çok düşük hesaplama karmaşıklığı nedeniyle, çok büyük veri kümesinden çok doğru yüksek boyutlu kelime vektörleri hesaplayabilmektedir [228].

Kelime temsilleri, derin sinir ağlarının eğitimi için önemli bir girdidir. Performansı arttırmak ve daha doğru sınıflandırma sonuçları elde etmek için, bu çalışmada 100 boyutlu kelime vektörleri kullanılarak derin öğrenme modelleri eğitilmiştir. Önceden eğitilen bu kelime vektörlerinin performans etkisi detaylı sonuç grafikleriyle araştırılmıştır. Kullanılan kelime vektörleri film yorumlarının duygu sınıflandırması için skip-gram word2vec algoritması ile elde edilmiştir. Kelime vektörleri oluşturmak için parametre seçiminde Lai ve arkadaşlarının [234] yaptıkları kelime vektör oluşturma çalışması rehber alınmıştır.

6.6.2. Çok Katmanlı Algılayıcılar (MLP) Modeli

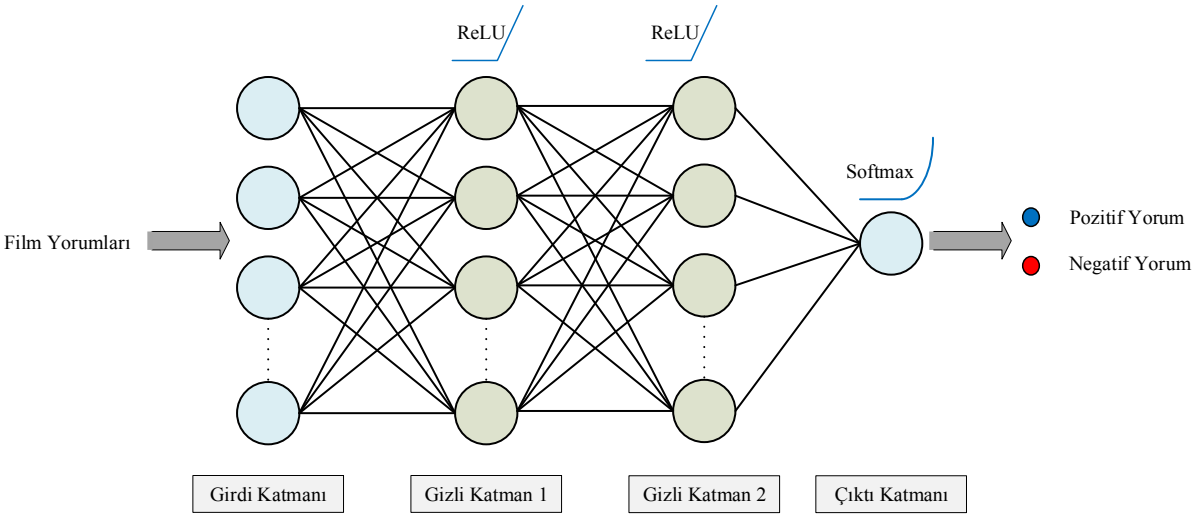
Lojistik regresyon sınıflandırıcısının bir versiyonu olarak da düşünülebilen MLP, en az bir gizli katmanı olan temel ileri beslemeli yapay sinir ağı türüdür. MLP yeni bir sinir ağı mimarisi değildir, derin öğrenme çağından önce bile metin sınıflandırma problemleri için makine öğrenimi uygulamalarında yaygın olarak kullanılmıştır [235, 236].

MLP katmanlarına, girdi verilerinin özellik değerlerinden çıktı nöronlarına doğru bir yönde başlayan bir dizi işlem uygulanır. Matematiksel formül ile tek gizli katmanlı bir MLP mimarisini gösterecek olursak, D girdi boyutu ve L çıktı boyutunu temsil eden $f: R^D \rightarrow R^L$ fonksiyonu şartında x girdi özelliklerinin bir vektörü ise , $f(x)$ aşağıdaki gibidir [40]:

$$f(x) = G(b^{(2)} + W^{(2)}\left(s\left(b^{(1)} + W^{(1)}x\right)\right)) \tag{6.1}$$

Burada $b^{(1)}, b^{(2)}$ bias vektörleri ve s, G aktivasyon fonksiyonlarıdır.

Bu çalışmada sunulan MLP modelde duygu sınıflandırması için Tensorflow üzerinde çalışan basit bir sinir ağı oluşturulmuş ve eğitilmiştir. Model bir girdi katmanı, iki gizli katman ve bir çıktı katmanından oluşmaktadır. Öncelikle film yorumları kelime vektörlerine dönüştürülür ve kelime torbaları kullanılarak sözlük (vocabulary) oluşturulur. Ağın girdisi olarak 30.000 sık geçen kelime kullanılır ve tahminlere çok az etkisi olduğu için sözlükteki yaygın kullanımı olmayan kelimeler ihmal edilir. Bu sayının belirlenmesi için, sözlük üzerinde denemeler gerçekleştirilmiş ve veri seti için en iyi girdi kelime sayısı bulunmuştur. Örneğin veri setinde 5 seferden daha az geçen kelimelerin genellikle yanlış yazım hataları olan kelimelerden oluştuğu gözlemlenmiştir. Girdi katmanından sonar ReLU aktivasyon fonksiyonu ile ağ için iki gizli katman eklenmiştir. Son olarak giriş cümlesinin pozitif ya da negatif sınıfa ait olup olmadığını tahmin etmek için Softmax aktivasyon fonksiyonu ile çıktı katmanında iki çıktı birimi kullanılmıştır. Bu ağ yapısı Şekil 6.4'te gösterildiği gibidir:

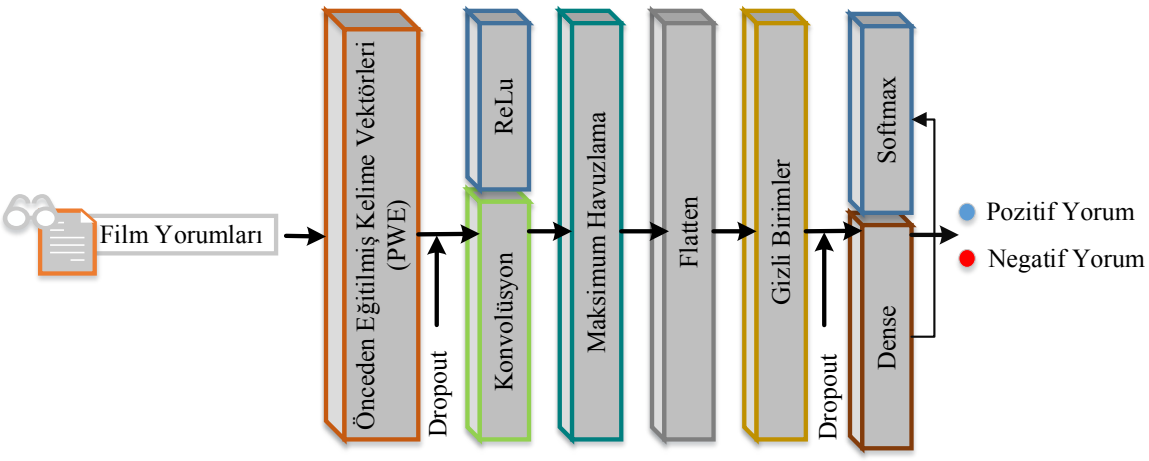


Şekil 6. 4. MLP model yapısı

6.6.3. Konvolüsyonel Sinir Ağları (CNN) Modeli

Literatürde kısaca ConvNet ya da CNN olarak adlandırılan Konvolüsyonel Sinir Ağları, görüntü tanıma ve metin sınıflandırması görevlerinde en popüler derin öğrenme modellerinden birisidir. Pek çok görüntü veya metin tabanlı makine öğrenimi problemlerini çözmede başarılı sonuçlar vermektedir. Girdi özellikleri olarak resimlerde pikseller ve metinler kelimeler kullanılarak verilerin gösterimleri elde edilmektedir.

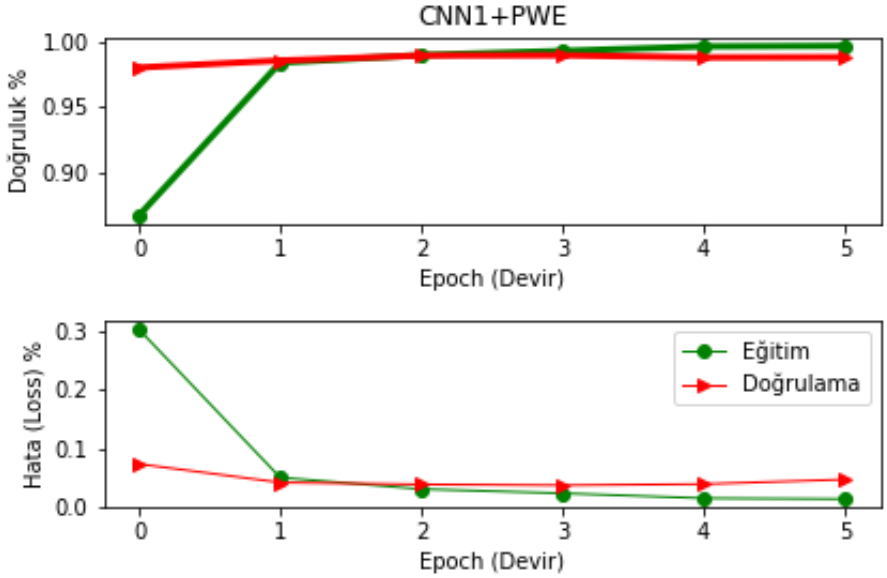
CNN mimarisi MLP modelde olduğu gibi bir dizi katmandan oluşur fakat modeldeki gizli katmanların türleri ile MLP modelden ayrılmaktadır. CNN modeli ya da LeNet [12] oluşturmak için üç temel katman türü kullanılmaktadır: Konvolüsyon Katman, Havuz Katmanı ve Tam Bağlantılı Katman. Havuzlama katmanında iki çeşit havuzlama (pooling) işlemi yapılabilir: ortalama havuzlama ve maksimum havuzlama. Türkçe film yorumları üzerinde duygu sınıflandırması için, literatürdeki çalışmalardan [225,162] esinlenerek elde edilen tek boyutlu CNN modeli Şekil 6.5'te verilmiştir.



Şekil 6. 5. CNN model mimarisi

Bu modelde öncelikle bir film yorumunu içeren cümledeki her bir kelime, vektör gösterimi ile değiştirilerek $M \in R^{SXD}$ olarak ifade edilen bir cümle matrisi elde edilir. Burada S maksimum cümle uzunluğunu ve D kelime vektör boyutunu temsil etmektedir. Filtre boyutu, bir seferde cümle matrisinde kaç kelime kaydırılacağını belirtmektedir. Konvolüsyona tabi tutulan tüm değerlere sıfırda eşik değeri olan $f(x) = max(0, x)$ ReLU aktivasyon fonksiyonu uygulanır. Sonuç olarak çekirdek olarak da adlandırılabilen filtre sayısı kadar özellik haritası elde edilir. Her bir özellik haritasına maksimum havuzlama işlemi uygulanarak en belirgin bilgiler çıkarılır. Tam bağlantılı katman, çıkış olarak pozitif ve negatif şekilde iki sınıf puanı hesaplamaktadır.

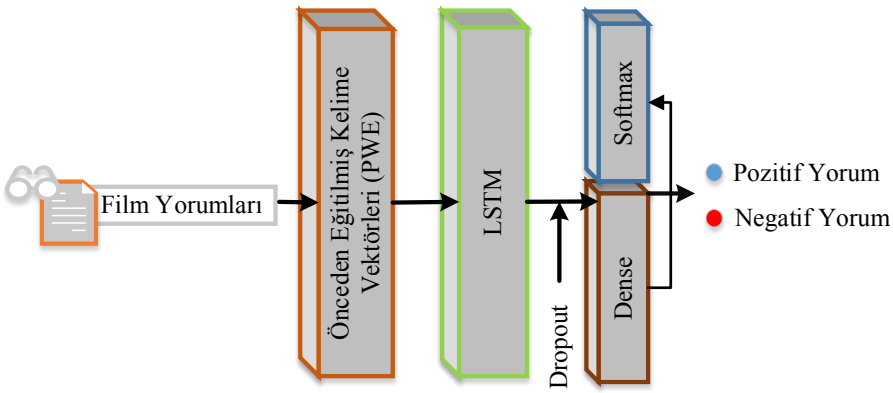
Veri setine ve model türüne göre seçilen en iyi hiper parametre değerleri ile Şekil 6.5'te sunulan CNN modelin eğitim ve doğrulama sonuçları Şekil 6.6'da verilmiştir. Şekil 6.6'da, tek bir konvolüsyon ve havuzlama katmanı içeren CNN1 modelin eğitim ve doğrulama aşamalarında her bir devir sayısı için doğruluk ve başarısızlık oranlarını göstermektedir. Bu modelde PWE kullanıldığı için CNN1+PWE olarak adlandırılacaktır. Model, tahmin edilen ve gerçek çıktı değerleri arasında ortalama karesel hatayı en aza indirgeyerek eğitilmiştir. Bu eğitim sonucunda elde edilen başarısızlık değerleri görülmektedir. Bu çalışmada kullanılan ve CNN2 olarak adlandırılacak olan diğer CNN mimarisi iki konvolüsyon ve havuzlama katmanından oluşmaktadır. CNN2 mimarisinin önceden eğitilmiş kelime vektörleri kullanan modeli de CNN2+PWE olarak geçmektedir.



Şekil 6. 6. CNN modelinin eğitim ve doğrulama sonuçları (temsil boyutu=100, batch boyutu =256, maksimum cümle uzunluğu=300, filtre sayısı=64, filtre boyutu=3x3)

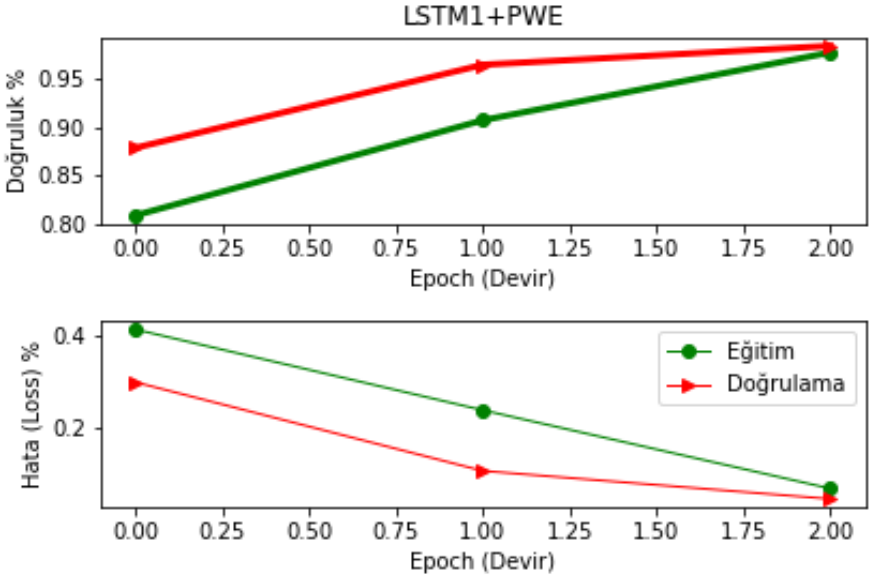
6.6.4. Uzun Kısa Süreli Bellek Ağları (LSTM) Modeli

Derin sinir ağları uygulamalarında ezberleme-aşırı öğrenme ve sıfırlanan gradyan (vanishing gradient) iki temel problemdir. Gizli katmandaki parametre sayısının artışından kaynaklı ezberleme problemini engellemek için *dropout* yöntemi kullanılmaktadır [237]. NLP'de yaygın olarak kullanılan RNN yöntemi, bir metin içerisinde verilen bir önceki kelimelerden bir sonraki kelimeyi tahmin edebilmektedir. Geleneksel sinir ağları gibi RNN de geriye yayılım algoritması kullanmaktadır. Bu geriye yayılım süresince gradyanlar, sıfırlanan gradyan problemi adı verilen sıfıra eğilim göstermektedir. Bu problemin çözümü için LSTM mimarisi sunulmuştur [174]. Derin sinir ağlarında karşılaşılan bu problemleri dikkate alarak bu çalışmada, farklı katmanlarda LSTM mimarisinin ve dropout eğitiminin Türkçe film yorumlarının sınıflandırılmasındaki performansı tartışılmıştır.



Şekil 6. 7. LSTM model mimarisi

Şekil 6.7, bu çalışmada kullanılan tek katmanlı LSTM1 modelinin yapısını göstermektedir ve Şekil 6.8'de en iyi hiper parametre seçimiyle kullanılan bu modelin doğruluk ve hata grafiklerini sunmaktadır. LSTM katmanı çoktan-bire bir RNN eğiterek verilen bir film yorumunun pozitif mi negatif mi olduğunu bulmaya çalışmaktadır. Pozitif ve negatif olma durumuna göre sırasıyla 1 ve 0 olarak etiketlenen farklı uzunluktaki yorumların uzunlukları 100 kelime ile sınırlandırılmıştır. Daha uzun yorumlar maksimum cümle uzunluğuna göre kesilirken daha kısa yorumlar sıfır-doldurma (zero-padding) işlemi ile yani gerekli miktarda sıfırlarla doldurulmaktadır. Aşırı öğrenmeyi engellemek için bu katmandan sonra bir dropout kullanılmıştır. Şekil 6.8, LSTM1 modelinin eğitim ve doğrulama aşamalarında her bir devir sayısı için doğruluk oranının artışına karşı başarısızlık oranının nasıl azaldığını göstermektedir. Bu modelde PWE kullanıldığı için LSTM1+PWE olarak adlandırılacaktır. LSTM2 olarak adlandırılacak olan diğer mimarisi iki LSTM katmanından oluşmaktadır ve PWE kullanılmış modeli LSTM2+PWE olarak isimlendirilmiştir.



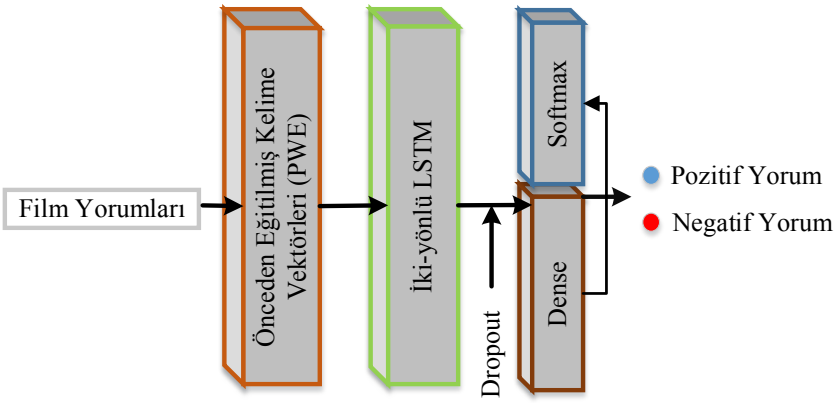
Şekil 6. 8. LSTM modelinin eğitim ve doğrulama sonuçları (temsil boyutu=100, batch boyutu=256, maksimum cümle uzunluğu=300, gizli katman boyutu=128)

6.6.5. İki-yönlü Uzun Kısa Süreli Bellek (BiLSTM) Modeli

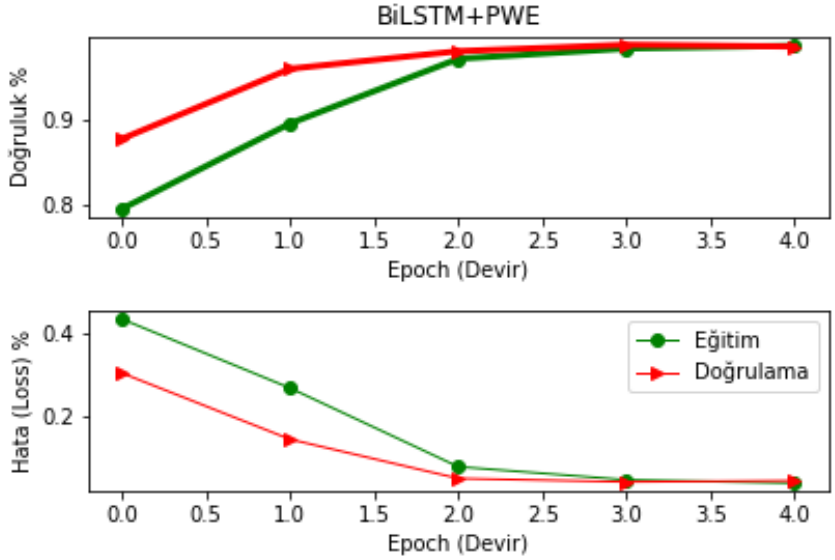
BiLSTM, hem pozitif hem de negatif zaman yönlerinde eğitim için LSTM mimarisinin bir modifikasyonudur. Diğer bir deyişle, girişin eş zamanlı olarak hem sırayla hem de ters sırada taranabildiği LSTM modelinin geliştirilmiş bir sürümüdür [173]. BiLSTM, ileri ve geri olmak üzere iki yönde yayılan iki paralel katmana sahiptir [238]. Böyle bir ağın final çıktısı bu iki katmanın çıktısının birleşimi olacaktır. ${h_F}^{(t)}$ ileri yayılım katmanının gizli durumu ve ${h_B}^{(t)}$ geri yayılım katmanının gizli durumu olmak üzere, BiLSTM ağının gizli durumu olan $h^{(t)}$ şu şekilde hesaplanır:

$$y_t = [{h_F}^{(t)} \oplus {h_B}^{(t)}] \tag{6.2}$$

Burada y_t her bir zaman adımının çıktısıdır. Bu gizli durum değerine doğrusal bir katman ve bir Softmax katman uygulanır [239].



Şekil 6. 9. BiLSTM model mimarisi



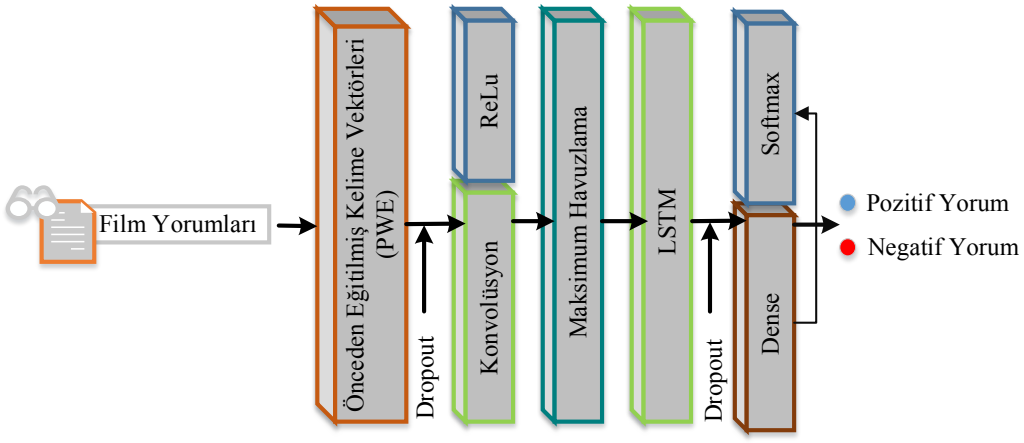
Şekil 6. 10. BiLSTM modelinin eğitim ve doğrulama sonuçları (temsil boyutu=100, batch boyutu=256, maksimum cümle uzunluğu=300, gizli katman boyutu=128)

BiLSTM modelinde iki yönde iki gizli paralel katman vardır, ileri ve geri geçişler bu katmanlardan yayılmaktadır. Bu çalışmada kullanılan BiLSTM modelde, kelime temsillerinden giriş yorumlarını okuyan ağın üst kısmında yığılmış ters yönde çalışan iki LSTM katman vardır. Bir LSTM soldan sağa kelimeleri okurken, diğer LSTM sağdan sola doğru okumaktadır. Her bir film yorumundan çıkarılan özellikler, ileri ve geri LSTM ağları üzerinden geçirilir ve

Softmax aktivasyon fonksiyonu kullanılarak sınıflandırma için çıktı katmana yönlendirilir. Softmax katmanından önce aşırı öğrenmeyi engellemek için dropout eklenmiştir. Şekil 6.9'da oluşturulan BiLSTM model ve Şekil 6.10'da en iyi hiper parametre seçimlerine göre eğitilen bu modelin eğitim ve doğruluk sonucu verilmiştir. Model sonucundan görüleceği üzere BiLSTM+PWE modeli aşırı öğrenme problemi yaşamadan veri setindeki yorumları güvenilir bir şekilde öğrenmiştir.

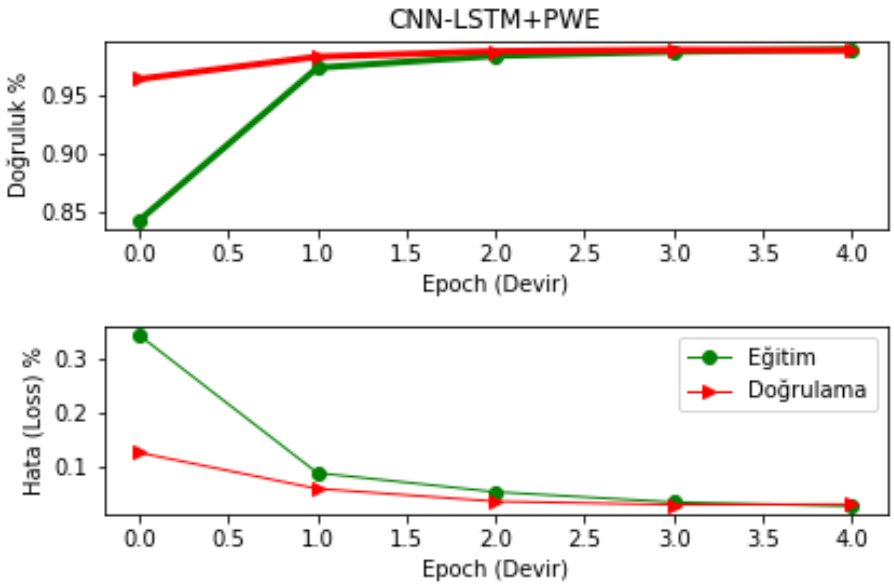
6.6.6. CNN-LSTM Modeli

CNN modeli, lokal özellikleri çıkarmada iyi olmasına karşın, sıralı bilgileri öğrenmede zayıf kalmaktadır. Bu problemi çözmek için kullanılan CNN-LSTM modeli Şekil 6.11'de ve bu modelin en iyi parametre seçimlerine göre veri seti üzerindeki doğruluk ve hata sonuç grafikleri Şekil 6.12'de verilmiştir. Derin sinir ağı eğitilirken, gerçek çıktının eğitilen ağın çıktısından ne kadar uzak olduğunu ölçmek için bir gösterge görevi gören başarısızlık fonksiyonu (loss function) kullanılmaktadır. Bu ağın eğitiminde başarısızlık oranlarının tespiti için Kategorik Çapraz Entropi fonksiyonu kullanılmıştır. Zhou ve arkadaşlarının [224] sunduğu C-LSTM modelinden farklı olarak konvolüsyon katmanından sonra parametre sayısını azaltarak ağın hesaplama zamanını azalttığı için maksimum havuzlama katmanı eklenmiştir.



Şekil 6. 11. CNN-LSTM model mimarisi

Model, tek katmanlı hiyerarşik CNN ve sıralı LSTM mimarilerinin birleşiminden oluşmaktadır. CNN, giriş yorumlarının lokal özelliklerini öğrenir, LSTM ve uzun-süreli bağımlılıkları öğrenerek bu özellikleri sıralı bir şekilde işler. İlk olarak skip-gram word2vec algoritması kullanılarak bu çalışma kapsamında oluşturulan büyük film yorumları veri setinden, kelime vektörleri eğitilmiştir. Giriş olarak alınan yorumdaki (burada cümle diye adlandırıyoruz) her bir kelime, eğitilen bu kelime vektörleri ile temsil edilir. Böylece modelin ilk katmanı kelime temsilleridir. Konvolüsyon katmanında, önemli özellikler diğer bir ifade ile öz-nitelikler çıkartılır ve seçilen ilişkili filtreler ile bu özellikler üzerinde bire bir çarpım işlemi gerçekleştirilerek çoğaltılır. Havuz katmanında ise çoğalan bu öz-nitelikler maksimum havuzlama işlemi ile azaltılır. Havuzlama katmanında çoğalan öz-nitelikler maksimum havuzlama işleme ile küçültülür. LSTM katmanı, konvolüsyon ve havuzlama katmanı sonrası aldığı özellikleri öğrenir ve verilen yorumun pozitif veya negatif olduğunu tahmin eder. Sunulan bu modelde konvolüsyon katmanından önce ve LSTM katmanından sonra 0,3 değerinde dropout kullanılmıştır.



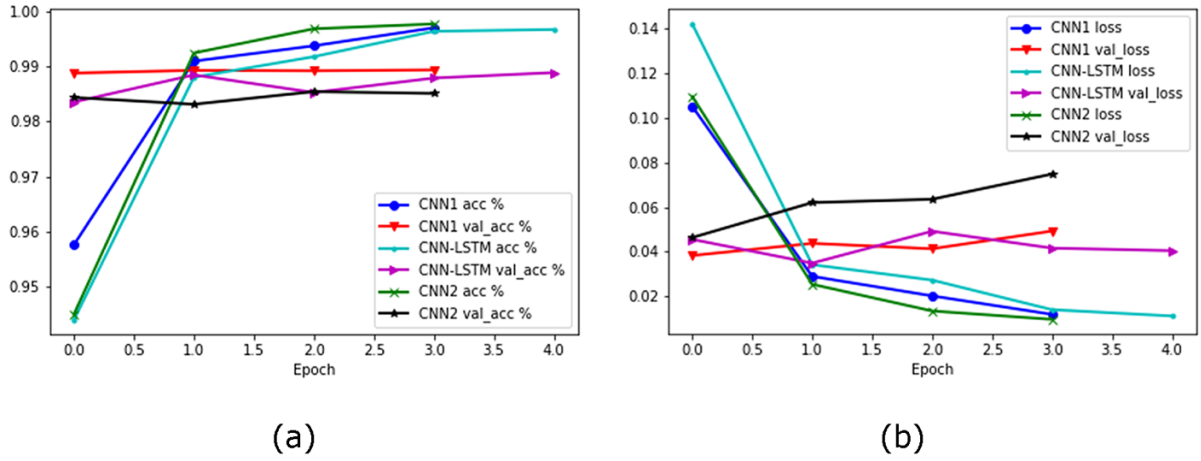
Şekil 6. 12. CNN-LSTM modelinin eğitim ve doğrulama sonuçları (temsil boyutu=100, batch boyutu=256, maksimum cümle uzunluğu=300, filtre sayısı=64, filtre boyutu=3x3, gizli katman sayısı=128)

6.7. Duygu Sınıflandırması Performans Sonuçları ve Tartışma

Bu tez bölümünde, değişen sayıda batch boyutu ve devir sayısı (epoch) hiper parametreler için çeşitli derin sinir ağları eğitilmiş ve test edilmiştir. Sunulan derin öğrenme modelleri 0,001 öğrenme katsayısı ve 32, 64, 128 ve 256 batch boyutları ile Adam optimizer [209] kullanılarak eğitilmiştir. Eğitim iterasyonlarının sayısı yani epoch sayısı erken durdurma tekniği kullanılarak otomatik olarak optimize edilmiş ve model eğitim sürecine göre belirlenmiştir. Erken durdurma (early stopping), doğrulama hatasını izleyerek model eğitiminin ne zaman durdurulacağını belirleyen ve Tensorflow ile desteklenen bir tekniktir. Bu şekilde, epoch sayılarını ayarlayarak hiper parametrelerin güçlü bir şekilde aşırı öğrenme problemine takılmasından kaçınmaya yardımcı olur.

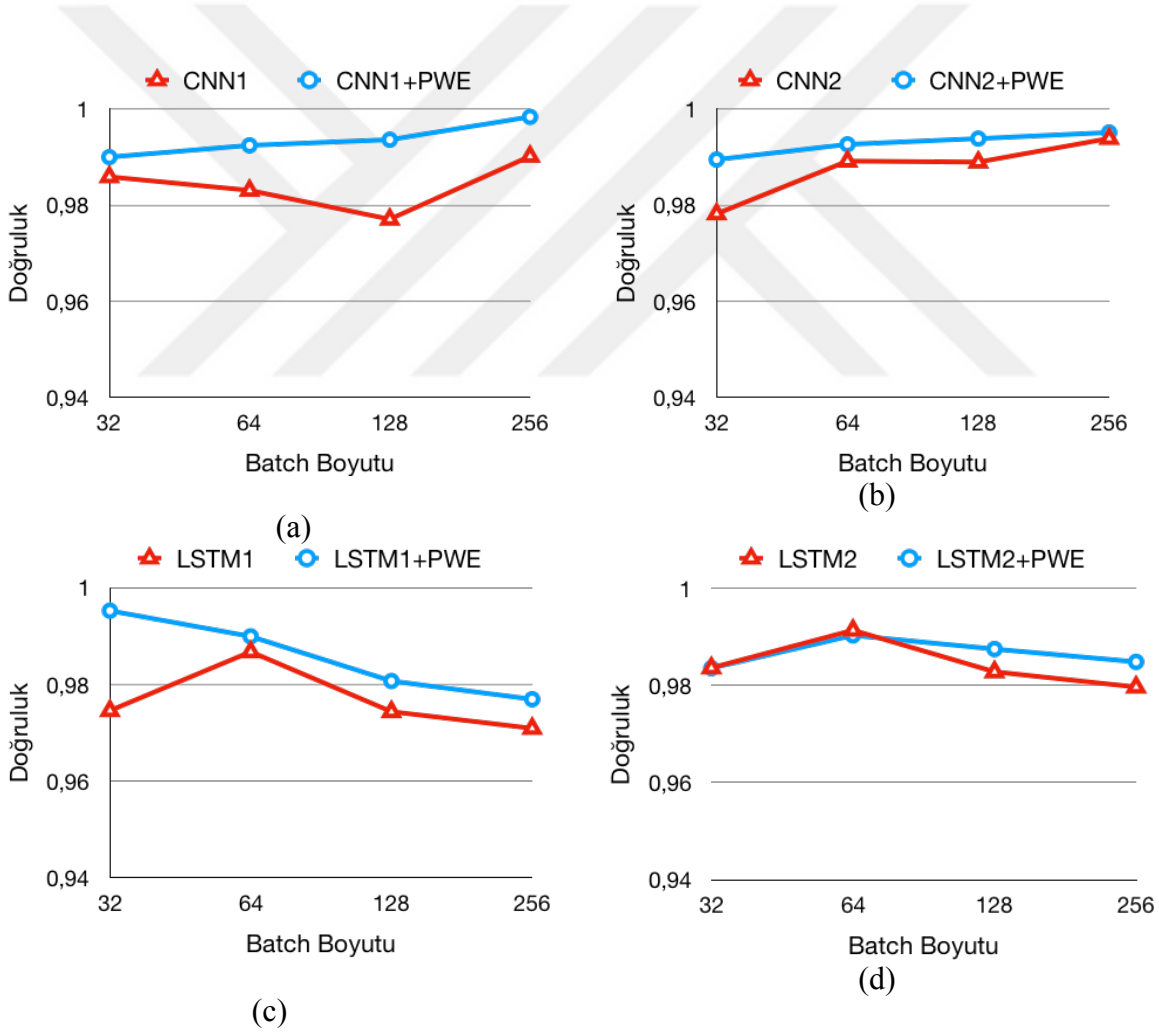
Eğitim ve doğrulama süreçleri sırasında, en iyi modele ulaşma çalışmalarında denenmiş modeller için bazı aşırı öğrenme durumları gözlemlenmiştir. Şekil 6.13'te eğitim aşamasında aşırı öğrenmeye maruz kalan modellerin hatalı doğruluk ve kayıp sonuçları verilmiştir. Eğitim ve doğrulama aşamaları için doğruluk değerlerinin birbirinden ayrıldığı ve aynı aşamalar için hata değerlerinin de birbirinden uzaklaştığı noktalar modellerin veriyi aşırı öğrendiğini göstermektedir. Örneğin mavi renkle gösterilen CNN1 eğitim adımı için doğruluk oranı ve kırmızı renkle gösterilen CNN1 doğrulama adımı için gösterilen doğruluk oranı 1.epoch sayısında hızla birbirinden uzaklaşmaya başlamıştır. Bu durum daha önceki bölümde anlatıldığı gibi aşırı öğrenmeyi işaret etmiştir. Benzer şekilde CNN1 eğitim sonuçlarından alınan hata oranı ve aynı modelin doğrulama sonuçlarından alınan hata oranı düşmek yerine yükselmeye başlamıştır. Şekil 6.13'te gösterilen eğitim sonuçları bu çalışma kapsamında denenmiş hatalı modellere aittir ve bu modeller hiper parametre ayarlamaları ile optimize edilerek iyi duruma getirilmiştir.

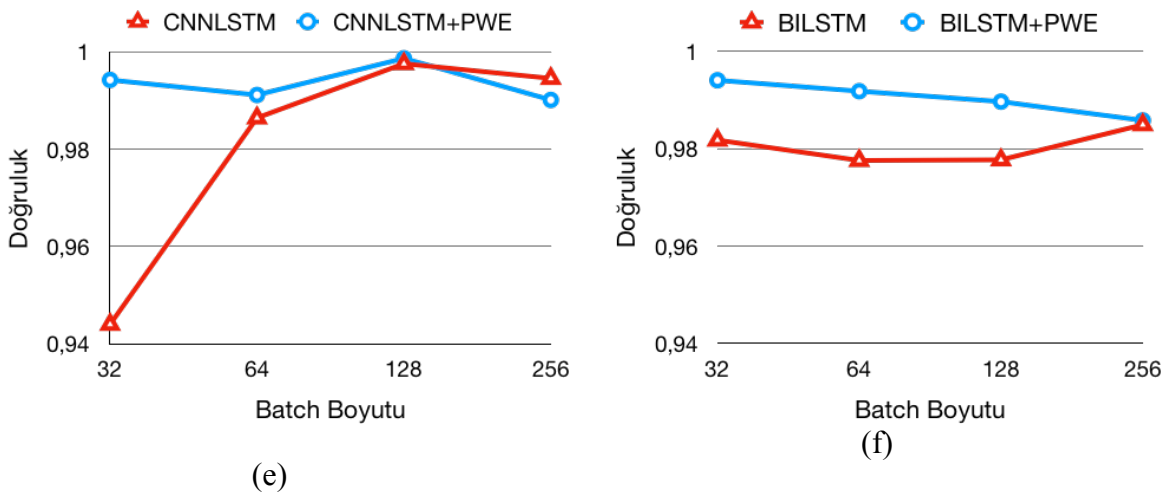
Önceden eğitilmiş kelime temsilleri ya da vektörleri (kısaca PWE diyoruz), dildeki sözdizimsel ve anlamsal düzenleri yakalayabilmektedir [240]. PWE ile derin sinir ağları NLP çalışmalarında başarılı sonuçlar elde etmektedir [241]. Bu tez çalışmasında da PWE ile çeşitli derin öğrenme modelleri test edilmiş ve metin sınıflandırma problemi üzerindeki etkileri analiz edilmiştir.



(a) (b)

Şekil 6. 13. Aşırı öğrenme problemine maruz kalmış hatalı modeller





Şekil 6. 14. (a) CNN1, (b) CNN2, (c) LSTM1, (d) LSTM2, (e) CNNLSTM, (f) BiLSTM modelleri için duygu sınıflandırma sonuçlarında PWE kullanımının etkileri

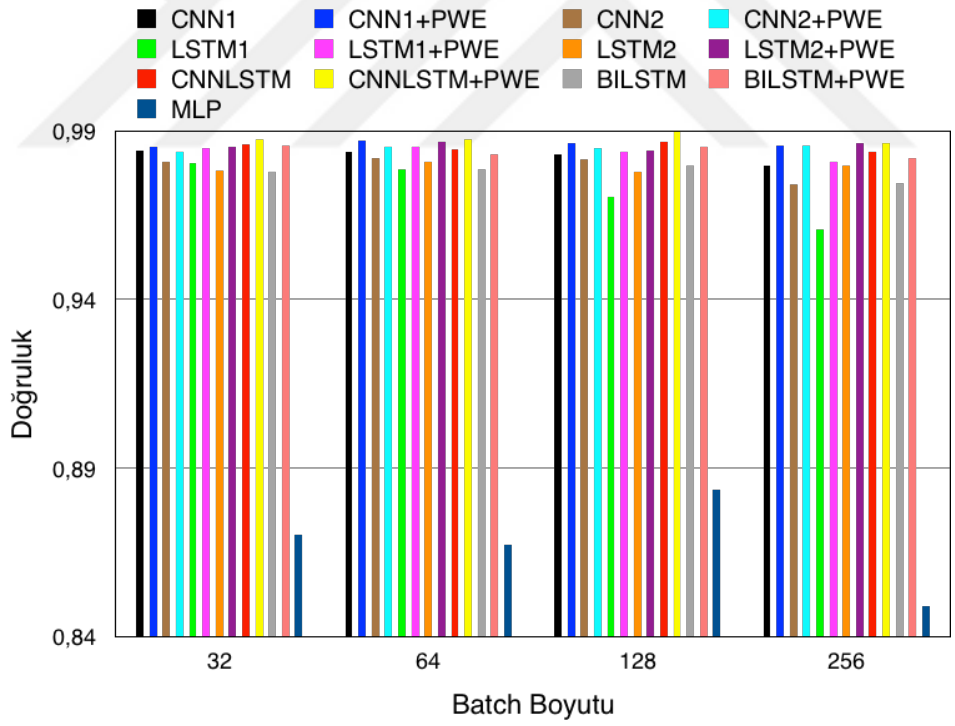
Şekil 6.14'te verilen deneysel sonuçlardan gözlemlenebilir ki PWE ile başlatılan modeller PWE kullanmadan eğitilen modellere göre daha iyi performans sergilemiştir. Ayrıca yapılan testlerden görülmüştür ki özellikle bir ya da iki katmanlı CNN modeller PWE kullanılmadığı durumda veriyi aşırı öğrenmiştir ve bu sorun PWE kullanımı ile çözülmüştür. Burada denilebilir ki CNN modeller için önceden eğitilmiş kelime temsillerin kullanılması önemli bir ölçüttür. Bu çalışmada veri setinin büyüklüğüne bağlı olarak en iyi değerlerin değiştiği farklı büyüklüklerdeki batch boyutları için doğruluk ve hata değerleri hesaplanmıştır. Özellikle CNN modeller olmak üzere tüm modellerde küçük batch boyutu aşırı öğrenmeye sebep olmuştur. Bu nedenle kullanılan veri seti üzerinde model eğitimi için, sunulan modeller PWE ile başlatılmayacaksa, 128 ve 256 batch boyutlarının kullanılması uygun olacaktır.

PWE ile derin öğrenme modelleri için PWE kullanmadan eğitilen modeller üzerindeki performans iyileştirme sonuçları Tablo 6.1'de ayrıntılı olarak açıklanmıştır. Tablo 6.1'den görüleceği gibi PWE tüm modeller için doğruluk oranlarında artış sağlamıştır. Bununla birlikte en önemli performans kazanımı %1.654 ve 0.050 ile sırasıyla CNN1 ve CNN-LSTM modellerinde olmuştur. Ayrıca bu modeller eğitim ve test sürelerine bakıldığında çalışma süresi bakımından avantaj sağlamaktadır. Buna karşılık iki katmanlı LSTM2 ve iki yönlü BiLSTM

modellerinin toplam hesaplama zamanı ve maliyeti diğer modeller ile kıyaslandığında çok yüksektir. Sonuç olarak düşük zaman maliyeti ve yüksek performans kazancı esas alındığında CNN1 ve CNN-LSTM modelleri bu tez çalışma kapsamında oluşturulan önceden eğitilmiş kelime temsilleri ile birlikte kullanıldığında iki duygu sınıflandırma çalışması için diğer modellere göre başarılı doğruluk sonuçları vermektedir.

Tablo 6. 1. PWE ile başlatılan derin modellerin performans kazanımları

Modeller	Şekil 6.14.	PWE ile Maksimum Performans Kazanımı (%)	Batch Boyutu	Eğitim Süresi (sn.)
CNN1	a	**1.654**	128	310
CNN2	b	0.011	32	520
LSTM1	c	0.020	32	955
LSTM2	d	0.005	256	6210
CNN-LSTM	e	**0.050**	32	481
BiLSTM	f	0.014	64	7514



Şekil 6. 15. Tüm öğrenme modelleri için duygu sınıflandırma sonucu performans karşılaştırılması

Tüm modeller için ikili duygu sınıflandırması sonucu alınan doğruluk oranları çeşitli batch boyutlarına göre Şekil 6.15'te sunulmuştur. Bu grafiklerden aşağıdaki sonuçlar çıkarabilmektedir:

- Tek katmanlı CNN1+PWE, iki katmanlı CNN2 ve tek katmanlı CNN1 modellerine göre daha iyi performans sağlamıştır.
- Tek katmanlı LSTM1+PWE, iki katmanlı LSTM2, tek katmanlı LSTM1 ve BiLSTM modellerine göre daha başarılıdır.
- PWE ile tek katmanlı ağlar için CNN1, LSTM1 modele göre daha başarılı performans sergilerken PWE ile iki katmanlı ağlar için, LSTM2, CNN2 modele göre daha iyi sonuçlar vermiştir.
- CNN-LSTM+PWE modeli diğer tüm modellerden daha başarılı performans sonucu vermiştir.
- MLP modeli diğer tüm derin öğrenme modelleriyle kıyaslandığında kötü sınıflandırma sonucu vermiştir.
- CNN1, CNN2 modelden daha iyi sonuç vermiştir. Dolayısıyla CNN modellerde duygu sınıflandırması için fazla katman kullanarak fazla parametre ve hesaplama karmaşıklığı oluşturmaya ihtiyaç kalmamıştır.

Tablo 6.2'de veri setinden örnek yorumlar ve bu yorumlar üzerinde test edilen CNN-LSTM + PWE modelin tahmin çıktıları verilmiştir. Modelin tahmin ettiği değer oranı [0-1] arasında ölçeklenmiştir. Eğer tahmin edilen değer oranı 0'a yakınsa tahmin etiketi 0 alınır tam tersi durumda 1 alınır. Gerçek etiket ile tahmini etiket aynı ise yorumun duygusunun doğru sınıflandırıldığı sonucu çıkarılmaktadır. Buna göre tabloda verilen rastgele örnek yorumların gerçek ve tahmin etiketinin örtüştüğü ve son yorum hariç doğru sınıflandırıldığı görülmektedir.

Bu tez çalışması kapsamında ayrıca model eğitim ve test verilerine dahil edilmeyen 4000 film yorumu için eğitilmiş modellerin testi gerçekleştirilmiştir. Bu test seti 2000 pozitif ve 2000 negatif yorum içermektedir. Her bir model için test doğruluğu Tablo 6.3'te verilmiştir. Tablodan görüleceği üzere eğitilmiş model harici test örnekleri üzerinde de yüksek ve başarılı sonuçlar

elde etmiştir. CNN-LSTM ve CNN1 modelleri performansı ile bu test sonuçlarında da ön plana çıkmıştır.

Tablo 6. 2. Rastgele seçilmiş bazı test örnekler üzerinde model çıktılarının açıklanması (T0:Tahmin oranı % sonucu, TE: Tahmini etiket, GE: Gerçek Etiket)

TO	TE	GE	Yorum
0.787164	1	1	film gercekten tam bi.. superdi ya..
0.000068	0	0	berbat tamamen zaman kaybi
0.820993	1	1	muthis bir film bayildim hic kovalamaca dovus eglence kahkaha . bu filmin serisi bitmesin . gercekten harika .
0.818120	1	1	muhtesem senaryo muhtesem oyunculuk muhtesem yonetmen . . kill bill i izlemeyen cok sey kaybetmis…
0.000029	0	0	asiri dandik bi film arkadaslar vakit kaybi olur sizin icin hele aile sakin ha sakin izlemeyin . tuhaf bi film ya .
0.426062	0	1	kotu amerikan. yaptigi icin oscar alacak demistim .. izleyince anladim tam tersine malesef oscari kaybetmis . demekki oscar guzel muhtesem .
0.000132	0	0	Hayal kirikligi . izleyin gorun .

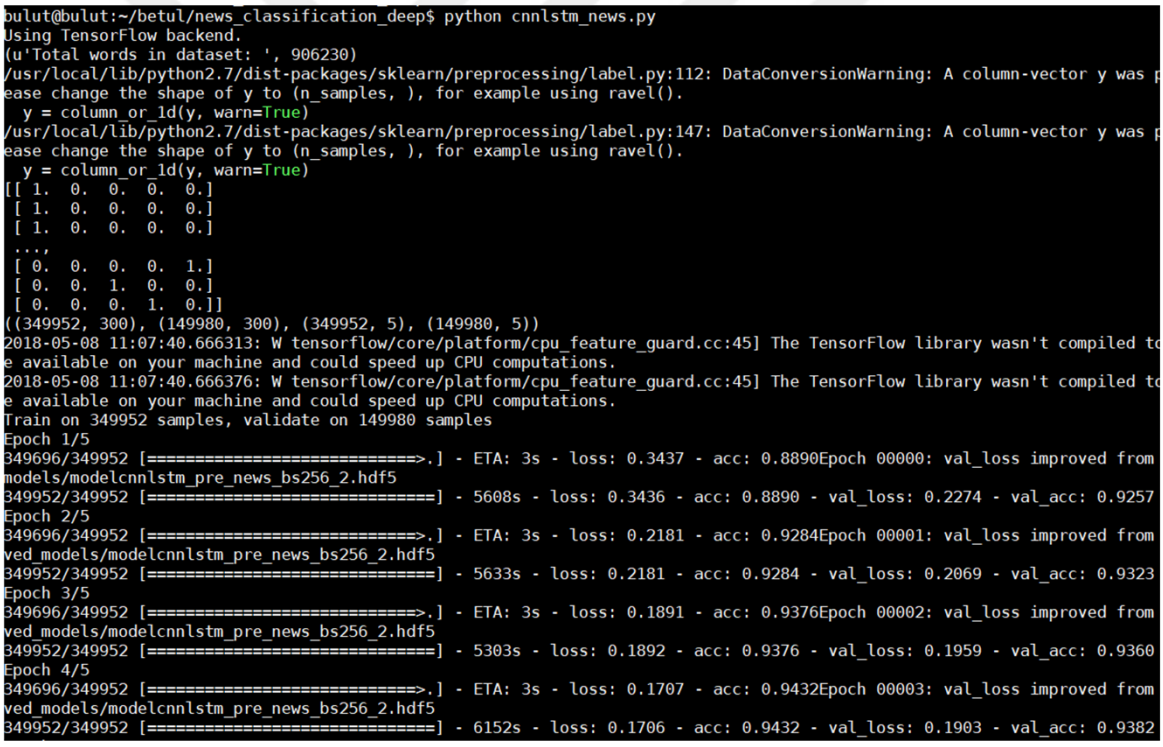
Tablo 6. 3. 4000 test örneği üzerinde sınıflandırma sonuçları

Model (PWE)	Yanlış Sınıflandırılan Yorum Sayısı	Test Sonucu (%)
MLP	869	78,27
CNN1	96	97,60
BILSTM	129	96,77
LSTM1	137	96,57
CNN2	95	97,62
CNNLSTM	**77**	**98,07**
LSTM2	139	96,52

6.8. Haber Sınıflandırması Performans Sonuçları ve Tartışma

Bu bölümünde duygu sınıflandırması çalışmasında başarılı performans gösteren CNN-LSTM modeli, önerilen en iyi parametre seçimleri ile birlikte kullanılarak haber veri seti

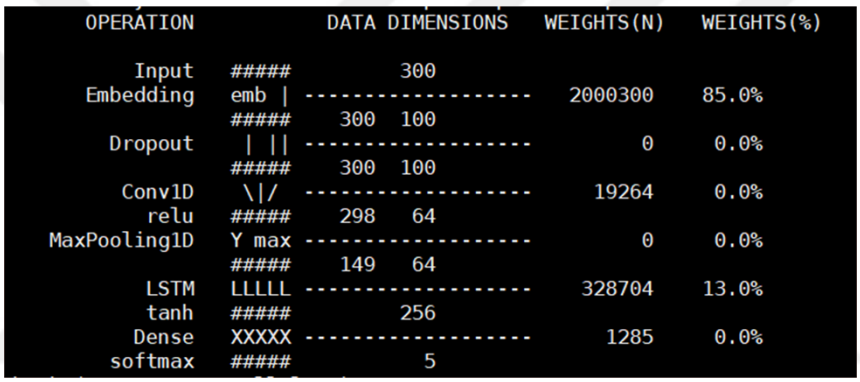
üzerinde de eğitilip test edilmiştir. Buradaki amaç, bir önceki bölümde tartışılan en iyi modellerden birisinin ayrı bir veri seti ve farklı sınıflandırma çalışmaları üzerinde de test edilerek çalışma performansı sonucunun güvenirliliği ve doğruluğu karşılaştırılmak istenmiştir. Haber sınıflandırma bu görevde kullanıldığı haliyle konu sınıflandırma, verilen bir dokümanın tanımlanmış konu başlıklarına göre kategoriye ayrılma işlemidir. Tez çalışması kapsamında çeşitli haber sitelerinden Java tabanlı web kazıma aracı ile toplanan 500.000 etiketli haber verisi üzerinde doküman sınıflandırma işlemi gerçekleştirilmiştir. Türkçe haber veri seti ekonomi, politika, kültür, dünya ve spor olmak üzere 5 ana başlık ile etiketlenmiş bu haberlerin her biri yaklaşık 100.000 veri içeri içermektedir. Yapılan eğitim ve testler aynı Intel Xeon X5570 sunucuda gerçekleştirilip olup çalışma ortamının model eğitilirken alınan ekran görüntüsü Şekil 6.16'da verilmiştir.



Şekil 6. 16. Model geliştirme ortamından alınan ekran görüntüsü

Teknoloji, yaşam, magazin gibi diğer haberleri de içeren yaklaşık 1 milyon haber veri seti üzerinde de modelin temsil katmanında kullanmak için önceden eğitilmiş kelime temsilleri elde edilmiştir. Burada da PWE ile adlandırılacak olan kelime temsillerinin etkisini model üstünde görebilmek için PWE kullanan ve kullanmayan olmak üzere iki çeşit CNN-LSTM model eğitilip test edilmiştir. Şekil 6.17'de modelin yapısı sunulmuştur, görselleştirmek amacıyla Piotr Migdal [242] tarafından sunulan bir keras kütüphanesi kullanılmıştır. Kütüphane kullanımı için kod parçası aşağıda verilmiştir:

```
from keras_sequential_ascii import sequential_model_to_ascii_printout
sequential_model_to_ascii_printout(model)
```



Şekil 6. 17. CNN-LSTM model yapısı

Şekil 6.17'deki görsel incelendiğinde, CNN-LSTM+PWE modelin girdi ve temsil katmanı (embedding), konvolüsyon katmanı (Conv1D), havuzlama katmanı (MaxPooling1D), LSTM katmanı ve çıktı katmanından (Dense) oluştuğu görülmektedir. Çıktı katmanında ekonomi, politika, kültür, dünya ve spor haber etiketine karşılık gelen 5 nöron birimi bulunmaktadır. Her bir sınıfın ilgili olduğu nöron [0-1] arasında bir değer alarak tahmin edilebilmesi için eğitilecektir. Tahmin çıktıların toplamını 1'e çeken Softmax aktivasyon fonksiyonu ile bu 5 etiketli veri sınıflandırılmaktadır.

Tablo 6. 4. CNN-LSTM modelinin hiper parametreleri

Parametre	**Değeri**
Sozluk_boyutu	30.000
Temsil_boyutu	100
Max_cumle_uzunlugu	300
Filtre_sayisi	64
Filtre_boyutu	3x3
Gizli_katman_boyutu	256
Batch_boyutu	256

Tablo 6.4'te verilen parametre değerleri ile birlikte oluşturulan model yapısının kod parçası aşağıda verilmiştir:

```
model = Sequential()
model.add(Embedding(sozluk_boyutu,
                    temsil_boyutu,
                    weights=[temsil_agirligi],
                    input_length=max_cumle_uzunlugu))
model.add(Dropout(0.3))
model.add(Conv1D(filters=filtre_sayisi,
                 kernel_size=filtre_uzunlugu))
model.add(Activation("relu"))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(gizli_katman_boyutu,
               dropout=0.2,
               recurrent_dropout=0.2))
model.add(Dense(5))
model.add(Activation("softmax"))
```

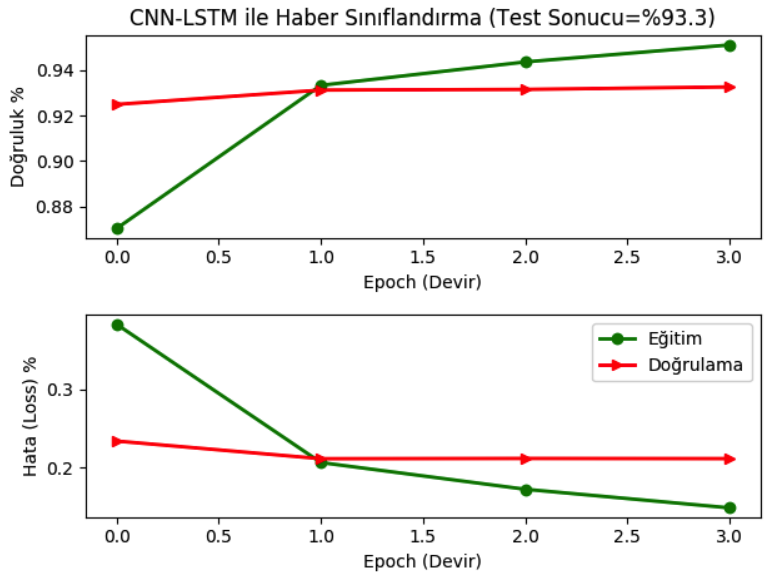
Kelimeler için öncelikle sıralı bir model oluşturulur. Bu modelin girdisi sözlükteki kelimelerin kelime_id karşılıklarıdır. PWE ile başlatılmayan CNN-LSTM modelde temsil ağırlıklarına başlangıç olarak rastgele ve küçük değerler atanır. Eğitim süresince model geriye yayılım kullanarak bu ağırlıkları güncellemektedir. Buna karşılık PWE bu ağırlık değerleri önceden eğitilmiş kelime temsillerinin bulunduğu *haber_word2vec.vec* dosyasından alınmaktadır. Bu dosyada [kelime, vektör] şeklinde kelimelerin sayısal vektör değerleri tutulmaktadır. Eğitilen bu dosyadan *temsil_agirliklari* adı verilen bir ağırlık matrisi oluşturulur. Temsil katmanına bu ağırlık matrisi ile ağırlıklar yüklenir. Derin sinir ağı bu ağırlıkları geriye yayılım kullanarak günceller ve modelin iyi bir başlangıç ağırlık değerleri olduğu için daha hızlı ve başarılı performans sonucu alınır:

```
word2vec=KeyedVectors.load_word2vec_format("haber_word2vec.vec",
                                                    binary=False)
temsil_agirliklari = np.zeros((sozluk_boyutu, temsil_boyutu))
for kelime, index in word2index.items():
    try:
        temsil_agirliklari[index, :] = word2vec[kelime]
    except KeyError:
        pass
```

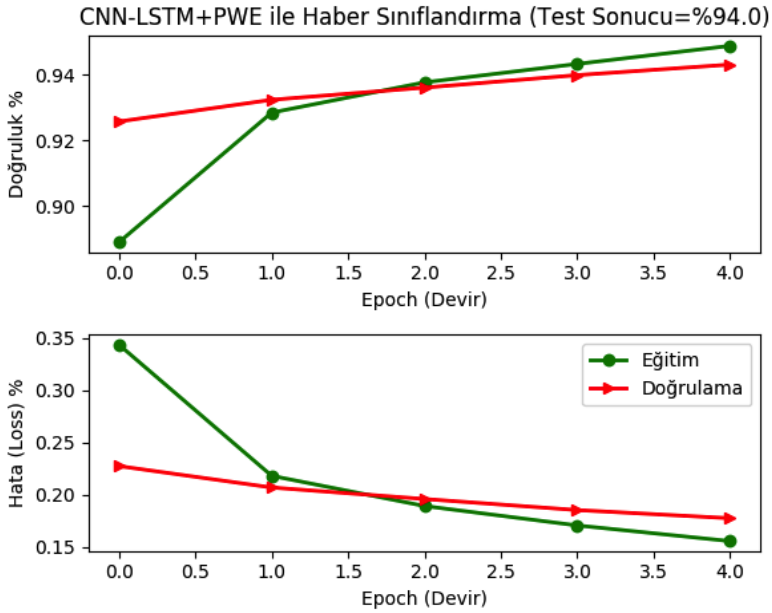
Modelin eğitimine sunulmadan önce veri seti üzerinde bir takım işlemler gerçekleştirilmektedir. Öncelikle açık kaynak bir Python Kütüphanesi olan Pandas ile DataFrame olarak adlandırılan veri yapısı kullanılarak veri yükleme işlemi gerçekleştirilir. Her bir haber burada cümle olarak geçecektir. Her bir cümle maksimum cümle uzunluğuna göre ya kesilir ya da cümleye sıfır doldurma işlemi uygulanır. Etiketler ise kategorik formata çevrilir. Yapılan bu işlemler sonucu X olarak adlandırılan ilk 5 cümlenin ve bu cümlelerin y ile adlandırılan etiketleri aşağıdaki sonuçlarda görüldüğü gibi çıktı vermiştir. 3.haberdeki kelimelerin sayısı tanımlanan maksimum cümle uzunluğundan fazla olduğu için kesilmiştir. Diğer haberler ise kısa olduğundan cümlelerin başına 0 eklenerek sıfır doldurma işlemi gerçekleştirilmiştir. Etiketler ise konularına göre tam sayı olarak kodlanmıştır:

```
print(X[:5])
   [[0 0 0 ..., 290 29 23]
    [0 0 0 ..., 56 254 152]
    [225 1133 13585 ..., 3492 1521 23]
    [0 0 0 ..., 62 7817 3762]
    [0 0 0 ..., 3196 1 1758]]
print(y[:5])
   [4. 1. 4. 3. 5.]
```

Modellerin eğitim ve doğrulama aşamaları için 349,696 haber verisi, test aşaması için 149,980 haber verisi kullanılmıştır. Yani veri setinin %70 bölümü eğitim-doğrulama ve %30 bölümü test setine ayrılmıştır. Veri setinde toplam 936.230 benzersiz kelime mevcuttur. CNN-LSTM ve CNN-LSTM+PWE modellerinin sınıflandırma sonuçları sırasıyla Şekil 6.18 ve 6.19'da verilmiştir.



Şekil 6. 18. CNN-LSTM modelinin haber sınıflandırma sonuçları



Şekil 6. 19. CNN-LSTM+PWE modelinin haber sınıflandırma sonuçları

Modellerin haber sınıflandırmadaki performans sonuçlarına bakıldığında yüksek doğruluk ve düşük hata oranları ile yüksek başarı elde ettiği görülmektedir. Ayrıca duygu sınıflandırmada olduğu gibi PWE kullanımı modelin veriyi daha iyi öğrenmesini ve test verisi üzerinde daha yüksek %0,7 performans kazanımı ile daha iyi sınıflandırma sonucu verdiği görülmektedir. Tablo 6.5'te modellerin test üzerindeki performans karşılaştırmaları ve Tablo 6.6'da rastgele seçilmiş test örneklerinin çıktıları sunulmuştur.

Tablo 6. 5. Modellerin performans karşılaştırılması

Model	**Test Puanı %**	**Eğitim Süresi (sn)**
CNN+LSTM	0.932744365914	21304.30
CNN+LSTM+PWE	0.94002533671	29564.71

Tablo 6. 6. Rastgele seçilmiş bazı test örnekleri üzerinde haber sınıflandırması için model çıktısı (1:Politika, 2:Spor, 3:Ekonomi, 4:Kültür, 5:Dünya)

Tahmin Oranı (%)	**Tahmin Etiketi**	**Gerçek Etiket**	**Haber**
77.423328	4	4	uluslararasi antalya piyano festivali ispanyol ruzgariyla basladi festivalinde menderes turel surprizi antalya buyuksehir belediyesinin duzenledigi uluslararasi antalya piyano…
84.681666	2	2	real madrid evinde rahat kazandi la liganin alt siralarinda bulunan rakibine karsi ustun bir oyun sergilemeyen real madrid buna ragmen rahat bir galibiyet aldi…
56.764811	1	5	erdogan aliyev gorusmesi oncesi resmi toren cumhurbaskani recep tayyip erdogan azerbaycan cumhurbaskani ilham aliyev ile gerceklestirecegi gorusme icin ziyaret ettigi azerbaycan cumhurbaskanligi sarayinda…
86.990446	1	1	guvenlik ve istihbarat komisyonu toplandi guvenlik ve istihbarat komisyonu komisyon baskani emrullah isler baskanliginda tbmmde topladi toplantida devletin istihbarat hizmetleri ile…
94.034940	3	3	borsa gunu dususle tamamladi borsa gunu yuzde dususle tamamladi borsa istanbul bist endeksi gunu yuzde dususle tamamladi endeks puanlik azalisla bin puandan kapanirken toplam islem hacmi…

Eğitim ve doğrulama sonuçlarını iki boyutta görselleştirmek için Matplotlib tercih edilmiştir. Model eğitiminde aşırı öğrenmeyi engellemek için ayrıca bazı kontrol noktaları (check points) tanımlanmıştır. İlki doğrulama hatasındaki performansı izleyerek doğrulama hatası arttığında yada bir önceki hataya göre gelişim göstermediğinde eğitimi durduran dolayısıyla epoch sayısını da belirleyen erkenden durdurma kontrolüdür. Bu kontrol Geoff Hinton tarafından ücretsiz ve güzel bir öğlen yemeği olarak ifade edilmiştir. İkincisi de en iyi doğrulama sonuçlarını her bir epoch zamanında modelin bir kopyası olarak kaydeden model kontrolüdür. Keras kütüphanesinde bu kontrol noktaları aşağıdaki kod parçaları ile tanımlanmıştır:

```
Erken_durdurma=EarlyStopping(monitor='val_loss',patience=2,verbose=0)
Model_kaydetme=ModelCheckpoint(filepath="cnnlstm_model.hdf5",
                monitor='val_loss',save_best_only =True, verbose=1)
```

6.9. Sonuç

Bu çalışmanın ilk bölümünde BEYAZPERDE film yorumları veri seti üzerinde çeşitli mimarilerde derin öğrenme modelleri oluşturularak ikili duygu sınıflandırması gerçekleştirilmiştir. Tüm veri seti 44,617 pozitif ve negatif yorumlardan oluşmaktadır. Bu veri setindeki 4000 örnek modellerin testleri için ayrılmıştır. Modeller Adam optimizer kullanılarak eğitilmiştir ve 32, 64, 128 ve 256 batch boyut seçimine göre karşılaştırılmıştır. Erken durdurma tekniği kullanarak eğitim örneklerinin sayısı yani epoch sayısı otomatik optimize edilmiştir. Duygu sınıflandırmasının başarısı CNN, LSTM, BiLSTM ve CNN-LSTM birleşik modelleri üzerinde karşılaştırılmıştır. Ayrıca daha büyük bir film yorumları veri seti üzerinde skip-gram word2vec algoritması uygulayarak 100 boyutlu önceden eğitilmiş kelime temsilleri (PWE) elde edilmiştir. Derin öğrenme üzerinde bu kelime temsillerinin etkisi tartışılmıştır.

Çalışmanın ikinci bölümünde HABERLER veri seti üzerinde duygu sınıflandırma için kullanılan derin modellerin performanslarına bakılarak CNN-LSTM modeli ile 5 kategoride haber sınıflandırması başarıyla gerçekleştirilmiştir. Yaklaşık 10 milyon haber verisi üzerinde skip-gram word2vec algoritması uygulanarak elde edilen kelime vektörlerinin kullanılan model

üzerindeki performans kazanımı tartışılmıştır. Bu tez çalışmasının sonuçları ve katkıları aşağıda maddeler halinde verilmiştir:

1. Metin sınıflandırmada derin öğrenme modelleri uygulayarak, geleneksel sinir ağlarının aşağıdaki kısıtlamaları ve problemleri çözülmüştür:
    - *Sabit metin girdisi boyutu* = Bir sinir ağı mimarisi sabit boyutlu girdi almaktadır. Metin sınıflandırma gerçekleştirilirken değişen uzunlukta cümleler ve kelimeler düşünüldüğünde değişken sayıda girdi oluşmaktadır. Geleneksel bir sinir ağı ile bu girdiler modelleyemez.
    - *Hafıza/bellek problemi* = Sinir ağının bir başka kısıtlaması bellek eksikliğidir. Örneğin duygu sınıflandırmasında tüm cümlenin duygusunu çıkarmak için kelimelerin diziliş sırasını hatırlamak önemlidir. Buna karşılık geleneksel bir sinir ağındaki her bir giriş nöronu birbirinden bağımsız çalışmaktadır. Dolayısıyla cümledeki bir önceki kelime ile sonraki kelime arasında bir ilişki kuralamamaktadır.
2. Oluşturulan önceden eğitilmiş kelime vektörleri ile derin öğrenme modellerinin performansı arttırılmıştır.
3. Yaklaşık 1 milyon benzersiz kelime içeren çok büyük haber verileri toplanmıştır ve bu veri seti üzerinde Türkçe kelime vektörleri eğitilmiştir.
4. Farklı mimari ve katmanlarda derin öğrenme modelleri oluşturularak ve bu modellerin performansını sunarak Türkçe duygu sınıflandırma çalışmalarına destek sağlanmıştır.
5. Kullanılan derin öğrenme modelleri ile hiper parametre optimizasyonundan sonra aşırı öğrenme probleminin üstesinden gelinerek duygu sınıflandırmada %98 doğruluk oranlarında çok başarılı sonuçlar alınmıştır.
6. En yüksek doğruluk oranına ulaşan CNN-LSTM modeli sadece çıktı katmanı değiştirilerek yani 5 sınıfa göre eğitilerek 150 bin haber testi üzerinde %94 oran ile yüksek bir sınıflandırma başarısı elde edilmiştir. Türkçe çoklu metin sınıflandırma çalışmalarına böylelikle katkı sağlanmıştır.

7. Bu çalışma kapsamında haber verilerinden eğitilen kelime vektörleri ile model eğitildiğinde hem %0,7 performans kazanımı sağlanmış hem de modelin veriyi aşırı öğrenmesi problemi çözülmüştür.
8. Türkçe metin sınıflandırma çalışmaları için derin öğrenme modelleri ve önemli parametre seçimleri önerilmiştir. Buna göre CNN1 ve CNN-LSTM modelleri yüksek performansı ve düşük zaman maliyeti ile diğer modellerden ön plana çıkmıştır. Önceden büyük bir veri seti üzerinde eğitilmiş Türkçe kelime temsilleri ile birlikte derin öğrenme modellerinin kullanması hem çalışma süresini azaltmış hem de performans kazanımı sağlamıştır.
9. Burada MLP model olarak adlandırılan çok katmanlı algılayıcılar, diğer tüm derin öğrenme modellerine göre duygu sınıflandırmada çok başarısız kalmıştır. Bu durum derin sinir ağlarının yapay zeka alanında ümit vadeden bir gelişme olduğunu göstermektedir. Dolayısıyla doğal işleme alanında bu tez çalışmasının da yoğunlaşmış olduğu derin öğrenme mimarilerinin tercih edilmesi doğru karar olacaktır.

7. ORTALAMA DOKÜMAN VEKTÖRLERİ İLE DUYGU ANALİZİ

Duygu analizi çalışmalarında bilinen en yaygın problem, metinlerin özellik vektörleri öğrenilirken kelime sırasının kaybedilmesi ve kelime anlamlarının göz ardı edilmesidir. Cümleler, paragraflar ve dokümanlar gibi farklı uzunluk ve içerikteki metinler üzerinde başarılı sonuçlar elde eden Paragraf Vektörü, bu problemin çözümü için etkili bir gözetimsiz algoritmadır. Bu tez çalışmasında doc2vec olarak da adlandırılan Paragraf Vektör kullanılarak, metin sınıflandırması çalışmaları için yeni bir Ortalama Doküman Vektör yöntemi önerilmiştir. Türkçe duygu sınıflandırması çalışmalarında doğru ve güvenilir bir karşılaştırma yapmak için BEYAZPERDE film yorumları veri seti ve duygu etiketi sistemi sunulmuştur. Ayrıca gerçekleştirilen Türkçe duygu sınıflandırması çalışması için hiper parametre ayarlamaları üzerine öneriler sunulmuştur. Duygu sınıflandırması üzerine yapılan deneysel çalışmalar göstermiştir ki, önerilen yaklaşım en doğru model hiper parametresi seçiminde %88.45 doğruluk oranı ile başarılı bir performans sergilemiştir.

7.1. Amaç ve Kapsam

Son beş yıl kadar, güçlü hesaplama kaynakları olmaksızın büyük metin verilerinden konu ve duyguları ortaya çıkarmak mümkün değildi. Mühendislerin, kelimeleri ve belgeleri büyük ölçüde anlamaya yetecek etkili yöntemleri yoktu. Günümüzde ise derin sinir ağları kullanarak, yapılandırılmamış metinlerden çeşitli öğrenme modelleri eğitilerek anlamlı bilgi çıkarılabilmektedir. Büyük metin verilerini analiz etmek ve sınıflandırmak için, son zamanlarda araştırmacılar duygu analizi üzerine yoğunlaşmaktadır. Görüş madenciliği olarak da bilinen duygu analizi, tweet'ler, mesajlar, yorumlar, eleştiriler ve benzeri kısa metinlerin genel görüşünü bulmayı hedefleyen NLP uygulamalarından birisidir.

Duygu sınıflandırması alanında yapılan çalışmaların çoğu İngilizce dili için geliştirilmiş ve test edilmiştir. Özellikle Türkçe için, bu alanda katkı sunulan çalışma sayısı oldukça azdır. Bu tez çalışmasında, doc2vec tabanlı geliştirilen Ortalama Doküman Vektörleri (Average Document Embedding-ADE) yaklaşımı kullanılarak Türkçe ve İngilizce film yorumlarında

duygu sınıflandırması gerçekleştirilmiştir. Gerçekleştirilen bu çalışma ile şu soruların cevaplarını bularak metin sınıflandırması çalışmalarına ışık tutmak amaçlanmıştır: (1) Paragraf vektör modelleri Türkçe metinlerin gösteriminde etkili mi? (2) Seçilen dbow, dbow_skip, dm_mean ve dm_concat doc2vec modellerinden hangisi daha başarılı? (3) Hiper parametre ayarları ile bu modellerin başarısını arttırmak mümkün mü? (4) Ortalama doküman vektörü yaklaşımının İngilizce ve Türkçe metinleri sınıflandırmasındaki başarısı nedir? Sonuç olarak bu çalışmada, Paragraf vektör modellerinin Türkçe metinlerin sayısal vektör temsilinde başarılı olduğu görülmüştür. Dağıtık kelime torbaları modellerinin (dbow ve dbow_skip), dağıtık bellek modellerine (dm_mean ve dm_concat) göre hem Türkçe hem de İngilizce metinlerde daha iyi sonuçlar verdiği ispatlanmıştır. ADE yaklaşımının metinleri sınıflandırmada seçilen dikkatli parametre ayarlamaları ve vektör modelleri ile hızlı ve etkili bir performans sağlandığı gözlemlenmiştir. Ayrıca Türkçe duygu sınıflandırması alanında çalışan araştırmacıların ortak kullanılacağı BEYAZPERDE film yorumları veri seti ve bu veri setinin geliştirilmesi için Duygu Etiketleme Sistemi sunulmuştur. Sunulan veri seti üzerinde önerilen yaklaşımlar test edilerek, Türkçe duygu sınıflandırması çalışmasına katkı sağlanması hedeflenmiştir.

7.2. İlgili Çalışmalar

Dağıtık kelime temsillerinin yüksek boyutlu vektörlerle gösterimi, kelimelerin benzer guruplara ayrılmasını sağlayarak NLP alanındaki problemlerin çözümüne büyük bir katkı sağlamıştır [170, 228, 159, 162, 243-245]. Mikolov ve arkadaşları [228] word2vec modelini büyük veri kümelerinden kelime vektörlerini öğrenme amaçlı tasarlamışlardır. Her bir kelime için temsil vektörü oluşturarak kelimeler arasındaki farklı dizilimleri ve anlamsal ilişkileri tespit etmeye çalışmışlardır. Word2vec modelinde kelime vektörlerini oluşturmak için Skip-gram ve Devamlı Kelime Torbaları (Continous Bag of Words -CBOW) algoritmaları tanımlanmıştır. Hiyerarşik Softmax ve Negatif Örnekleme ise eğitim metodu olarak kullanılmıştır. CBOW modelde kelime tahmini, hedef kelimeyi çevreleyen kelimeler üzerinden yapılırken, Skip-gram modelde komşu kelime (bağlam) tahmini, verilen merkezi kelime ile yapılmaktadır. CBOW modelde kelimeyi çevreleyen kelime toplulukları ya da kelime bağlamları vektörel olarak

toplanılarak tek bir kelime vektörü elde edilmesi hedeflenmektedir. Daha sonraki çalışmalarda kelime modelleri genişletilerek cümle, paragraf bu çalışmada kullanıldığı haliyle doküman temsilleri elde edilmiştir [246-249].

Le ve Mikolov [159], cümle ve dokümanların vektörlerle temsili için, word2vec modelini genişleterek paragraf vektörleri olarak da bilinen doc2vec gözetimsiz modelini geliştirmişlerdir. Ayrıca paragraf vektörleri yardımıyla film yorumlarının duygu analizini yapmışlardır. Paragraf vektörlerinde doküman temsili dbow ve dm olarak iki guruba ayrılmıştır. Dbow modeli skip-gram modeline benzer şekilde çalışmakla birlikte, skip-gram'da giriş vektörü tek kelimeden oluşmasına karşılık, dbow modelinde giriş vektörü verilen dokümanın tamamını içeren bir temsil vektörüdür. Kelime sıralamasını göz ardı eden dbow modeli dmpv modeline göre daha basit bir yapıya sahiptir. Paragraf vektör modelinin tanıtıldığı çalışmada [159], dm (pv-dm) modelinin dbow (pv-dbow) modelinden daha iyi performans sergilediğini iddia etmesine rağmen aksi yönde Lau ve Baldwin [243] dbow modelin dm modelden daha iyi olduğunu göstermişlerdir. Bu tez çalışmasında da yapılan testlerde dbow modellerin, dm modellerine göre daha iyi performans sergilediğini gözlemlenmiştir.

Paragraf vektör dm modelinde bağlam kelime tahmini, kelime vektörleri ile birleştirilmiş doküman vektörleri üzerinde yapılmaktadır. [243]'deki çalışmada önceden eğitilmiş kelime temsilleri (pre-trained word embeddings) kullanılarak paragraf vektörlerinin geliştirilebileceği gösterilmiştir. Dai ve arkadaşları [244] paragraf vektörlerini kullanarak Wikipedia ve arXiv makalelerine semantik benzerliği en yakın olan makeleleri bulmaya çalışmışlardır. Skip-gram modelinde olduğu gibi kelime temsilleri ile paragraf vektörleri birlikte eğitilerek paragraf vektörleri daha nitelikli hale dönüştürülmüştür. Belgenin anlamsal benzerliğini bulmak için yapmış oldukları kıyaslamada, paragraf vektörleri ile Ortalama Kelime Temsilleri (Average Word Embeddings) Wikipedia makalelerinde, Saklı Dirichlet Ataması (Latent Dirichlet Allocation-LDA) yöntemine göre daha iyi performans sergilediğini göstermişlerdir. Bu çalışmada da paragraf vektörleri kullanılarak Ortalama Doküman Temsilleri yöntemi sunulmuştur ve Ortalama Kelime Temsilleri yöntemine göre dikkatli parametre seçimi yapıldığında daha iyi performans sağladığı görülmüştür.

7.3. Kelimelerin Dağıtık Gösterimleri

Vektör uzayındaki kelimelerin dağıtık gösterimleri, NLP çalışmalarında kullanılan öğrenme algoritmalarının performanslarını önemli ölçüde arttırmaktadır. Dağıtık gösterimler ya da bugünkü kullanımıyla kelime vektörleri ya da temsilleri (word embeddings), CBOW ve Skip-Gram modellerinden biri ile derin bir sinir ağı kullanılarak öğrenilmektedir. Bu gösterimlerin genel özelliği benzer içerikli kelimeleri gruplamasıdır. Dağıtık kelime gösterimlerinin ilk kullanımı aynı zamanda sinir ağları için geriye yayılım algoritmalarının kullanımının yaygınlaşmasına katkıda bulunan çalışma [170] ile 1986 yılına uzanmaktadır. Ancak NLP çalışmasındaki kullanımı 2003 yılında Bengio [245] ile başlamıştır. Önerdikleri olasılıksal sinir dil modeli, birkaç önceki kelimeler (bağlam) verilerek bir sonraki kelimenin (hedef) koşullu olasılığı tahmin etme yoluyla $w_1, w_2, w_3 \dots w_t$ kelimelerin tüm dizilimlerinin ortak olasılığını tahmin etmeyi hedeflemektedir:

$$P(w_1, w_2 \dots, w_t) = \prod_t P(w_t | w_{t-1}, \dots, w_{t-n+1}) \tag{7.1}$$

Burada w_t bir metinin t konumundaki kelime, V sözlük olmak üzere koşullu olasılık, $\sum_v f(v, w_{t-1}, \dots, w_{t-n+1}) = 1$ ile normalize edilmiş $f(w_{t-1}, \dots, w_{t-n+1})$ fonksiyonu kullanılarak hesaplanmaktadır [245]. Sinir ağı daha sonra bir Softmax çıktı katmanı ile bu pozitif olasılıkların toplamını 1'e eşitlemektedir. Eğitimi hızlandırmak için kullanılan bu hiyerarşik dil modeli word2vec [228] çalışmasında geliştirilerek, Softmax hiyerarşik model tabanlı ikili bir Huffman ağacı ile sözlüğün gösterimi elde edilmiştir. Daha sonra bu tez çalışmasında ayrıntılı incelenen doc2vec ya da paragraf vektörleri modelinde [159] de bu hiyerarşik yapı kullanılmıştır.

Word2vec, temsil katmanını kullanarak kelimeler hakkında semantik bilgi içeren vektör gösterimlerini bulan bir algoritmadır. Temsiller, farklı sınıflardaki kelimeleri ve dokümanları daha etkili bir şekilde temsil etmek için kullanılan derin sinir ağ yöntemidir. Temsil katmanı başlangıç için rastgele ağırlıkları olan basit bir tam bağlı katmandır. Bu katman, eğitim setindeki tüm kelimeler için bir temsil öğrenmektedir. Benzer içerikli kelimelerin birbirine yakın vektör

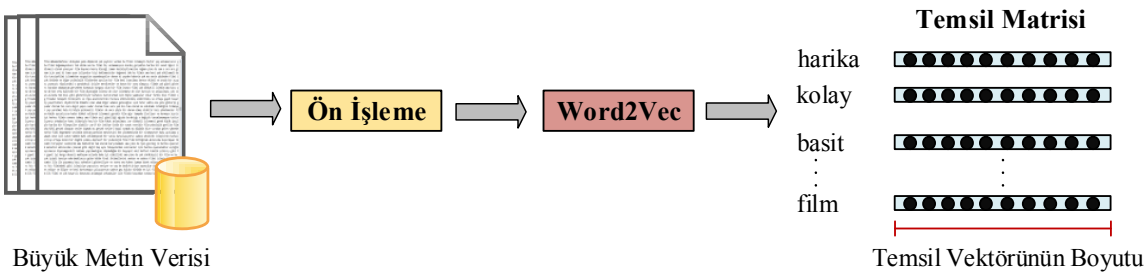
gösterimleri olduğu bu algoritmada, iki mimari sunulmaktadır: Skip-gram ve CBOW. Skip-gram mimarisi CBOW mimarisine göre daha iyi çalışmaktadır. Skip-gram mimarisinde, kelimelerin anlamını ve bağlamını (context) anlamak için ağ eğitimi gerçekleştirilmektedir. Örneğin siyah ve beyaz kelimeleri ağa verildiğinde, benzer kelimeleri ve bağlamları tahmin edecektir dolayısıyla benzer temsil vektörleri olacaktır. CBOW mimarisi ise Skip-gram mimarisinin tersi olarak çalışmaktadır.

7.4. Word2vec

Bu model girilen bir metindeki ya da cümledeki kelimelerin sayısal vektörlerini, kelimelerin içeriklerini ya da anlamlarını göz önünde bulundurarak oluşturmayı hedeflemektedir. Benzer kelimelerin benzer içerikleri olduğu için bu kelimeler temsil uzayında birbirine yakın yerleştirilecektir. Aşağıdaki cümlelerden yola çıkarak basit ve kolay kelimelerinin çok benzer içerikleri olduğunu ve vektör uzayında da birbirine çok yakın yerleştirilmesi sonucu ortaya çıkmaktadır.

- *"Zor bir iş yapmak için tembel bir insan seçiyorum. Çünkü tembel bir insan bunu yapmanın **kolay** bir yolunu bulacaktır".*
  (Bill Gates)
- *"Entelektüel; **basit** bir şeyi karmaşık söyleyebilen kişidir; sanatçı ise zor bir şeyi **kolay".***
  (Charles Bukowski)
- *"Unix **basittir**, ancak ne kadar **basit** olduğunu anlamak için dahi olmanız gerekir".*
  (Dennis Ritchie)

Genellikle isimlerden önce gelen ve sıfat niteliği taşıyan basit ve kolay kelimelerin sayısal vektör gösterimleri benzer olacaktır. Böylece girdi cümlenin anlamsal özelliklerini yakalamak için cümle içindeki anlamsal olarak benzer kelimeler word2vec model ile tespit edilebilmektedir.



Şekil 7. 1. Word2vec modelinin örnek çalışma akışı

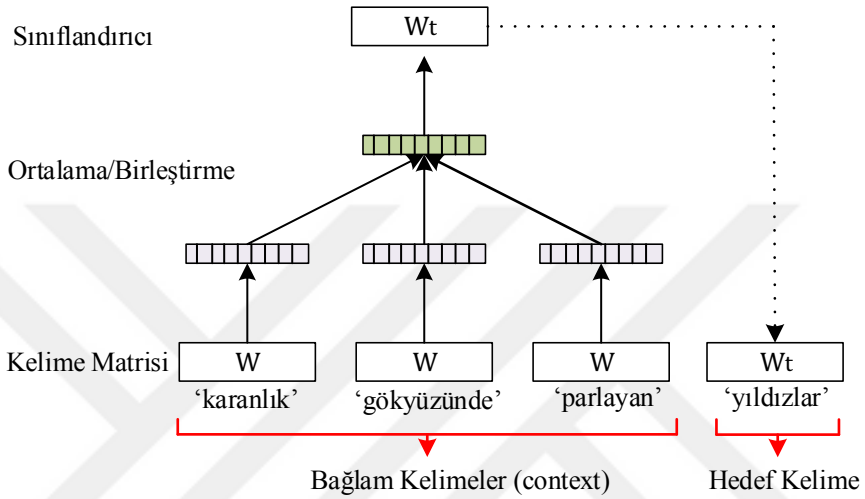
Word2vec modelinin eğitimi için Şekil 7.1'de gösterildiği gibi, öncelikle büyük bir doküman veri seti ağa girdi olarak verilmelidir. Bu veri seti noktalama işaretlerinin kaldırılması, az geçen kelimelerin kaldırılması ve Türkçe karakterlerin düzeltilmesi gibi ön işleme adımlarından geçirilerek word2vec modeline verilir. Word2vec modelinin ön işlemeden geçirilmiş eğitim seti üzerinde eğitilmesi sonucunda tüm benzersiz kelimeler için bir belirlenen temsil vektörü boyutunda ağırlık matrisi oluşacaktır. Word2vec modeli her seferinde dokümandaki bir cümleyi belirlenen pencere boyutunda taramaktadır ve penceredeki orta kelimenin sayısal vektörünü kelimenin bağlamlarına göre tahmin etmektedir.

7.4.1. Devamlı Kelime Torbaları (CBOW)

"Cümlelerin ve kelimelerin dağıtık gösterimleri" başlıklı makalede [159], word2vec modelinin daha açık gösterimi için Şekil 7.2' deki örnek bir kelime torbası modeli ele alınmıştır. Kelime matrisi birkaç grup kelimeyi alır ve ortalayarak ya da birleştirerek girdi alınan kelime grubuna bağlı bir sonraki kelimeyi tahmin etmektedir.

Şekil 7.2' de gösterilen kelime torbası word2vec model, "yıldızlar" kelimesini tahmin etmek için bu kelimenin bağlam kelimeleri olan "karanlık", "gökyüzünde" ve "parlayan" kelimelerini kullanmıştır. Her kelime W temsil matrisinde sayısal bir vektörle gösterilmektedir. Bu sayısal vektörlerin toplamı ya da birleşimi, bir cümledeki bir sonraki kelimenin tahmini için özellikler olarak kullanılmaktadır. Matematiksel ifade ile verilen $w_1, w_2, w_3 \dots w_t$ kelimelerinin bir

diziliminde, w_t hedef kelimenin vektörünü bulmak için modelin daha önce gördüğü bağlam kelime vektörlerinin toplamı kullanılmaktadır. Tipik bir dil modeli kelimeler ve onların bağlamlarının (komşuları ya da etrafındaki kelimeleri) olasılığını (likehood) en üst düzeye çıkarmak için eğitilmektedir. Kelime torbası word2vec model bağlamı verilen her kelimenin olasılığını bulmayı hedeflemektedir. Tüm dokümanın ortalama log olasılığını maksimize etmeyi hedefleyen amaç fonksiyonu aşağıdaki denklemde verilmiştir:



Şekil 7. 2. Kelime torbası modelinin gösterimi

$$\frac{1}{T}\sum_{t=1}^{T} \log\ P(w_t|\sum_{-c\leq j\leq c, j\neq 0} w_{t+j}) \tag{7.2}$$

Burada c pencere parametresi, hedef kelimenin bağlam kelimeleri ile tanımlanır. Softmax ile yapılacak olan çoklu sınıf tahmin denklemi aşağıdaki gibidir. Her y_i, her bir çıktı i kelime için normalleştirilmemiş log olasılığıdır:

$$p(w_t|w_{t-c},\dots,w_{t+c}) = \frac{exp^{y_{wt}}}{\sum_i exp^{y_i}}\ , y = b + Uh(w_{t-c},\dots,w_{t+c};W) \tag{7.3}$$

Burada U, b softmax parametreleridir ve h değeri, W matrisinden çıkarılan kelime vektörlerinin birleşimi ya da ortalaması ile bulunur.

7.4.2. Skip-gram Modeli

Skip-gram model verilen bir kelimenin ilişkili-etrafında olan kelimeleri tahmin etmektedir. Aşağıdaki örnek cümleden yola çıkarak, bu modelin nasıl çalıştığına bakılacaktır:

- *"Karanlık gökyüzünde parlayan yıldızlar güzel görünüyor."*

Bu cümle, daha önce anlatılan CBOW modelin eğitim çıktısı olarak aşağıdaki (bağlam kelimeler, hedef kelime) örnek çiftlerine bölünebilmektedir:

- ([parlayan, güzel], yıldızlar), pencere boyutu=1
- ([karanlık, parlayan, yıldızlar],gökyüzünde), pencere boyutu=2
- ([karanlık, gökyüzünde, parlayan], yıldızlar), pencere boyutu=3
- ([gökyüzünde, parlayan, yıldızlar, güzel], görünüyor), pencere boyutu=4

Skip-gram modelde ise Şekil 7.3' te gösterildiği gibi, CBOW mimarisinin tersi olarak verilen her bir merkezi kelimenin bağlam kelimeleri bulunmaktadır. Tablo 7.1'de yukarıda verilen cümle için 2 pencere boyutu kullanılarak yani her bir merkezi kelime için 2 kelime öncesi ve 2 kelime sonrasına bakılarak örneklenen (merkezi kelime, bağlam kelime) eğitim çiftlerinden bazı örnekler verilmektedir.

Skip-gram modelde ayrıca, sözlükteki rastgele kelimeler ile her bir giriş kelimesini eşleyerek negatif örnekler türetilmektedir:

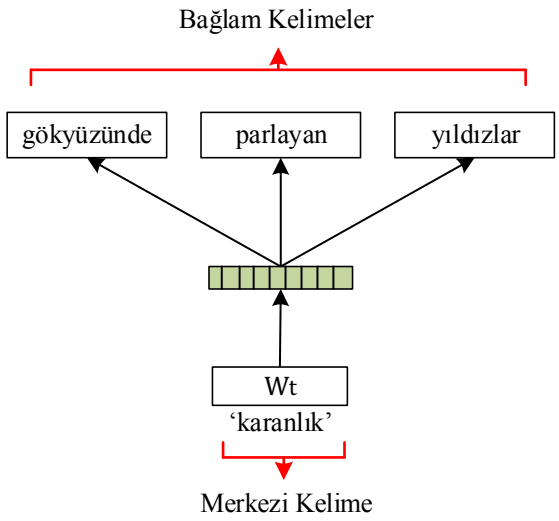
- *(parlayan, kedi), (parlayan, ev), (yıldızlar, yeşil), (yıldızlar, araba)(...)*

Son olarak, sınıflandırıcı için negatif ve pozitif örnekler üretilmektedir:

- *((parlayan, yıldızlar),1), ((parlayan, ev),0), ((yıldızlar, yeşil),0), ((yıldızlar, güzel),1)(...)*

Tablo 7. 1. Skip-gram modelinden bazı örnek eğitim çiftleri

Girdi Cümleler						**(merkezi kelime, bağlam kelime)**
k a r a n l ı k	gökyüzünde	parlayan	yıldızlar	g ü z e l	görünüyor	(karanlık, gökyüzünde) (karanlık, parlayan)
k a r a n l ı k	**gökyüzünde**	parlayan	yıldızlar	g ü z e l	görünüyor	(gökyüzünde, karanlık) (gökyüzünde, parlayan) (gökyüzünde, yıldızlar)
k a r a n l ı k	gökyüzünde	**parlayan**	yıldızlar	g ü z e l	görünüyor	(parlayan, karanlık) (parlayan, gökyüzünde) (parlayan, yıldızlar) (parlayan, güzel)
k a r a n l ı k	gökyüzünde	parlayan	**yıldızlar**	g ü z e l	görünüyor	(yıldızlar, karanlık) (yıldızlar, gökyüzünde) (yıldızlar, güzel) (yıldızlar, görünüyor)
k a r a n l ı k	gökyüzünde	parlayan	yıldızlar	**g ü z e l**	görünüyor	(güzel, parlayan) (güzel, yıldızlar) (güzel, görünüyor)



Şekil 7. 3. Skip-gram modelinin gösterimi

Matematiksel olarak, $w_1, w_2, w_3 \dots w_t$ eğitim kelimelerinin diziliminde Skip-gram modelinin amacı ortalama log olasılığını maksimize etmektir [16]:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log P(w_{t+j}|w_t) \tag{7.4}$$

Son olarak CBOW modelde olduğu gibi softmax fonksiyonu kullanarak tahmin yapılmaktadır. Skip-gram model eğitiminde önerilen bir diğer işlem alt-örneklemedir (subsampling). Çok sık geçen "bu", "ama", "bir" ya da "the", "a", "for" gibi kelimelerin bağlam kelimeleri pek anlamlı değildir. Bu kelimelerin bazıları atıldığında, veri seti hem gürültüden arındırılır hem de daha hızlı eğitim ve temsiller elde edilir. Eğitim setindeki her bir w_i kelimesi için aşağıdaki olasılık hesaplaması ile bu çok sık geçen kelimeler çıkarılabilmektedir:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{7.5}$$

Burada t eşik değeri ve $f(w_i)$, w_i kelimesinin frekansıdır. Sezgisel olarak bulunduğu söylenen bu formülün [16] nadir geçen kelimelerin vektörlerini öğrenmede performansı arttırdığı görülmüştür. Bu yönüyle literatürde CBOW modelin daha hızlı olmasına karşın Skip-gram mimarinin sık geçmeyen kelimelerin tahmininde daha iyi çalıştığı söylenmektedir.

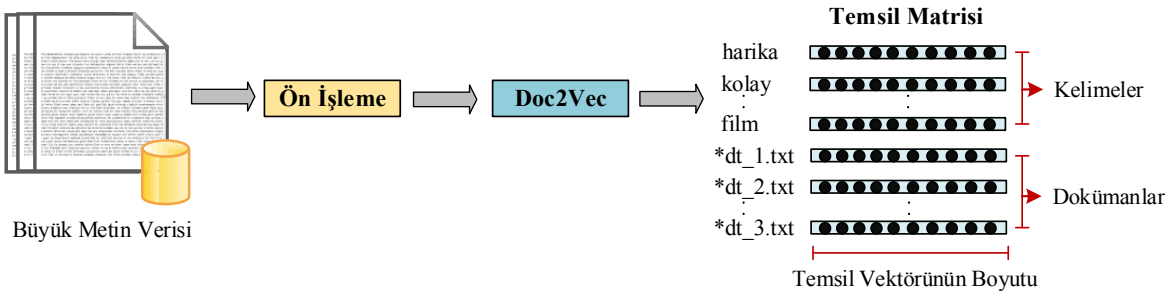
7.5. Doküman Vektörleri

Doküman vektörleri metin sınıflandırma ya da duygu analizi gibi NLP uygulamaları için doküman benzerlik ölçümlerinde yaygın olarak kullanılmaktadır. Doküman benzerlik çalışmalarında ön plana çıkan algoritmalardan birisi doc2vec ya da paragraf vektörüdür. Burada "paragraf" kelimesi, uzun cümle ya da bir paragraf bölümü olarak değişken uzunluktaki metin parçalarını ifade etmektedir. Paragraf vektörleri, önceki çalışmaların aksine cümleler,

paragraflar ve dokümanlar gibi farklı ya da değişken uzunluktaki girdi dizilimlerinin vektör gösterimlerini oluşturabilmektedir. Le ve Mikolov [159] makalelerinde önerdikleri paragraf vektörleri yönteminin avantajlarını göstermek için birçok veri seti üzerinde karşılaştırmalar yapmıştır. Kelime torbası ve diğer karmaşık yöntemler ile kıyaslandığında, duygu analizi testlerinde hata oranı açısından %16'dan daha fazla bir iyileşme sağlarken, metin sınıflandırma çalışmasında yaklaşık %30 geliştirme ile başarılı sonuçlar elde etmiştir.

7.6. Doc2Vec

Bir veya birden fazla cümlelerden oluşan dokümanların vektörlerinin elde edilmesinde word2vec kullanılarak öğrenilen kelime vektörlerinin ortalaması ya da ağırlıklı toplamının alınması gibi bazı yaklaşımlar sunulmuştur. Ancak bu yaklaşımlar ile aynı kelimelerin farklı dizilimlerde bulunduğu dokümanların aynı vektör uzayında eşlenmesi problemi oluşmuştur. Bu problemin çözümü için önerilen doc2vec modeli, kelime vektörlerini öğrenmek için oluşturulan derin sinir ağının girişine doküman id ya da paragraf id olarak isimlendirilecek olan bir paragraf tanımlayıcı eklenmektedir. Böylelikle sinir ağı girdi olarak doküman id (dokümanların 1-hot kodlanmış vektörü) ile birlikte ilgili dokümandaki kelimeleri alıp, farklı kelime uzunluklarındaki dokümanların vektör temsillerini öğrenebilmektedir. Şekil 7.4'te örnek bir doc2vec modelinin çalışma akış diyagramı verilmiştir.

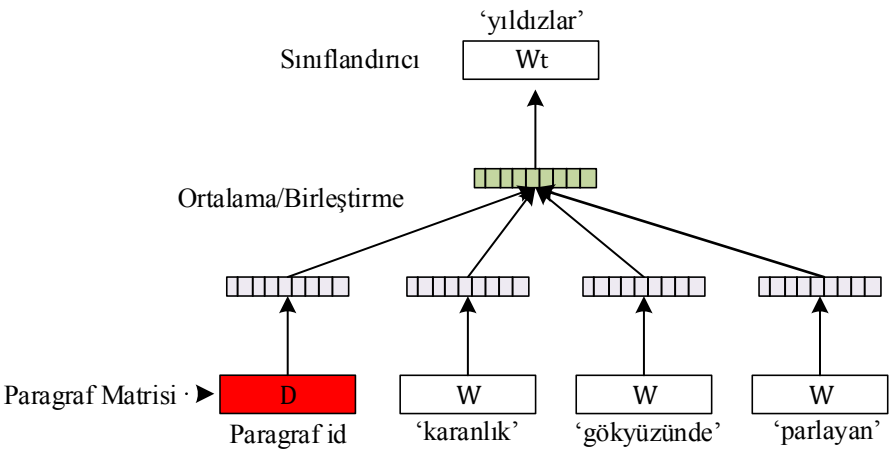


Şekil 7. 4. Doc2vec modelinin örnek çalışma akışı

Doc2vec modelinde, kelime vektörleri tüm doküman vektörlerinde her bir bağlam için güncellenirken doküman vektörleri yalnızca ilgili dokümandaki bağlamlar için güncellenmektedir. Doc2vec ya da paragraf vektör modeli, Dağıtık Bellek (dm) ve Dağıtık Kelime Torbaları (dbow) olmak üzere iki algoritma yöntemi sunmaktadır. Bu iki yöntem ile verilen bir dokümanın vektörleri ile birlikte dokümandaki her bir benzersiz kelimenin vektör gösterimleri öğrenilmektedir. Dağıtık bellek, CBOW word2vec modeline benzer şekilde çalışırken, dağıtık kelime torbaları skip-gram word2vec modeline benzer mantıkta çalışmaktadır.

7.6.1. Dağıtık Bellek (DM)

Dağıtık bellek (distributed memory) modeli, Şekil 7.2'de gösterilen devamlı kelime torbası modellerine benzerdir fakat paragraf vektör platformunda her paragraf D matrisindeki bir sütunla temsil edilen benzersiz bir vektörle eşlenirken ve her kelime de W matrisindeki bir sütunla temsil edilen benzersiz bir vektörle eşlenmektedir. Böylece Denklem 7.3'deki tek değişiklik h değerinin W matrisinden çıkarılan kelime vektörlerine ek olarak D matrisindeki paragraf vektörlerini de almasıdır.



Şekil 7. 5. Dağıtık bellek modelinin gösterimi

Şekil 7.5'deki dağıtık bellek modeline bakıldığında, paragraf vektörü ve kelime vektörlerinin bir grup bağlam kelimelerden bir hedef kelime tahmini yapmak için birleştirildiği ya da ortalandığı görülmektedir.

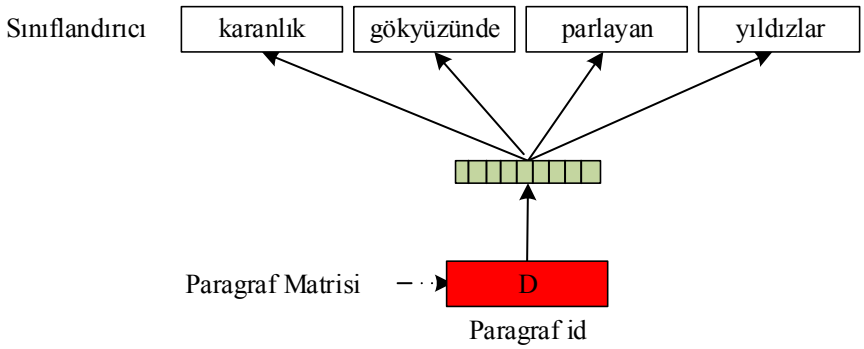
İsminden de anlaşılacağı üzere dağıtık bellek modelinde paragraf vektörü, mevcut bağlamdan ya da paragrafın konusundan neyin eksik olduğunu hatırlayan bir bellek olarak düşünülmektedir. Bağlam kelimeler, paragraf üzerinde bir pencere kaydırılarak ya da taranarak seçilmektedir. Paragraf vektörü aynı paragraftan üretilen ya da seçilen tüm bağlam kelimelerde paylaşılır, ancak tüm paragraflar arasında paylaşılmaz. Bununla birlikte tüm paragraflardaki kelimelerin aynı anlam ve içeriği bulunduğundan W kelime vektörleri matrisi paragraflar arasında paylaşılmaktadır.

Bu model, paragraf ve kelime vektörlerinin eğitiminde word2vec modelinde olduğu gibi olasılıksal gradyan azaltma tekniğini kullanmaktadır. Olasılıksal gradyan azaltmanın her adımında rastgele seçilmiş bir paragraftan sabit uzunluklu bir bağlam örneklenir. Geriye yayılım ile ağdan hata gradyanı hesaplanır ve daha iyi bir model elde etmek için bu gradyan model parametrelerini güncellemek için kullanılır.

Eğitilmiş modelin test edilme aşamasında, girdi olarak gelen yeni metnin paragraf vektörünün hesaplanması için gradyan azaltma ile çıkarım (inference) adımı gerçekleştirilir. Bu adımda W kelime vektörleri ve U, b softmax parametreleri sabit tutulurken gradyan azaltma yalnızca D paragraf vektörleri üzerinde uygulanır.

7.6.2. Dağıtık Kelime Torbaları (DBOW)

Dağıtık kelime torbaları, skip-gram modele benzer şekilde paragraf vektörü kullanarak bağlam kelimeleri tahmin etmeyi hedeflemektedir. Fakat Şekil 7.6'da görüldüğü gibi girdideki kelime ihmal edilerek, model çıktısında paragraftan rastgele örneklenmiş kelimeler tahmin edilmektedir. Pratik uygulamasında, olasılıksal gradyan azaltma işleminin her iterasyonunda belirlenen metin penceresinden rastgele bir kelime üretilir ve paragraf vektörü verilen bir sınıflandırma oluşturulur [159].



Şekil 7. 6. Dağıtık kelime torbaları modelinin gösterimi

Dağıtık kelime torbaları kavramsal olarak basittir ve dokuman vektörlerini öğrenirken kelime vektörlerinin saklanmasına gerek duymadığı için az veri depolaması sağlamaktadır.

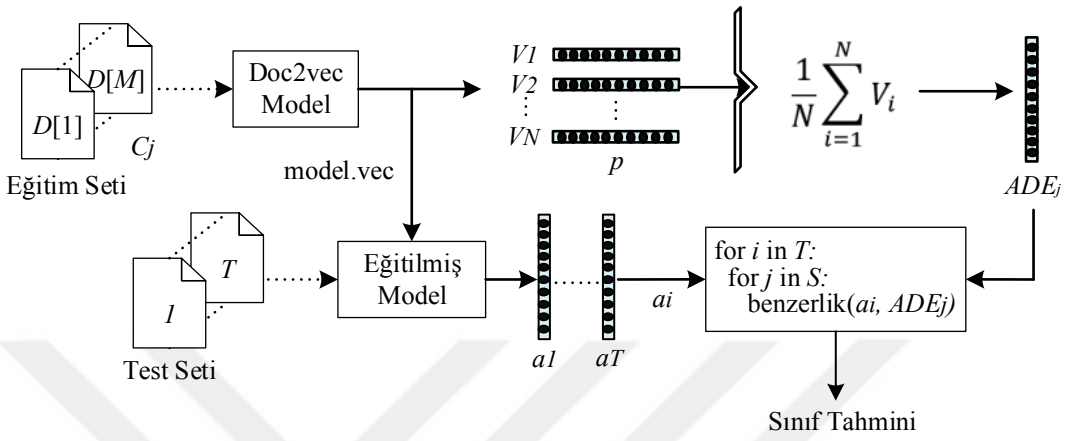
Doc2vec modellerinin avantajları ve dezavantajları özet olarak aşağıda verilmiştir:

- *Avantajları:* Kelimelerin anlamlarını çıkarabilmesi ve kelime sıralarını göz önünde bulundurabilmesi yönüyle kelime torbası ve n-gram modellere göre üstünlük sağlamaktadır. Orijinal makalesinde belirtilen en önemli avantajı ise etiketsiz veriden öğrenebilmesi ve yeterince etiketli verisi olmayan işlemler için iyi çalışmasıdır.
- *Dezavantajları:* Yeni bir metin için paragraf vektörü çıkarımı aşamasında olasılıksal gradyan azaltma gibi yinelemeli (iterative) optimizasyon ve çoklu gradyan hesaplamaları gerektirmektedir. Her yeni bir metin ya da doküman için böyle bir optimizasyon çalıştırılması maliyetli bir durumdur. Ayrıca her yeni gelen doküman, dokuman koleksiyonuna eklenir ve model eğitiminin yeniden başlatılma gereksinimi ortaya çıkar. Dahası, farklı uzunluklarda ve içeriklerde kelimelerden oluşan dokümanların aşırı ya da eksik öğrenme problemlerinin kontrolü zor olacaktır [250].

7.7. Ortalama Doküman Vektörü (ADE) Yaklaşımı

Duygu sınıflandırması çalışması için her bir yorum bir doküman olarak düşünülmektedir. Word2vec modellere benzer şekilde paragraf vektörleri, bu çalışmada kullanıldığı ismi ile

doküman vektörleri, verilen bir metin bölümünün anlamsal içeriğini en iyi şekilde yakaladığı için, her bir yorumun duygusunu çıkarmada tercih edilmiştir. Önerilen ADE yaklaşımının çalışma sistemi Şekil 7.7'de sunulmuştur:



Şekil 7. 7. Ortalama Doküman Vektörleri (ADE) yaklaşımı

Şekil 7.7'de gösterilen yaklaşımda kullanılan semboller ve metrikler aşağıda açıklanmıştır:

- C_J, ilgili sınıf numarası (id), D dokümanı ve C sınıfı temsil etmektedir.
- M, eğitim setindeki toplam doküman sayısını ve V eğitim dokümanları vektörünü göstermektedir.
- N, ilgili C_J sınıfındaki toplam doküman sayısıdır.
- p, dokümanların modelde belirlenen sabit uzunluktaki vektör boyutudur.
- ADE_{CJ} , ilgili C_J sınıfının ortalama doküman temsili gösterimidir.
- T, test setindeki toplam doküman sayısını göstermektedir
- $a_i, 1 \leq i \leq T$ olmak üzere i. test dokümanının vektörünü göstermektedir.
- $\{j \in \mathbb{Z} : 1 \leq j \leq S\}$ olmak üzere ADE_j, C_J sınıfının ortalama doküman vektörüdür, S burada sınıf sayısıdır.

Her hangi bir metin sınıflandırma işleminde, $C_1, C_2 \ldots C_s$ sınıflarına ait M sayıda eğitim dokümanının doküman id ile indekslenmiş etiketi ve içeriği oluşturulan farklı doc2vec modelleri

ile eğitilerek, M sayıda doküman etiketi ve doküman vektörü elde edilmektedir. C_J sınıfına ait $V_1, V_2 \ldots V_N$ doküman vektörleri toplamının , N doküman sayısına oranlanması ile C_J sınıfa ait ADE_J ortalama doküman temsili elde edilmektedir:

$$\{j \in \mathbb{Z} : 1 \leq j \leq S\}, ADE_j = \frac{1}{N}\sum_{i=1}^{N} V_i \quad (7.6)$$

Her sınıfın ortalama doküman vektörü bu şekilde hesaplandıktan sonra, sınıfı bilinmeyen yeni bir test dokümanın doküman vektörü alınarak her bir sınıfın ortalama doküman vektörü ile arasında kosinüs benzerlik hesaplaması gerçekleştirilir. Yeni test dokümanı, en yüksek benzerlik ile sonuçlanan ilgili sınıfa yerleştirilir.

7.8. Veri Seti ve Geliştirme Ortamı

Duygu sınıflandırması çalışması için iki veri seti kullanılmıştır. İlki 50.000 İngilizce film yorumu içeren IMDB [251] veri seti (indirme linki: Stanford Üniversitesi Yapay Zeka Laboratuvarı, http://ai.stanford.edu/), ikincisi ise 40.000 Türkçe film yorumu içeren BEYAZPERDE [252] veri setidir (indirme linki: Fırat Üniversitesi Büyük Veri ve Yapay Zeka Laboratuvarı, http://buyukveri.firat.edu.tr/). Bu tez çalışması kapsamında sunulan beyazperde film yorumları veri seti, Türkçe duygu sınıflandırma çalışmalarında doğru ve güvenilir karşılaştırmalar yapmak için oluşturulmuştur. Geliştirilen Java tabanlı bir web kazıma (web scraping) aracı [253] ile www.beyazperde.com sitesinden kullanıcı ve editör yorumları toplanmıştır. Oluşturulan veri seti; sayılar, işaretler ve çeşitli simgeler kaldırılması gibi ön işleme adımlarından geçtikten sonra yaklaşık 13 milyon kelime ve 80 bin benzersiz kelime içermektedir. Tablo 7.1, bu veri setinde, kullanıcıların verdiği puanlara göre yorum sayılarının dağılımını göstermektedir.

Tablo 7. 1. BEYAZPERDE etiketli veri seti puan dağılımı

Puan	0	1	2	3	4	5	**Toplam**
Yorum Sayısı	4.879	14.288	33.665	14.425	66.946	21.946	155.816

Beyazperde sitesinde yayınlanan filmlere verilen puanlara göre pozitif ve negatif etiketleme yapılmıştır. 0 ve 1 puan alan yorumlar negatif, 5 puan alan yorumlar pozitif olarak kabul edilmiştir. Pozitif ve negatif film yorumlarını içeren veri setinden öne çıkan 1, 2 ve 3 gram kelime bulutları Şekil 7.8'de gösterilmiştir.

Pozitif Kelime Bulutları

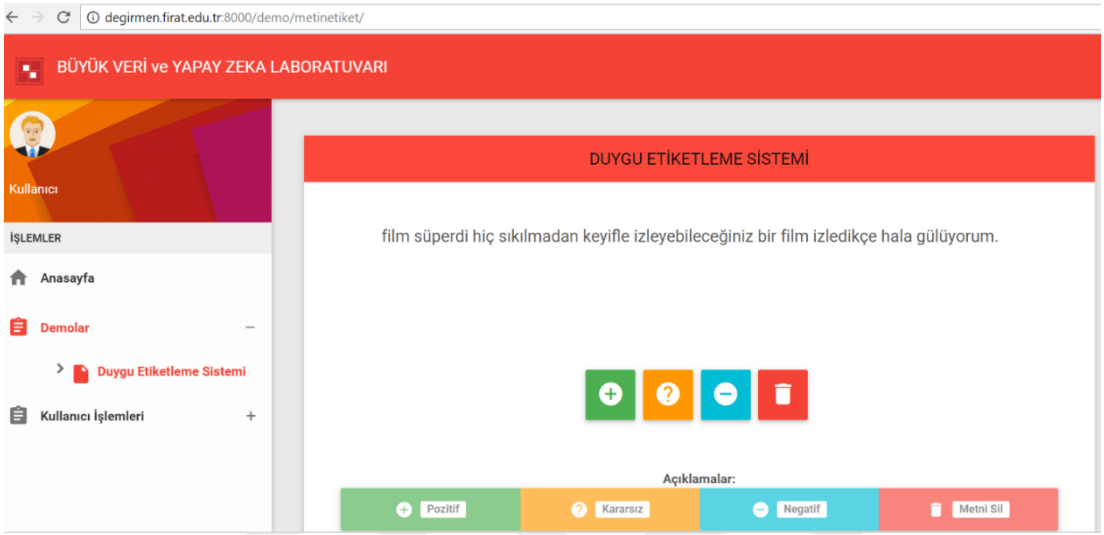
Negatif Kelime Bulutları

Şekil 7. 8. BEYAZPERDE veri setindeki pozitif ve negatif n-gram kelime bulutları

Türkçe film yorumlarından eğitim setinden bağımsız ayrı bir test seti oluşturmak için web kazıma aracı ile www.sinemalar.com sitesinden puanlanmamış kullanıcı yorumları toplanmıştır. Toplanan bu yorumları model sonucu tahmin edilen sınıf etiketi ile karşılaştırmak için Şekil 7.9'daki Duygu Etiketleme Sistemi geliştirilmiştir. Bu sistemin ara yüzü, Python programla dili ile yazılmıştır ve açık kaynak Django web uygulama platformunda geliştirilmiştir. Verilerin saklanması ve sorgulanması için NoSQL veri tabanlarından MongoDB kullanılmıştır. Film yorumları, MongoDB veri tabanında id ve yorum içeriği şeklinde aşağıdaki gibi tutulmaktadır.

```
_id: ObjectId("Sada67942294bd3cf4b892a0")
yorum: "film süperdi hiç sıkılmadan keyifle izleyebileceğiniz bir
        film izledikçe hala gülüyorum"
id: 1
```

Bu yorumlar rastgele id sırasına göre sistem ara yüzünde görüntülenip kullanıcının etiketlemesine sunulmaktadır. Gösterilen yorumların doğru duygusunun etiketlenmesi ve kontrolü amacıyla en az üç kere aynı duygu etiketi (pozitif, negatif ya da nötr) sayısına ulaşması gerekmektedir.

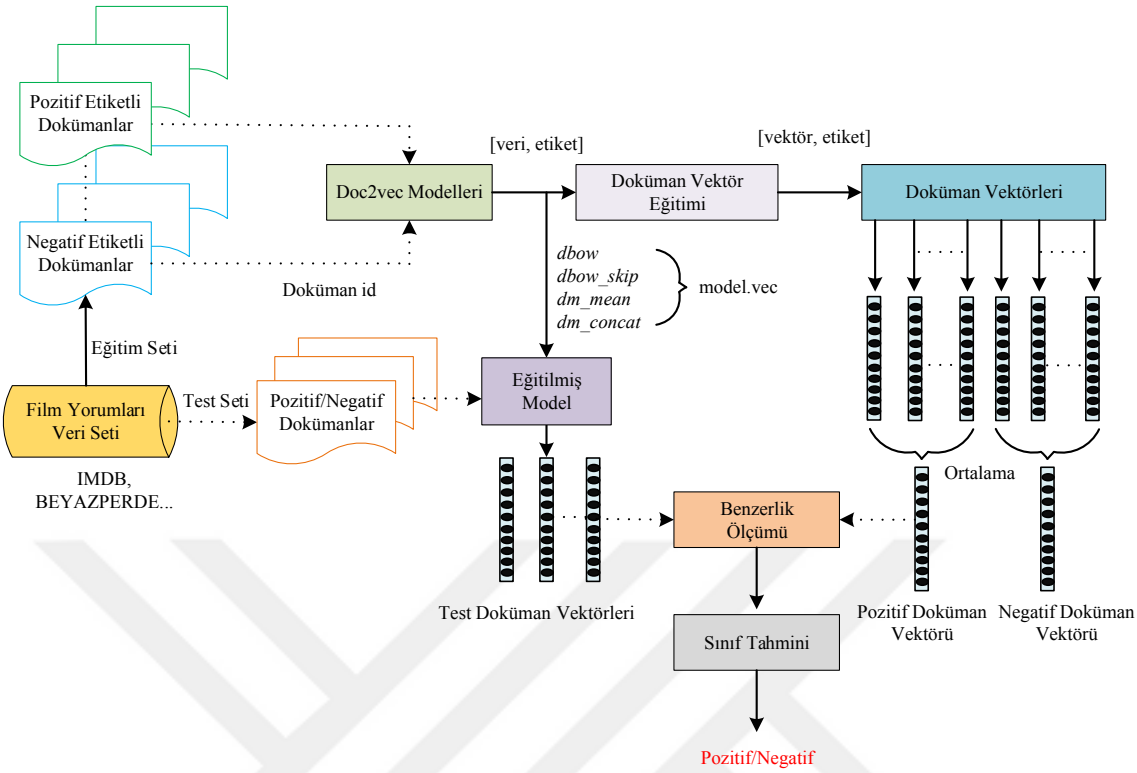


Şekil 7. 9. Etiketi bilinmeyen film yorumları için geliştirilen duygu etiketleme sistemi

Bu tez çalışmasında yapılan eğitim ve testler Ubuntu 14.04 işletim sistemi yüklü, Intel Xeon X5570 2.93-GHz işlemciye ve 85 GB hafızaya sahip bir sunucu üzerinde gerçekleştirilmiştir. Eğitim ve test uygulamalarının gerçekleştirilmesi için Python dili tercih edilmiştir.

7.9. Önerilen Yaklaşım ile Duygu Sınıflandırması

Bu çalışmanın amacı önerilen Ortalama Doküman Vektörleri (Averaged Document Embedding - ADE) metin sınıflandırma yöntemi ile duygu analizi gerçekleştirmektir. Yöntemin performans başarısını ortaya koymak ve her iki İngilizce ve Türkçe metin dili üzerinde karşılaştırmasını değerlendirmek için film yorumları veri setleri üzerinde pozitif ve negatif olmak üzere ikili duygu sınıflandırması gerçekleştirilmiştir. Duygu sınıflandırması sistemine genel bakış Şekil 7.10'da sunulmuştur.



Şekil 7. 10. Ortalama doküman vektörleri yaklaşımı ile ikili duygu sınıflandırması

Pozitif ve negatif film yorumları içeren dokümanlar, doküman id ile indekslenerek [doküman içeriği, doküman etiketi] bilgisi ile aşağıda açıklanan doc2vec modellerine eğitilmek üzere verilmektedir:

Dağıtık Kelime Torbaları (DBOW): N sayıda doküman içeren bir eğitim seti için, eğitim yapılırken ağın girdi katmanındaki her bir doküman 1-hot vektör ile kodlanacaktır. Böylelikle girdi katmanındaki her bir birim ya da nöron eğitim setindeki tek bir doküman id ile eşleşecektir. Ağa giren bir dokümanın tanımlayıcısı (id) ilişkili olduğu birimde 1 olacak ve diğer birimler 0 olacaktır. Dolayısıyla ağın girdisi olarak yalnızca doküman id bilgisi kullanılacaktır. Girdi katman ile gizli katman arasında tam bağlantı oluşturularak girdi katmandaki her birim yani doküman id birimi gizli katmandaki birimlerle eşlenecektir. Bu birimler arasındaki ağırlıklar başlangıç olarak çok küçük değerler ile rastgele seçilmektedir. Gizli katmanının boyutu p

paragraf vektör boyutunda olacaktır. D ağırlık matrisinin boyutu $p \times N$ olacaktır. Sözlükte M sayıda benzersiz kelime olduğu düşünüldüğünde çıktı katmanında M sayıda birim olacaktır. Bu durumda gizli katman ile çıktı katman birimleri arasındaki U ağırlık matrisinin boyutu $M \times p$ olacaktır. Dağıtık kelime torbaları modelinde, ağa girdi olarak verilen doküman id ile ilişkili dokümandan rastgele örneklenmiş kelimelerin tahmini için çıktı katmanın aktivasyonları softmax aktivasyonundan geçirilerek çok sınıflı sınıflandırma işlemi gerçekleştirilmektedir.

Bu tez çalışmasında doküman vektörlerinin eğitimi için iki farklı dağıtık kelime torbaları modeli kullanılmıştır. Kısaca *dbow* olarak adlandırılacak olan ilk model yukarıda anlatılan ve paragraf vektör yönteminin önerildiği ilk çalışmada [159] tanıtılmıştır. Burada doküman vektörlerinin eğitiminde yalnızca doküman id ile tanımlanan doküman vektörleri kullanılmaktadır. *dbow_skip* olarak isimlendirilecek olan ikinci model ise dağıtık kelime torbaları modelinin eş zamanlı olarak skip-gram model ile kelime vektörleri ve doküman vektörlerinin birlikte eğitildiği modeldir. Dai ve arkadaşlarının yayınladığı paragraf vektörleri ile doküman temsil etme çalışmasında [244], bu modelin skip-gram modelde olduğu gibi kelime vektörleri eklenerek eğitilmesinde doküman vektörlerinin daha iyi çalıştığının gözlemlendiği söylenmiştir.

Dağıtık Bellek (DM): N sayıda doküman ve sözlükte M sayıda benzersiz kelime içeren bir eğitim setinde dağıtık bellek modelinde kelime vektörlerinin ve doküman vektörlerinin öğrenilmesi istenildiğinde gizli katmanın boyutu p doküman vektör ve q kelime vektör boyutunda olacaktır. Bu durumda ağın parametre sayısı $N \times p \times M \times q$ ile sonuçlanacaktır [159]. Doküman vektörleri, dağıtık kelime torbaları modelinde olduğu gibi burada da 1-hot vektör ile kodlanarak doküman id ile temsil edilecektir. Ağa girdi olarak verilen dokuman id ve bağlam kelime vektörlerinden hedef kelime tahmini için çıktı katmanın aktivasyonları softmax aktivasyonundan geçirilerek çok sınıflı sınıflandırma işlemi gerçekleştirilmektedir. Doküman vektörlerinin eğitimi için iki farklı dağıtık bellek modeli kullanılmıştır. *dm_mean* ile isimlendirilecek ilk modelde dokuman vektörleri ve kelime vektörlerinin ortalaması alınmaktadır. *dm_concat* ile gösterilen ikinci modelde ise doküman vektörleri ile kelime vektörleri birleştirilerek (concatenating) ilgili dokümanların tahmini vektör hesaplaması

gerçekleştirilir. *dbow*, *dbow_skip*, *dm_mean* ve *dm_concat* modelleri ile doküman vektörlerinin elde edilmesinde, paragraf vektör uygulamalarında standart olarak kabul edilen Gensim kütüphanesi [254] kullanılmıştır.

Çıkarılan doküman vektörleri "model.vec" uzantılı dosyalar, [doküman etiketi, doküman vektörü] bilgisi ile saklanmaktadır. Burada etiket dosya ismidir. ADE yaklaşımı ile pozitif ve negatif duygu etiketli dokümanların ortalama vektör değeri hesaplanmaktadır. Bu işlem sonucu iki sınıfı temsil eden pozitif ve negatif ortalama doküman vektörü olmak üzere iki vektör elde edilmektedir. Bir dokümanın ya da cümlenin tüm terimlerinin (kelimelerinin) vektör olarak temsil ediliği vektör uzay modelinde sıklıkla kosinüs benzerlik tercih edilmektedir. Burada test setinden seçilen yeni etiketsiz dokümanların vektörleri ile bu iki vektör arasında kosinüs benzerlik hesaplaması gerçekleştirilerek, ilgili test dokümanların hangi sınıfa yakın olduğu tahmin edilmektedir. Literatürde doc2vec ile doküman vektörlerinin doğru sınıflandırmasının yapılması için tüm doküman vektörlerinin birbirleri ile uzaklıklarına bakılmaktadır. Zaman maliyetli ve karmaşık çok fazla hesaplama gerektiren bu yöntemlerin aksine önerilen yaklaşımla yeni gelen test metinleri yalnızca sınıf sayısı kadar vektör ile kıyaslanmaktadır. Bir test dokümanı yalnızca sınıf sayısı kadar vektör ile kıyaslanacağı doğrusal regresyon ya da Destek Vektör Makineleri yöntemleri aksine çok basit bir sınıflandırma hesaplaması gerekecektir. Doküman benzerlik çalışmalarında da kullanılabilecek bu yaklaşım ile geleneksel yöntemlerin aksine *t* test süresi *t/sınıf sayısı* kadar kısalacaktır. Bunun sonucunda ilgili dokümana benzer dokümanlar yalnızca sınıfındaki dokümanlara bakılarak tespit edilecektir.

7.10. Önerilen Yaklaşımın Duygu Sınıflandırması Üzerinde Performans Sonuçları

Bu çalışmanın amacı önerilen ADE metin sınıflandırma yöntemi ile duygu analizi gerçekleştirmektir. Yöntemin performans başarısını ortaya koymak ve her iki İngilizce ve Türkçe metin dili üzerinde karşılaştırmasını değerlendirmek için IMDB ve BEYAZPERDE film yorumları veri setleri üzerinde pozitif ve negatif olmak üzere ikili duygu sınıflandırması gerçekleştirilmiştir. Ayrıca *dbow*, *dbow_skip*, *dm_mean* ve *dm_concat* paragraf vektörü modelleri ile elde edilen doküman vektörlerinin önerilen ortalama doküman vektörleri yaklaşımı

ile sınıflandırmasındaki performans başarıları karşılaştırılmıştır. Doküman olarak düşünülen yeni bir film yorumunun doküman vektörünün, ortalama pozitif ve ortalama negatif doküman vektörü ile karşılaştırılmasının yapılması için kosinüs benzerlik kullanılmıştır.

7.10.1. Değerlendirme Ölçütleri

Gerçekleştirilen ortalama doküman vektörleri yönteminin sınıflandırma sonuçlarını doğru bir şekilde değerlendirmek ve farklı parametre değerli ile karşılaştırmak için kullanılan ölçütler aşağıda özetlenmiştir.

1. *Doğruluk (Accuracy):* Makine öğrenmesi ve veri analizi çalışmalarında en yaygın kullanılan ölçüttür. Doğru tahminlerin yani doğru sınıflandırılmış dokümanların (film yorumlarının) sayısının toplam doküman sayısına oranıdır:

$$Doğruluk = \frac{tp+tn}{tp+tn+fp+fn} \tag{7.7}$$

2. *Kesinlik (Precision):* Pozitif olarak sınıflandırılmış ve gerçekte de pozitif etiketli olan yorumların yani doğru tahmin edilmiş pozitif yorumların sayısının, pozitif olarak sınıflandırılmış tüm yorumlara oranıdır:

$$Kesinlik = \frac{tp}{tp+fp} \tag{7.8}$$

3. *Hassasiyet (Recall):* Doğru tahmin edilmiş pozitif yorumların sayısının, tüm pozitif etiketli yorumlara oranıdır:

$$Hassasiyet = \frac{tp}{tp+fn} \tag{7.9}$$

4. *F-ölçütü (F-measure, F1-score):* İkili sınıflandırma problemlerinde test doğruluğu için kullanılan bir ölçüttür. Kesinlik ve hassasiyet ölçütlerinin harmonik ortalaması ile hesaplanmaktadır. Bu ölçüt 1 değerine ne kadar yakınsa yöntemin o kadar iyi çalıştığı sonucunu göstermektedir.

$$F = \frac{2\times(Kesinlik\times Hassasiyet)}{(Kesinlik+Hassasiyet)} \tag{7.10}$$

Bu ölçütlerin hesaplamasında kullanılan *tp*, *tn*, *fp* ve *fn* terimleri şunlardır:

- *Doğru tahmin edilen pozitif yorum sayısı (true positive-tp)*: Sınıflandırıcı, verilen bir film yorumunu pozitif yani olumlu olarak tahmin ederse ve yorumun gerçek etiketi de pozitifse *tp* kabul edilir. Burada *tp* sayısı pozitif tahmin edilmiş, gerçekte pozitif etiketli yorumların sayısını göstermektedir.
- *Doğru tahmin edilen negatif yorum sayısı (true negative-tn):* Sınıflandırıcı, verilen bir film yorumunu negatif olarak tahmin ederse ve yorumun gerçek etiketi de negatifse *tn* kabul edilir. Burada *tn* sayısı negatif tahmin edilmiş, gerçekte negatif etiketli yorumların sayısını göstermektedir.
- *Yanlış tahmin edilen pozitif yorum sayısı (false positive-fp)*: Sınıflandırıcı, verilen bir film yorumunu negatif tahmin ederse ve yorumun gerçek etiketi de pozitifse *fp* kabul edilir. Burada *fp* sayısı negatif tahmin edilmiş, gerçekte ise pozitif etiketli yorumların sayısını göstermektedir.
- *Yanlış tahmin edilen negatif yorum sayısı (false negative-fn)*: Sınıflandırıcı, verilen bir film yorumunu pozitif tahmin ederse ve yorumun gerçek etiketi de negatifse *fn* kabul edilir. Burada *fn* sayısı pozitif tahmin edilmiş, gerçekte ise negatif etiketli yorumların sayısını göstermektedir.

7.10.2. Hiper Parametre Optimizasyonu

Tablo 7. 2. Hiper Parametre Açıklamaları

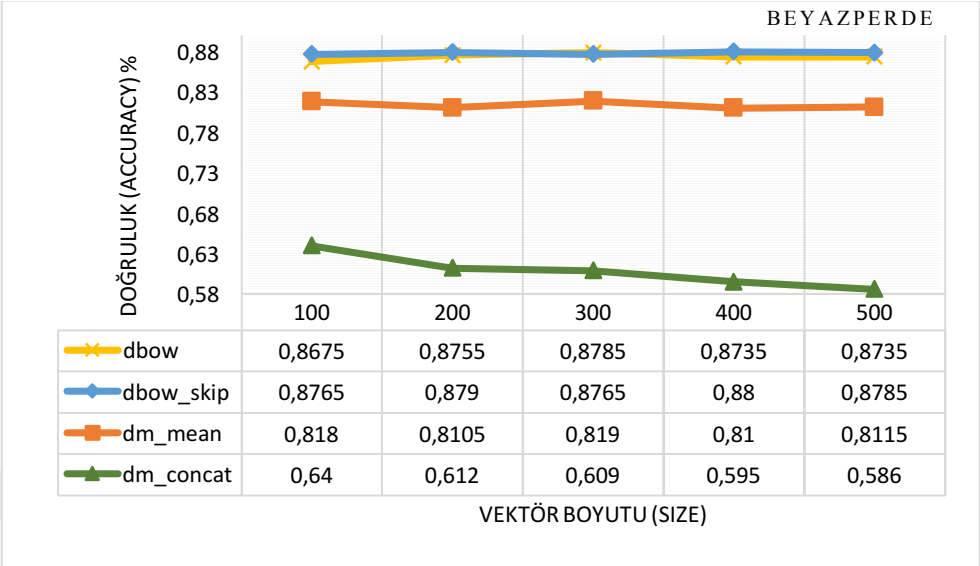
Hiper Parametre	**Tanımı**
Vektör Boyutu (size)	Doküman vektörünün boyutu
Devir Sayısı (epoch)	Eğitim iterasyonlarının sayısı
Minimum Geçme Sayısı (min_count)	Kelimeler için minimum eşik frekansı yani geçme sayısı
Alt-örnekleme (sample)	Yüksek frekanslı kelimeler için eşik değeri
Negatif Örnekleme (negative)	Negatif kelime örneklerinin sayısı
Pencere (window)	Sağ/sol bağlam kelime pencere boyutu

Pozitif ve negatif olmak üzere iki sınıfa ait ortalama doküman vektörlerinin elde edilmesinde [159]'te tanımlanan paragraf vektör modelleri kullanılmıştır. Bu tez çalışması için seçilen *dbow*, *dbow_skip*, *dm_mean* ve *dm_concat* modellerinde kullanılan hiper parametreler ve tanımları Tablo 7.2'de verilmiştir.

Aşağıdaki kod parçası Gensim Kütüphanesi kullanılarak gösterilen modellerin oluşturulmasındaki örnek hiper parametrelerin kullanımını vermektedir:

```
import gensim.models as gsm
dbow_model= gsm.Doc2Vec(dm=0, dbow_words=0, size=100, window=4,
                        negative=5, sample=1e-3, min_count=5, workers=8)
dbow_skipgram_model= gsm.Doc2Vec(dm=0, dbow_words=1, size=100, window=4,
                        negative=5, sample=1e-3, min_count=5, workers=8)
dm_mean_model= gsm.Doc2Vec(dm=1, dm_mean=1, size=100, window=4,
                        negative=5, sample=1e-3, min_count=5, workers=8)
dm_concat_model= gsm.Doc2Vec(dm=1, dm_concat=1, size=100, window=4,
                        negative=5, sample=1e-3, min_count=5, workers=8)
```

7.10.3. Uygulama Sonuçları



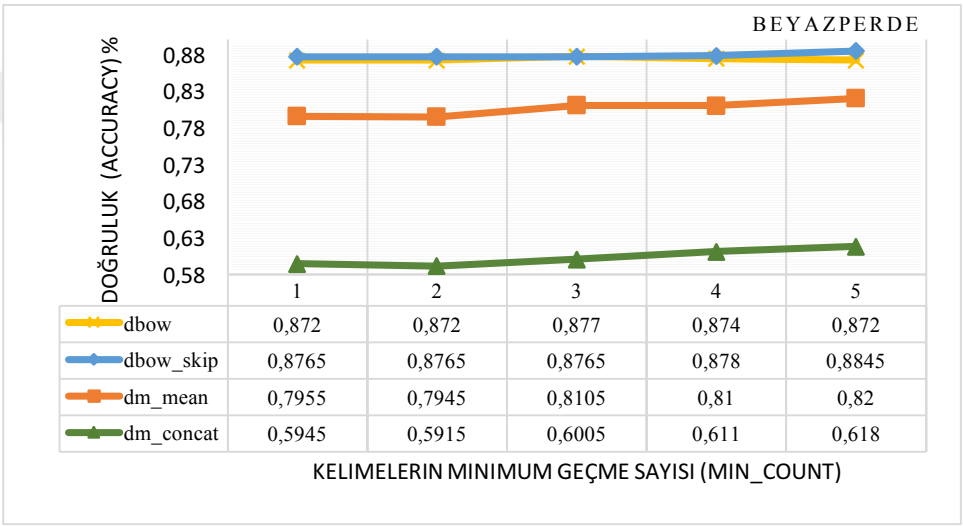
Şekil 7. 11. Farklı doküman vektör boyutlarına göre ortalama doküman vektörlerinin duygu sınıflandırması sonuçları

Tablo 7. 3. Vektör boyutuna göre duygu sınıflandırması için hesaplanan ölçütler

		dbow	**dbow_skip**	**dm_concat**	**dm_mean**
boyut 100	Kesinlik	0,860	0,868	0,650	0,838
	Hassasiyet	0,886	0,894	0,661	0,821
	F-ölçütü	0,872	0,881	0,656	0,829
	Doğruluk	0,868	0,877	0,640	0,818
boyut 300	Kesinlik	0,866	0,866	0,606	0,839
	Hassasiyet	0,531	0,896	0,635	0,822
	F-ölçütü	0,659	0,881	0,620	0,830
	Doğruluk	**0,879**	0,877	0,609	0,819
boyut 500	Kesinlik	0,864	0,865	0,590	0,830
	Hassasiyet	0,892	0,900	0,611	0,815
	F-ölçütü	0,878	0,882	0,600	0,823
	Doğruluk	0,874	0,879	0,586	0,812

Değerlendirme ölçütleri ve hiper parametre seçimlerine göre paragraf vektör modelleri kullanılarak gerçekleştirilen ortalama doküman vektörleri yaklaşımının BEYAZPERDE ve IMDB film yorumları veri seti üzerindeki performans sonuçları aşağıda sunulmuştur.

Şekil 7.11'de 100, 200, 300, 400 ve 500 olmak üzere farklı doküman vektör boyutlarına, Şekil 7.12'de 1, 2, 3, 4 ve 5 olmak üzere kelimelerin minimum geçme sayılarına ve 10, 100 ve 1000 olmak üzere eğitim devri sayısına göre elde edilen dört farklı doc2vec model tabanlı ortalama doküman vektörlerin BEYAZPERDE film yorumları veri seti üzerinde duygu sınıflandırması doğruluk sonuçları verilmiştir.



	1	2	3	4	5
dbow	0,872	0,872	0,877	0,874	0,872
dbow_skip	0,8765	0,8765	0,8765	0,878	0,8845
dm_mean	0,7955	0,7945	0,8105	0,81	0,82
dm_concat	0,5945	0,5915	0,6005	0,611	0,618

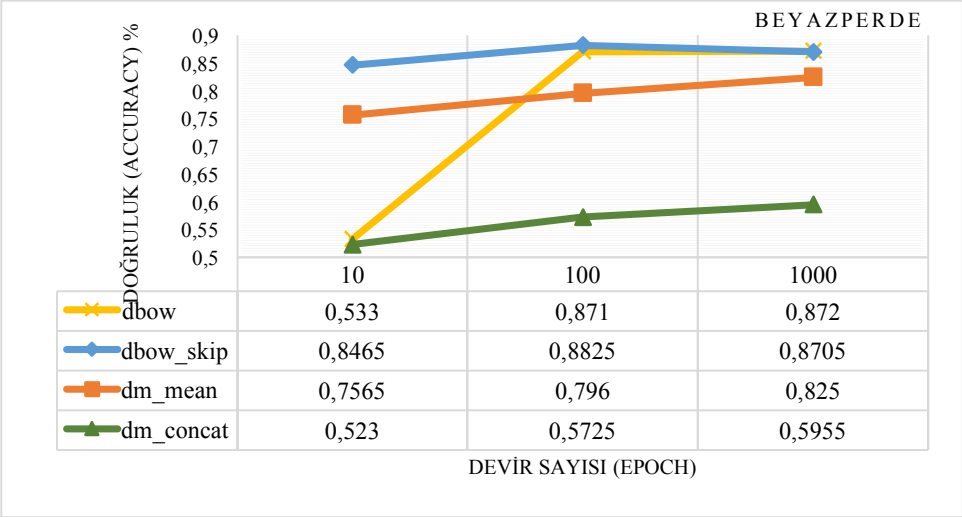
Şekil 7. 12. Kelimelerin minimum geçme sayısına göre ortalama doküman vektörlerinin duygu sınıflandırması sonuçları (BEYAZPERDE veri seti)

dbow ve *dbow_skip* modelleri ile oluşturulan pozitif ve negatif ortalama vektörleri, Türkçe duygusunu sınıflandırmada *dm_mean* ve *dm_concat* modellerine göre daha iyi sonuç vermiştir. 5 minimum geçme sayısı, 100 eğitim devir sayısı ve 300 vektör boyutu hiper parametre seçimi ile Tablo 7.4'te daha net görüleceği gibi *dbow_skip* modeli %88,5 doğruluk oranı ile en yüksek sonucu almıştır.

Tablo 7.3'te koyu renkle gösterilen *dbow* modeli en iyi %87,9 doğruluk oranına ulaşırken Tablo 7.5'deki *dm_mean* ve Tablo 7.3'deki *dm_concat* modelleri sırasıyla en yüksek %82,5 ve %64 doğruluk oranında kötü bir sınıflandırma sağlamıştır. Bu sonuçlar doğrultusunda Türkçe metinlerden doküman vektörlerin elde edilmesinde *dm_concat* modeli önerilmemektedir.

Tablo 7. 4. Kelimelerin minimum geçme sayısına göre duygu sınıflandırması için hesaplanan ölçütler

		dbow	dbow_skip	dm_concat	dm_mean
min_count 1	Kesinlik	0,856	0,871	0,592	0,819
	Hassasiyet	0,897	0,892	0,621	0,798
	F-ölçütü	0,876	0,881	0,606	0,808
	Doğruluk	0,872	0,877	0,595	0,796
min_count 3	Kesinlik	0,873	0,870	0,612	0,825
	Hassasiyet	0,531	0,893	0,623	0,817
	F-ölçütü	0,877	0,881	0,618	0,821
	Doğruluk	0,877	0,877	0,601	0,811
min_count 5	Kesinlik	0,861	0,878	0,636	0,832
	Hassasiyet	0,532	0,901	0,638	0,827
	F-ölçütü	0,657	0,889	0,637	0,830
	Doğruluk	0,872	**0,885**	0,618	0,820

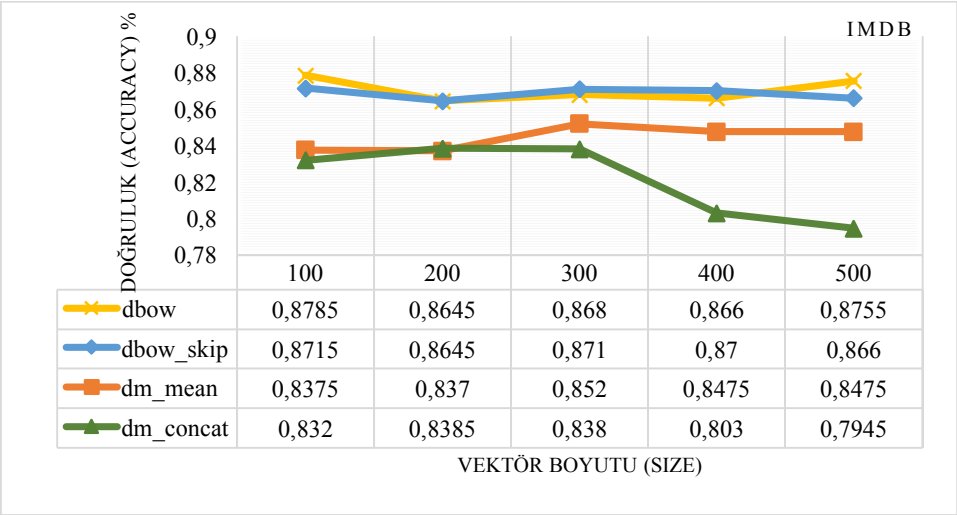


Şekil 7. 13. Devir sayısına göre ortalama doküman vektörlerinin duygu sınıflandırması sonuçları (BEYAZPERDE veri seti)

Tablo 7. 5. Devir sayısına göre duygu sınıflandırması için hesaplanan ölçütleri

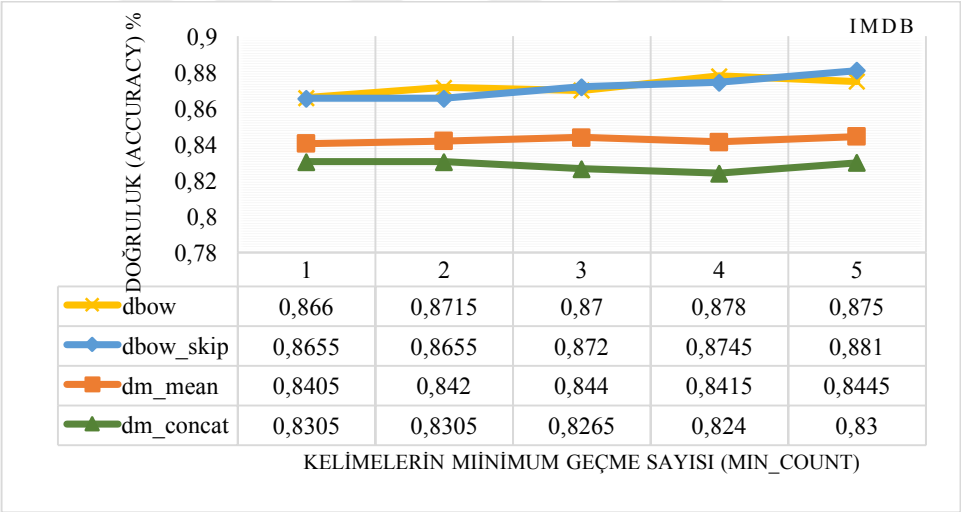
		dbow	dbow_skip	dm_concat	dm_mean
epoch 10	Kesinlik	0,472	0,843	0,491	0,803
	Hassasiyet	0,561	0,862	0,553	0,752
	F-ölçütü	0,513	0,853	0,521	0,777
	Doğruluk	0,533	0,847	0,523	0,757
epoch 100	Kesinlik	0,861	0,880	0,576	0,816
	Hassasiyet	0,532	0,896	0,598	0,801
	F-ölçütü	0,657	0,888	0,587	0,808
	Doğruluk	0,871	0,883	0,573	0,796
epoch 1000	Kesinlik	0,864	0,866	0,589	0,848
	Hassasiyet	0,890	0,886	0,623	0,825
	F-ölçütü	0,877	0,876	0,606	0,836
	Doğruluk	0,872	0,871	0,596	0,825

Şekil 7.14'de 100, 200, 300, 400 ve 500 olmak üzere farklı doküman vektör boyutlarına, Şekil 7.15'de 1, 2, 3, 4 ve 5 olmak üzere kelimelerin minimum geçme sayılarına ve Şekil 7.16'da 10, 100 ve 1000 olmak üzere eğitim devri sayısına göre elde edilen dört farklı doc2vec model tabanlı ortalama doküman vektörlerin IMDB film yorumları veri seti üzerinde duygu sınıflandırması doğruluk sonuçları verilmiştir. *dbow* ve *dbow_skip* modelleri ile oluşturulan pozitif ve negatif ortalama vektörleri, Türkçe duygu sınıflandırmada olduğu gibi burada da *dm_mean* ve *dm_concat* modellerine göre daha iyi sonuç vermiştir. *dbow_skip* modeli %88,1 doğruluk oranı ile en yüksek sonucu almıştır. %87,85 ile *dbow* modeli Şekil 7.14'de görülen hiper parametre değeri ile ikinci en iyi model olarak gözlemlenmiştir. *dm_mean* ve *dm_concat* modelleri sırasıyla en yüksek %85,4 ve %83,2 doğruluk oranında daha kötü bir sınıflandırma sağlamıştır. *dm_concat* modeli Türkçe metinlerden doküman vektörlerin elde edilmesindense İngilizce metinlerin gösteriminde daha başarılıdır. Fakat dil bağımsız bir model tercih edilmesi istendiğinden bu tutarsız doğruluk oranından dolayı bu modelin kullanılması tavsiye edilmemektedir.



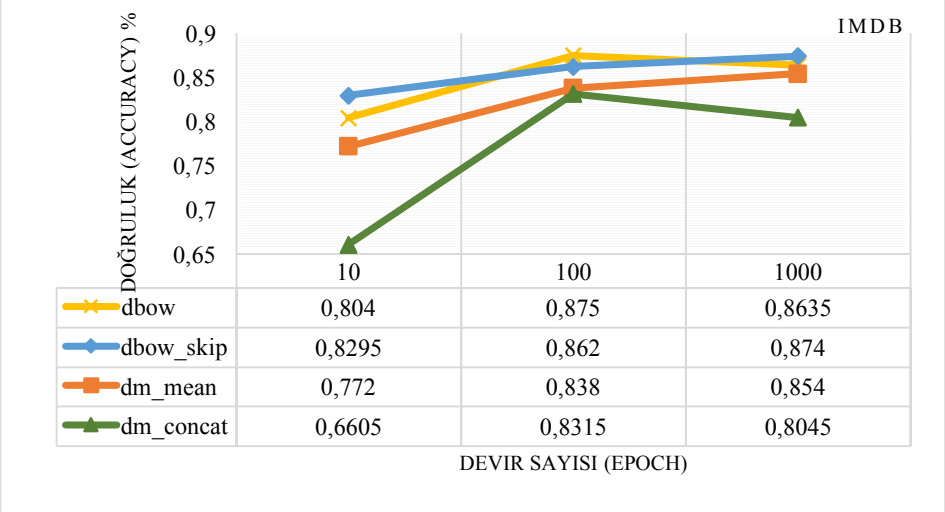
	100	200	300	400	500
dbow	0,8785	0,8645	0,868	0,866	0,8755
dbow_skip	0,8715	0,8645	0,871	0,87	0,866
dm_mean	0,8375	0,837	0,852	0,8475	0,8475
dm_concat	0,832	0,8385	0,838	0,803	0,7945

Şekil 7. 14. Farklı dokuman vektör boyutlarına göre ortalama doküman vektörlerinin duygu sınıflandırması sonuçları (IMDB veri seti)



	1	2	3	4	5
dbow	0,866	0,8715	0,87	0,878	0,875
dbow_skip	0,8655	0,8655	0,872	0,8745	0,881
dm_mean	0,8405	0,842	0,844	0,8415	0,8445
dm_concat	0,8305	0,8305	0,8265	0,824	0,83

Şekil 7. 15. Kelimelerin minimum geçme sayısına göre ortalama doküman vektörlerinin duygu sınıflandırması sonuçları (IMDB veri seti)



Şekil 7. 16. Devir sayısına göre ortalama doküman vektörlerinin duygu sınıflandırması sonuçları (IMDB veri seti)

Önerilen yaklaşım sonucu hesaplanan sınıflandırma performansının IMDB veri seti üzerinde gerçekleştirilen literatürdeki diğer çalışmalar ile karşılaştırılması amacıyla [13]'te sunulan diğer yaklaşımların modele dahil olmayan ayrı test seti üzerindeki sınıflandırma hata oranları Tablo 7.6'da sunulmuştur.

Tablo 7. 6. IMDB veri seti üzerinde çeşitli doküman temsilleri yaklaşımları ile sınıflandırma hata oranları

Model	**Hata oranı (%)**
BOW	12.59
RNN-LM [255]	13.59
DEA [188]	12.54
Word2vec+AVG	12.69
Word2vec+IDF	11.92
Doc2vec	12.10
Doc2vecC [228]	11.70
Skip-thought Vektör [256]	17.42
ADE (doc2vec+AVG)	**11.90**

Tablo 7.6 incelendiğinde, Kelime torbaları (BOW) modeli, Tekrarlı Sinir Ağları tabanlı (RNN-LM) yöntemi (-0,7), word2vec kullanılarak eğitilen kelime vektörlerinin ortalaması word2vec+AVG (-0,8) ve ağırlıklı ortalaması alınan Word2vec+IDF yöntemi (-0.02), dağıtık cümle kodlayıcı Skip-thought vektörleri (-5,52), her bir dokümanı kelimelerin vektörleri olarak düşünen Doc2vecC yöntemi (+0,2), lojistik regresyon sınıflandırıcı ile sınıf tahmini gerçekleştiren Doc2vec yöntemlerine (-0,2) göre önerilen ADE yaklaşımı parantez içinde verilen daha az ya da yaklaşık benzer hata oranları ile iyi bir performans sergilemiştir.

7.11. Sonuçlar

Bu tez çalışmasında, metin sınıflandırma üzerinde çalışabilecek Ortalama Doküman Vektörü (ADE) yaklaşımı sunulmuştur. Önerilen yöntemin performans değerlendirmesi yapmak için İngilizce ve Türkçe veri setleri üzerinde duygu sınıflandırması gerçekleştirmiştir. Tablo 7'de gösterilen diğer yöntem ve yaklaşımlar ile kıyaslandığında ADE yaklaşımı doküman vektörleri ile duygu sınıflandırmada IMDB veri seti için %88,1 ve BEYAZPERDE veri seti için %88,5 doğruluk oranı ile başarılı bir performans sergilemiştir. Her iki dilde de alınan performans sonuçları incelendiğinde dağıtık kelime torbaları modellerinden elde edilen ortalama doküman vektörleri, dağıtık bellek modellerine göre daha iyi çalışmaktadır. Özellikle dağıtık bellek ile hedef kelime tahmininde bağlam kelimelerin birleştirildiği model sonuçları diğer modellere göre çok başarısız kalmıştır. Buna karşılık doğru parametreler seçildiğinde önerilen yaklaşım ile sınıflandırmada, dağıtık kelime torbalarındaki doküman vektörleri ile birlikte kelime vektörlerinin eğitimi en iyi sonucu vermiştir. ADE yaklaşımı doc2vec modelleri kullanarak ilgili sınıfa ait ortalama doküman vektörü elde edilmesi temeline dayanmaktadır. Basit ve hızlı bir çözüm sunan bu yaklaşım, bu tez bölümünün ana katkısını oluşturmak ile birlikte bu bölümün temel katkıları aşağıda maddeler halinde verilmiştir.

1. Türkçe duygu sınıflandırma çalışmalarında doğru ve güvenilir karşılaştırmalar yapmak için erişilebilir bir BEYAZPERDE film yorumları veri setinin sunulması,
2. Türkçe duygu sınıflandırması çalışmalarının geliştirilmesi için Duygu Etiketleme Sistemi geliştirilmesi,

3. Doküman vektörleri (Doc2vec) modelleri kullanarak yeni bir Ortalama Doküman Vektörleri yaklaşımının sunulması,
4. Doc2vec modellerinin hiper parametre ayarlamaları ile geliştirilmesi,
5. Türkçe duygu sınıflandırması çalışmaları için seçilecek doküman vektör modellerinin karşılaştırılması ve performans sonuçlarının sunulması,
6. Önerilen yaklaşımın literatürdeki diğer yöntem ve yaklaşımlar ile doğru bir şekilde kıyaslanabilmesi için IMDB veri seti üzerinde de test edilmesi,
7. Dil bağımsız olarak çalışabilecek bir yaklaşım ile büyük metin verilerinin hızlı ve etkili bir şekilde sınıflandırılması.

8. SONUÇLAR

Bu tez çalışmasında büyük veri teknolojileri ve derin öğrenme mimarileri detaylı bir şekilde analiz edilmiştir, dört temel uygulama ve çeşitli yöntemler sunularak akademik ve endüstriyel alanlara katkı sunulması hedeflenmiştir.

Tez çalışmasının katkıları ve sonuçları aşağıda maddeler halinde sunulmuştur:

1. İlk çalışmada çağrı merkezleri için bulut tabanlı dağıtık performans analizi ve değerlendirme sistemi geliştirilmiştir. Önerilen sistem OpenStack açık kaynaklı bulut bilişim altyapısı üzerine inşa edilmiştir. Bu bulut yapısı üzerindeki büyük verileri analiz etmek için Apache HBase, Mahout, Pig ve ZooKeeper teknolojileri kullanılmıştır. Enterprise Java (JSF, Glassfish, Servlet) teknolojileri ve MongoDB NoSQL veritabanı yardımıyla kullanıcı arayüzleri ve veri depolama sistemleri geliştirilmiştir. Önerilen sistem, iç ve dış çağrı kayıtlarını dağıtık bir şekilde işleyen bulut tabanlı bir performans ölçüm sistemi sunarak müşteri memnuniyeti, satış ve pazarlama, hizmet kalitesi ve performans yönetiminde yüksek bir performans ile önemli bir katkı sağlayacaktır.
2. Büyük veri teknolojileri kullanarak ders kitaplarının okunabilirliğinin analizi için kullanılabilecek Hadoop tabanlı bir dağıtık okunabilirlik sistemi sunulmuştur. Bu sistem için OpenStack bulut bilişim yazılımı kullanılarak 10 düğümden oluşan bir Dağıtık Hadoop kümesi mimarisi oluşturulmuştur. Önerilen sistemin karşılaştırmalı performans sonuçlarına bakıldığında etkili ve başarılı bir şekilde çalıştığı gözlemlenmiştir. Bu sistem ile eğitimciler ve aileler için öğrencilerin okuma seviyelerine uygun okuma materyalleri bulma konusunda yardımcı olunması düşünülmüştür.
3. Çeşitli derin öğrenme modelleri ve hiper parametre optimizasyonları ile Beyazperde film yorumları ve haberler veri seti üzerinde ikili ve çoklu metin sınıflandırma işlemleri gerçekleştirilmiştir. Gerçekleştirilen uygulamalar ile geleneksel sinir ağlarının sabit metin girdi boyutu ve bellek problemi kısıtlamaları ve problemleri çözülmüştür. Büyük metin veri setleri üzerinde önceden eğitilmiş Türkçe kelime temsilleri ile birlikte derin öğrenme modellerinin kullanılması hem çalışma süresini azaltmış hem de performans kazanımı sağlamıştır. Kullanılan derin öğrenme modelleri ile hiper parametre

optimizasyonundan sonra aşırı öğrenme probleminin üstesinden gelinerek duygu sınıflandırmada %98 ve beş kategoride haber sınıflandırmada %94 doğruluk oranlarında, çok başarılı sonuçlar alınmıştır. Türkçe metin sınıflandırma çalışmaları için derin öğrenme modelleri ve önemli parametre seçimleri önerilmiştir. Buna göre tek katmanlı CNN ve CNN-LSTM modelleri yüksek performansı ve düşük zaman maliyeti ile diğer modellerden ön plana çıkmıştır. MLP model olarak adlandırılan çok katmanlı algılayıcılar, diğer tüm derin öğrenme modellerine göre duygu sınıflandırmada çok başarısız kalmıştır. Bu durum derin sinir ağlarının yapay zeka alanında ümit vadeden bir gelişme olduğunu göstermektedir. Dolayısıyla doğal işleme alanında bu tez çalışmasının da yoğunlaşmış olduğu derin öğrenme mimarilerinin tercih edilmesi doğru karar olacaktır.

4. Son olarak metin sınıflandırma üzerinde çalışabilecek Ortalama Doküman Vektörleri (ADE) yöntemi sunulmuştur. ADE yaklaşımı paragraf vektör modelleri kullanarak ilgili sınıfa ait ortalama doküman vektörü elde edilmesi temeline dayanmaktadır. Dil bağımsız olarak çalışabilecek, basit ve hızlı bir çözüm sunan bu yaklaşım ile büyük metin verilerinin hızlı ve etkili bir şekilde sınıflandırılması, bu tez bölümünün ana katkısını oluşturmaktır. Türkçe ve İngilizce duygu sınıflandırması çalışmaları için seçilecek doküman vektör modellerinin karşılaştırılması ve performans sonuçlarına göre önerilen önerilen yöntem ilgili bölümde verilen literatürdeki yaklaşımlara göre başarılı bir sonuç elde etmiştir.
5. Tez çalışması kapsamında akademik alana katkı sunmak amacıyla [252,257–262] referanslı makaleler sunulmuştur. TUBİTAK Girişimcilik Aşamalı Destek Programı kapsamında 2140243 numaralı "Çağrı merkezleri için bulut tabanlı dağıtık performans analizi ve değerlendirme sistemi" başlıklı proje başarıyla tamamlanmıştır.

KAYNAKLAR

[1] **M.M. Najafabadi, F. Villanustre, T.M. Khoshgoftaar, N. Seliya, R. Wald, E. Muharemagic**, 2015, Deep learning applications and challenges in big data analytics, *J. Big Data*. 2 1.

[2] **P. Russom, Others**, 2011, Big data analytics, *TDWI Best Pract. Report, Fourth Quart.* 19 1–34.

[3] **G.E. Hinton, S. Osindero, Y.-W. Teh**, 2006, A fast learning algorithm for deep belief nets, *Neural Comput.* 18 1527–1554.

[4] **Y. Bengio**, 2013, Deep learning of representations: Looking forward, *Int. Conf. Stat. Lang. Speech Process.* 1–37.

[5] **C.C. Aggarwal, C. Zhai**, 2012, Mining text data, *Springer Science & Business Media.*,

[6] **C. Dunne, B. Shneiderman, R. Gove, J. Klavans, B. Dorr**, 2012, Rapid understanding of scientific paper collections: Integrating statistics, text analytics, and visualization, *J. Assoc. Inf. Sci. Technol.* 63 2351–2369.

[7] **A. Grob-Klubmann, N. Hautsch**, 2009, When Machines Read the News: Using Automated Text Analytics to Quantify High Frequency News Impacts.

[8] **G. Miner, J. Elder IV, T. Hill**, 2012, Practical text mining and statistical analysis for non-structured text data applications, *Academic Press.*,

[9] **X. Zhang, Y. LeCun**, 2015, Text understanding from scratch, *ArXiv Prepr. ArXiv1502.01710*.

[10] **X. Zhang, J. Zhao, Y. LeCun**, 2015, Character-level convolutional networks for text classification *Adv. Neural Inf. Process. Syst.* 649–657.

[11] **Y. LeCun, B.E. Boser, J.S. Denker, D. Henderson, R.E. Howard, W.E. Hubbard, L.D. Jackel**, 1990, Handwritten digit recognition with a back-propagation network, *Adv. Neural Inf. Process. Syst.* 396–404.

[12] **Y. LeCun, L. Bottou, Y. Bengio, P. Haffner**, 1998, Gradient-based learning applied to document recognition, *Proc. IEEE*. 86 2278–2324.

[13] **G.E. Hinton**, 1984, Distributed representations.

[14] **S. Roukos, J. Quin, T. Ward**, 2014, Multi-lingual Text Leveling, *Int. Conf. Text,*

Speech, Dialogue. 19–26.

[15] **T. Mikolov, A. Deoras, S. Kombrink, L. Burget, J. Černocky**, 2011, Empirical evaluation and combination of advanced language modeling techniques, *Twelfth Annu. Conf. Int. Speech Commun. Assoc.*

[16] **T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean**, 2013, Distributed representations of words and phrases and their compositionality *Adv. Neural Inf. Process. Syst.* 3111–3119.

[17] **L. Deng, D. Yu, Others**, 2014, Deep learning: methods and applications, *Found. Trends®in Signal Process.* 7 197–387.

[18] **G.E. Hinton**, 2007, Learning multiple layers of representation, *Trends Cogn. Sci.* 11 428–434.

[19] **G.E. Hinton, R.R. Salakhutdinov**, 2006, Reducing the dimensionality of data with neural networks, *Science (80-.).* 313 504–507.

[20] **Z. Cao, F. Wei, L. Dong, S. Li, M. Zhou**, 2015, Ranking with Recursive Neural Networks and Its Application to Multi-Document Summarization. *AAAI*. 2153–2159.

[21] **M. Denil, A. Demiraj, N. de Freitas**, 2014, Extraction of salient sentences from labelled documents, *ArXiv Prepr. ArXiv1412.6815*.

[22] **S. Zhong, Y. Liu, B. Li, J. Long**, 2015, Query-oriented unsupervised multi-document summarization via deep learning model, *Expert Syst. Appl.* 42 8146–8155.

[23] **F. Wei, W. Li, Q. Lu, Y. He**, 2010, A document-sensitive graph model for multi-document summarization, *Knowl. Inf. Syst.* 22 245–259.

[24] **Y. Ouyang, W. Li, S. Li, Q. Lu**, 2011, Applying regression models to query-focused multi-document summarization, *Inf. Process. Manag.* 47 227–237.

[25] **S. Kombrink, T. Mikolov, M. Karafiát, L. Burget**, 2011, Recurrent neural network based language modeling in meeting recognition, *Twelfth Annu. Conf. Int. Speech Commun. Assoc.*

[26] **L. Deng**, 2012, Three classes of deep learning architectures and their applications: a tutorial survey, *APSIPA Trans. Signal Inf. Process.*

[27] **P. Latouche**, 2007, Distributed machine learning algorithms.

[28] **D. Laney**, 2001, {3D} Data Management: Controlling Data Volume, Velocity, and Variety, http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

[29] **J. Dean, S. Ghemawat**, 2008, MapReduce: simplified data processing on large clusters, *Commun. ACM*. 51 107–113.

[30] **T. White**, 2012, Hadoop: The definitive guide, *O'Reilly Media, Inc.*

[31] **D. Borthakur, Others**, 2008, HDFS architecture guide, *Hadoop Apache Proj.* 53.

[32] **A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy**, 2009, Hive: a warehousing solution over a map-reduce framework, *Proc. VLDB Endow.* 2 1626–1629.

[33] **S. Hoffman**, 2013, Apache Flume: distributed log collection for Hadoop, *Packt Publishing Ltd.*

[34] **M.K. Islam, A. Srinivasan**, 2015, Apache Oozie: The Workflow Scheduler for Hadoop, *O'Reilly Media, Inc.*

[35] **C. Olston, B. Reed, U. Srivastava, R. Kumar, A. Tomkins**, 2008, Pig latin: a not-so-foreign language for data processing, *Proc. 2008 ACM SIGMOD Int. Conf. Manag. Data*. 1099–1110.

[36] **Y. Bu, B. Howe, M. Balazinska, M.D. Ernst**, 2010, HaLoop: Efficient iterative data processing on large clusters, *Proc. VLDB Endow.* 3 285–296.

[37] **J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, G. Fox**, 2010, Twister: a runtime for iterative mapreduce, *Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput.* 810–818.

[38] **S. Owen**, 2012, Mahout in action.

[39] **N.M. Nasrabadi**, 2007, Pattern recognition and machine learning, *J. Electron. Imaging*. 16 49901.

[40] **T.M. Mitchell**, 1999, Machine learning and data mining, *Commun. ACM*. 42 30–36.

[41] **O. Bousquet, L. Bottou**, 2008, The tradeoffs of large scale learning, *Adv. Neural Inf. Process. Syst.* 161–168.

[42] **S. Sonnenburg, G. Rätsch, K. Rieck**, 2007, Large scale learning with string kernels.

[43] **K. Kumar, C. Bhattacharya, R. Hariharan**, 2008, A randomized algorithm for large scale support vector learning, *Adv. Neural Inf. Process. Syst.* 793–800.

[44] **K.M. Svore, C.J. Burges**, 2011, Large-scale learning to rank using boosted decision trees, *Scaling Up Mach. Learn. Parallel Distrib. Approaches*. 2.

[45] **D. Peteiro-Barral, B. Guijarro-Berdiñas**, 2013, A survey of methods for distributed machine learning, *Prog. Artif. Intell.* 2 1–11.

[46] **K. Li, C. Gibson, D. Ho, Q. Zhou, J. Kim, O. Buhisi, D.E. Brown, M. Gerber**, 2013, Assessment of machine learning algorithms in cloud computing frameworks, *Syst. Inf. Eng. Des. Symp. (SIEDS), 2013 IEEE*. 98–103.

[47] **E.R. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. Gonzalez, M.J. Franklin, M.I. Jordan, T. Kraska**, 2013, MLI: An API for distributed machine learning, *Data Min. (ICDM), 2013 IEEE 13th Int. Conf.* 1187–1192.

[48] **J. Erickson**, 2009, Database Technologies: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications, *IGI Global*.

[49] **A. Jacobs**, 2009, The pathologies of big data, *Commun. ACM*. 52 36–44.

[50] **B. Liu, S.-G. Cao, Q.-C. Li, Q. Li**, 2011, A hierarchical distributed data mining architecture, *Mach. Learn. Cybern. (ICMLC), 2011 Int. Conf.* 1 40–44.

[51] **X. Wu, X. Zhu, G.-Q. Wu, W. Ding**, 2014, Data mining with big data, *IEEE Trans. Knowl. Data Eng.* 26 97–107.

[52] **N. Ye, Others**, 2003, The handbook of data mining, *Lawrence Erlbaum Associates, Publishers Mahwah, NJ/London*.

[53] **Y. Guo, J. Sutiwaraphun**, 1999, Probing knowledge in distributed data mining, in: Pacific-Asia Conf. Knowl. Discov. Data Min., : p. 443–452.

[54] **M. Hai, S. Zhang, L. Zhu, Y. Wang**, 2012, A survey of distributed clustering algorithms, *Ind. Control Electron. Eng. (ICICEE), 2012 Int. Conf.* 1142–1145.

[55] **N. Zhang, H. Bao**, 2009, Research on distributed data mining technology based on Grid, *Database Technol. Appl. 2009 First Int. Work.* 440–443.

[56] **K. Ericson, S. Pallickara**, 2013, On the performance of high dimensional data clustering and classification algorithms, *Futur. Gener. Comput. Syst.* 29 1024–1034.

[57] **S. Pallickara, J. Ekanayake, G. Fox**, 2009, Granules: A lightweight, streaming runtime for cloud computing with support, for map-reduce, *Clust. Comput. Work. 2009. Clust. IEEE Int. Conf.* 1–10.

[58] **R.M. Esteves, R. Pais, C. Rong**, 2011, K-means clustering in the cloud--a Mahout test, in: Adv. Inf. Netw. Appl. (WAINA), 2011 IEEE Work. Int. Conf., : p. 514–519.

[59] **P.M. de Avila, R.S. da Silva, L.A.V. Francisco, R.P. Pantoni, D. Buzatto, S.D. Zorzo**, 2014, Comparing K-means and mean shift algorithms performance using Mahout in a private cloud environment, *J. Commun. Comput.* 11 45–51.

[60] **M. Steinbach, G. Karypis, V. Kumar, Others**, 2000, A comparison of document clustering techniques, *KDD Work. Text Min.* 400 525–526.

[61] **J. Wan, W. Yu, X. Xu**, 2009, Design and implement of distributed document clustering based on MapReduce, *Proc. Second Symp. Int. Comput. Sci. Comput. Technol. (ISCSCT), Huangshan, PR China*. 278–280.

[62] **D. Shrestha**, 2009, Text mining with Lucene and Hadoop: Document clustering with feature extraction.

[63] **Y.K. Patil, V.S. Nandedkar**, 2014, Hadoop: A new approach for document clustering, *Int. J. Adv. Res. IT Eng*. 3 1–8.

[64] **C. Carpineto, S. Osiński, G. Romano, D. Weiss**, 2009, A survey of web clustering engines, *ACM Comput. Surv.* 41 17.

[65] **Y. Liu, X. Wang, Z. Xu, Y. Guan**, 2006, A Survey of Document Clustering [J], *J. Chinese Inf. Process.* 3 8.

[66] **C. Rong, Others**, 2011, Using Mahout for clustering Wikipedia's latest articles: A comparison between k-means and fuzzy c-means in the cloud *Cloud Comput. Technol. Sci. (CloudCom), 2011 IEEE Third Int. Conf.* 565–569.

[67] **T.F. Gharib, M.M. Fouad, M.M. Aref**, 2010, Fuzzy document clustering approach using WordNet lexical categories, in: Adv. Tech. Comput. Sci. Softw. Eng., *Springer*, : p. 181–186.

[68] **T.T. Win, L. Mon**, 2010, Document clustering by fuzzy c-mean algorithm, in: Adv. Comput. Control (ICACC), 2010 2nd Int. Conf., : p. 239–242.

[69] **F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, Others**, 2011, Scikit-learn: Machine learning in Python, *J. Mach. Learn. Res.* 12 2825–2830.

[70] **G. Hackeling**, 2014, Mastering Machine Learning with scikit-learn, *Packt Publishing Ltd.*

[71] scikit-learn. http://scikit-learn.org (En Son Ziyaret-accessed 13/05/2018)

[72] **E. Bressert**, 2012, SciPy and NumPy: an overview for developers, *" O'Reilly Media, Inc.,"* .

[73] **A. Jovic, K. Brkic, N. Bogunovic**, 2014, An overview of free software tools for general data mining, *Inf. Commun. Technol. Electron. Microelectron. (MIPRO), 2014 37th Int. Conv.* 1112–1117.

[74] **M. Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, Others**, 2016, Apache spark: a unified engine for big data processing, *Commun. ACM*. 59 56–65.

[75] **N. Pentreath**, 2015, Machine Learning with Spark, *Packt Publishing Ltd*.

[76] **M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica**, 2012, Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, *Proc. 9th USENIX Conf. Networked Syst. Des. Implement.* 2.

[77] **M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, I. Stoica**, 2010, Spark: Cluster computing with working sets, *HotCloud*. 10 95.

[78] **R.S. Xin, J. Rosen, M. Zaharia, M.J. Franklin, S. Shenker, I. Stoica**, 2013, Shark: SQL and rich analytics at scale, *Proc. 2013 ACM SIGMOD Int. Conf. Manag. Data*. 13–24.

[79] **M. Armbrust, R.S. Xin, C. Lian, Y. Huai, D. Liu, J.K. Bradley, X. Meng, T. Kaftan, M.J. Franklin, A. Ghodsi, Others**, 2015, Spark sql: Relational data processing in spark, in: Proc. 2015 ACM SIGMOD Int. Conf. Manag. Data, : p. 1383–1394.

[80] **X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D.B. Tsai, M. Amde, S. Owen, Others**, 2016, Mllib: Machine learning in apache spark,

J. Mach. Learn. Res. 17 1235–1241.

[81] **R.S. Xin, J.E. Gonzalez, M.J. Franklin, I. Stoica**, 2013, Graphx: A resilient distributed graph system on spark, in: First Int. Work. Graph Data Manag. Exp. Syst., : p. 2.

[82] **F. Bredmar, E. Andersson, E. Bogren**, 2014, Scalable Machine Learning for Big Data.

[83] **M. Zaharia, T. Das, H. Li, S. Shenker, I. Stoica**, 2012, Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters, *HotCloud*. 12 10.

[84] **S. Shahrivari**, 2014, Beyond batch processing: towards real-time and streaming big data, *Computers*. 3 117–129.

[85] **H. Wang, B. Wu, S. Yang, B. Wang, Y. Liu**, 2014, Research of decision tree on yarn using mapreduce and spark, *Proc. 2014 World Congr. Comput. Sci. Comput. Eng. Appl. Comput.* 21–24.

[86] **D. Lawson**, 2014, Alternating direction method of multipliers implementation using Apache Spark.

[87] **C.-Y. Lin, C.-H. Tsai, C.-P. Lee, C.-J. Lin**, 2014, Large-scale logistic regression and linear support vector machines using spark, *Big Data (Big Data), 2014 IEEE Int. Conf.* 519–528.

[88] **F. Liang, C. Feng, X. Lu, Z. Xu**, 2014, Performance benefits of DataMPI: a case study with BigDataBench, in: Work. Big Data Benchmarks, Perform. Optim. Emerg. Hardw., : p. 111–123.

[89] **H. Qiu, R. Gu, C. Yuan, Y. Huang**, 2014, Yafim: a parallel frequent itemset mining algorithm with spark, *Parallel Distrib. Process. Symp. Work. (IPDPSW), 2014 IEEE Int.* 1664–1671.

[90] **Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin**, 2011, Graphlab: A distributed framework for machine learning in the cloud, *ArXiv Prepr. ArXiv1107.0922*.

[91] **Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, J.M. Hellerstein**, 2012, Distributed GraphLab: a framework for machine learning and data mining in the cloud, *Proc. VLDB Endow.* 5 716–727.

[92] **Y. Low, J.E. Gonzalez, A. Kyrola, D. Bickson, C.E. Guestrin, J. Hellerstein**, 2014,

Graphlab: A new framework for parallel machine learning, *ArXiv Prepr. ArXiv1408.2041*.

[93] **G. Malewicz, M.H. Austern, A.J.C. Bik, J.C. Dehnert, I. Horn, N. Leiser, G. Czajkowski**, 2010, Pregel: a system for large-scale graph processing, *Proc. 2010 ACM SIGMOD Int. Conf. Manag. Data*. 135–146.

[94] **Y. Zhang, Q. Gao, L. Gao, C. Wang**, 2012, Accelerate large-scale iterative computation through asynchronous accumulative updates, in: Proc. 3rd Work. Sci. Cloud Comput., : p. 13–22.

[95] **G. Chakraborty, M. Pagolu, S. Garla**, 2014, Text mining and analysis: practical methods, examples, and case studies using SAS, *SAS Institute.*

[96] **A. Kaklauskas**, 2015, Biometric and Intelligent Decision Making Support, *Springer.*,

[97] **A. Katal, M. Wazid, R.H. Goudar**, 2013, Big data: issues, challenges, tools and good practices, *Contemp. Comput. (IC3), 2013 Sixth Int. Conf.* 404–409.

[98] **S. LaValle, E. Lesser, R. Shockley, M.S. Hopkins, N. Kruschwitz**, 2011, Big data, analytics and the path from insights to value, *MIT Sloan Manag. Rev.* 52 21.

[99] **H. Takeuchi, L.V. Subramaniam, T. Nasukawa, S. Roy**, 2007, Automatic identification of important segments and expressions for mining of business-oriented conversations at contact centers, *Proc. 2007 Jt. Conf. Empir. Methods Nat. Lang. Process. Comput. Nat. Lang. Learn.*

[100] **G. Mishne, D. Carmel, R. Hoory, A. Roytman, A. Soffer**, 2005, Automatic analysis of call-center conversations, *Proc. 14th ACM Int. Conf. Inf. Knowl. Manag.* 453–459.

[101] **M. Garnier-Rizet, G. Adda, F. Cailliau, J.-L. Gauvain, S. Guillemin-Lanne, L. Lamel, S. Vanni, C. Waast-Richard, Others**, 2008, CallSurf: Automatic Transcription, Indexing and Structuration of Call Center Conversational Speech for Knowledge Extraction and Query by Content. *LREC.*

[102] **V. Pallotta, R. Delmonte, L. Vrieling, D. Walker**, 2011, Interaction Mining: The new frontier of Call Center Analytics *CEUR Workshop Proc.* 771.

[103] **S.K. Kopparapu**, 2015, Non-Linguistic Analysis of Call Center Conversations, *Springer.*

[104] **T. Gunarathne, B. Zhang, T.-L. Wu, J. Qiu**, 2013, Scalable parallel computing on clouds using Twister4Azure iterative MapReduce, *Futur. Gener. Comput. Syst.* 29 1035–1048.

[105] **H. Yang, A. Dasdan, R.-L. Hsiao, D.S. Parker**, 2007, Map-reduce-merge: simplified relational data processing on large clusters, *Proc. 2007 ACM SIGMOD Int. Conf. Manag. Data*. 1029–1040.

[106] **A.B.M. Moniruzzaman, S.A. Hossain**, 2013, Nosql database: New era of databases for big data analytics-classification, characteristics and comparison, *ArXiv Prepr. ArXiv1307.0191*.

[107] **N. Leavitt**, 2010, Will NoSQL databases live up to their promise?, *Computer (Long. Beach. Calif)*. 43.

[108] **A. Huang**, 2008, Similarity measures for text document clustering, *Proc. Sixth New Zeal. Comput. Sci. Res. Student Conf. (NZCSRSC2008), Christchurch, New Zeal.* 49–56.

[109] **C.K. Kent, N. Salim**, 2010, Features based text similarity detection, *ArXiv Prepr. ArXiv1001.3487*.

[110] **W.-T. Yih, C. Meek**, 2007, Improving similarity measures for short segments of text, in: AAAI, : p. 1489–1494.

[111] **W.H. Gomaa, A.A. Fahmy**, 2013, A survey of text similarity approaches, *Int. J. Comput. Appl.* 68.

[112] **R. Mihalcea, C. Corley, C. Strapparava, Others**, 2006, Corpus-based and knowledge-based measures of text semantic similarity, *AAAI*. 6 775–780.

[113] **M. Fedoryszak, D. Tkaczyk, Ł. Bolikowski**, 2013, Large scale citation matching using Apache Hadoop, *Int. Conf. Theory Pract. Digit. Libr.* 362–365.

[114] **P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, V. Vyas**, 2009, Web-scale distributional similarity and entity set expansion, in: Proc. 2009 Conf. Empir. Methods Nat. Lang. Process. Vol. 2-Volume 2, : p. 938–947.

[115] **A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, Others**, 2014, Deep speech: Scaling up end-to-end speech recognition, *ArXiv Prepr. ArXiv1412.5567*.

[116] **A.A. Akin, M.D. Akin**, 2007, Zemberek, an open source nlp framework for turkic languages, *Structure*. 10 1–5.

[117] **Y. Sun, X. Wang, Y. Yu**, 2014, Readability to Financial Report: A Comparative Study of Chinese and Foreign Countries, *Comput. Sci. Optim. (CSO), 2014 Seventh Int. Jt. Conf.* 223–227.

[118] **G.K. Berland, M.N. Elliott, L.S. Morales, J.I. Algazy, R.L. Kravitz, M.S. Broder, D.E. Kanouse, J.A. Muñoz, J.-A. Puyol, M. Lara, Others**, 2001, Health information on the Internet: accessibility, quality, and readability in English and Spanish, *Jama*. 285 2612–2621.

[119] **E.K.F. Leong, M.T. Ewing, L.F. Pitt**, 2002, E-comprehension: Evaluating B2B websites using readability formulae, *Ind. Mark. Manag.* 31 125–131.

[120] **R. Flesch**, 1948, A new readability yardstick, *J. Appl. Psychol.* 32 221.

[121] **J.P. Kincaid, R.P. Fishburne Jr, R.L. Rogers, B.S. Chissom**, 1975, Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.

[122] **G.H. Mc Laughlin**, 1969, SMOG grading-a new readability formula, *J. Read.* 12 639–646.

[123] **R. Gunning**, 1952, The technique of clear writing.

[124] **R.J. Senter, E.A. Smith**, 1967, Automated readability index.

[125] **J.S. Chall, E. Dale**, 1995, Readability revisited: The new Dale-Chall readability formula, *Brookline Books*.

[126] **J.M. Fletcher**, 2006, Measuring reading comprehension, *Sci. Stud. Read.* 10 323–330.

[127] **L.A. Sherman**, 1893, Analytics of literature: A manual for the objective study of English prose and poetry, *Ginn*.

[128] **B.A. Lively, S.L. Pressey**, 1923, A method for measuring the" vocabulary burden" of textbooks.

[129] **Y.-S. Lee, H.-C. Tseng, J.-L. Chen, C.-Y. Peng, T.-H. Chang, Y.-T. Sung**, 2012, Constructing a novel Chinese readability classification model using principal component analysis and genetic programming, *Adv. Learn. Technol. (ICALT), 2012 IEEE 12th Int.*

Conf. 164–166.

[130] **W.-T. Kuo, C.-S. Huang, C.-L. Liu**, 2010, Using Linguistic Features to Predict Readability of Short Essays for Senior High School Students in Taiwan, *Int. J. Comput. Linguist. Chinese Lang. Process. Vol. 15, Number 3-4, Sept. 2010*. 15.

[131] **B. Broda, B. Niton, W. Gruszczynski, M. Ogrodniczuk**, 2014, Measuring Readability of Polish Texts: Baseline Experiments, *LREC*. 573–580.

[132] **P. Daowadung, Y.-H. Chen**, 2011, Using word segmentation and SVM to assess readability of Thai text for primary school students, *Comput. Sci. Softw. Eng. (JCSSE), 2011 Eighth Int. Jt. Conf.* 170–174.

[133] **Y.-H. Chen, Y.-H. Tsai, Y.-T. Chen**, 2011, Chinese readability assessment using TF-IDF and SVM, *Mach. Learn. Cybern. (ICMLC), 2011 Int. Conf.* 2 705–710.

[134] **T. François, C. Fairon**, 2012, An AI readability formula for French as a foreign language, *Proc. 2012 Jt. Conf. Empir. Methods Nat. Lang. Process. Comput. Nat. Lang. Learn.* 466–477.

[135] **S. Aluisio, L. Specia, C. Gasperin, C. Scarton**, 2010, Readability assessment for text simplification *Proc. NAACL HLT 2010 Fifth Work. Innov. Use NLP Build. Educ. Appl.* 1–9.

[136] **S. Sato, S. Matsuyoshi, Y. Kondoh**, 2008, Automatic Assessment of Japanese Text Readability Based on a Textbook Corpus, *LREC*.

[137] **J. Anderson**, 1981, Analysing the Readability of English and Non-English Texts in the Classroom with Lix.

[138] **J. Sjöholm**, 2012, Probability as readability: A new machine learning approach to readability assessment for written Swedish.

[139] **N. Mat Daud, H. Hassan, N. Abdul Aziz**, 2013, A corpus-based readability formula for estimate of Arabic texts reading difficulty, *World Appl. Sci. J.* 21 168–173.

[140] **H.S. Al-Khalifa, A.A. Al-Ajlan**, 2010, Automatic readability measurements of the Arabic text: An exploratory study, *Arab. J. Sci. Eng.* 35 103–124.

[141] **E. Atesman**, 1997, Measuring readability in Turkish, *AU Tömer Lang. J.* 58 171–174.

[142] **E. (Lüle) MERT**, 2013, The Readability of the Texts in the Turkish Textbooks in

Turkey, *Mersin Univ. J. Fac. Educ.* 3 87–98.

[143] **A. OKUR, G. ARI**, 2013, Readability of Texts Turkish Textbooks in Grades 6, 7, 8., *Ilkogr. Online*. 12.

[144] **A.Z. Guven**, 2014, Readability of Texts in Textbooks in Teaching Turkish to Foreigners, *Anthropol.* 18 513–522.

[145] **W.H. DuBay**, 2004, The Principles of Readability, *Online Submiss.*

[146] **G. Hargis**, 2000, Readability and computer documentation, *ACM J. Comput. Doc.* 24 122–131.

[147] **J. Flood**, 1984, Understanding Reading Comprehension: Cognition, Language, and the Structure of Prose., *ERIC.*

[148] **M. Milone**, 2009, The development of ATOS: The Renaissance readability formula, *Renaissance Learning, Incorporated.*,

[149] **L.A.U.T. Pang**, 2006, Chinese readability analysis and its applications on the internet.

[150] **T.H. Hong, C.H. Yun, J.W. Park, H.G. Lee, H.S. Jung, Y.W. Lee**, 2013, Big data processing with MapReduce for E-book, *Int. J. Multimed. Ubiquitous Eng.* 8 151–162.

[151] Apache Tika. https://tika.apache.org/ (En Son Ziyaret-accessed 12/05/2018).

[152] **B. Whitby**, 2009, Artificial intelligence, *The Rosen Publishing Group*.

[153] **S. Cajal**, 1889, Nuevas aplicaciones del metodo de coloracion de Golgi. Sobre la red nerviosa ganglionar de las vellosidades intestinales *Gac. Med. Catal.* 12 613–616.

[154] **A.C. Guyton**, 1987, Basic neuroscience: anatomy and physiology.

[155] **M. Mayford, S.A. Siegelbaum, E.R. Kandel**, 2012, Synapses and memory storage *Cold Spring Harb. Perspect. Biol.* 4.

[156] **W.S. McCulloch, W. Pitts**, 1943, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.* 5 115–133.

[157] **F. Rosenblatt**, 1957, The perceptron, a perceiving and recognizing automaton Project Para, *Cornell Aeronautical Laboratory*.

[158] **E. Cambria, B. White**, 2014, Jumping NLP curves: A review of natural language processing research, *IEEE Comput. Intell. Mag.* 9 48–57.

[159] **Q. Le, T. Mikolov**, 2014, Distributed representations of sentences and documents, *Int.*

Conf. Mach. Learn. 1188–1196.

[160] **C.D. Manning, H. Schütze**, 1999, Foundations of statistical natural language processing, *MIT press.*

[161] **R. Socher**, 2014, Recursive deep learning for natural language processing and computer vision, Citeseer.

[162] **R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa**, 2011, Natural language processing (almost) from scratch, *J. Mach. Learn. Res.* 12 2493–2537.

[163] **A.M. Turing**, 2009, Computing machinery and intelligence, in: Parsing Turing Test, *Springer*, : p. 23–65.

[164] **B. Widrow, M.E. Hoff**, 1960, Adaptive switching circuits.

[165] **M. Minsky, S.A. Papert, L. Bottou**, 2017, Perceptrons: An introduction to computational geometry, *MIT press*.

[166] **P.J. Werbos**, 1994, The roots of backpropagation: from ordered derivatives to neural networks and political forecasting, *John Wiley & Sons*.

[167] **K. Fukushima, S. Miyake**, 1982, Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition, in: Compet. Coop. Neural Nets, *Springer*, : p. 267–285.

[168] **J.J. Hopfield**, 1982, Neural networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci.* 79 2554–2558.

[169] **D.H. Ackley, G.E. Hinton, T.J. Sejnowski**, 1985, A learning algorithm for Boltzmann machines, *Cogn. Sci.* 9 147–169.

[170] **D.E. Rumelhart, G.E. Hinton, R.J. Williams**, 1986, Learning representations by back-propagating errors, *Nature*. 323 533.

[171] **P. Smolensky**, 1986, Information processing in dynamical systems: Foundations of harmony theory.

[172] **J.L. Elman**, 1990, Finding structure in time, *Cogn. Sci.* 14 179–211.

[173] **M. Schuster, K.K. Paliwal**, 1997, Bidirectional recurrent neural networks, *IEEE Trans. Signal Process.* 45 2673–2681.

[174] **S. Hochreiter, J. Schmidhuber**, 1997, Long short-term memory *Neural Comput.* 9

1735–1780.

[175] **R. Salakhutdinov, G. Hinton**, 2009, Deep Boltzmann Machines, *Proc. Int. Conf. Artif. Intell. Stat.* 5 448–455.

[176] **G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov**, 2012, Improving neural networks by preventing co-adaptation of feature detectors, *ArXiv Prepr. ArXiv1207.0580*.

[177] **I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio**, 2014, Generative adversarial nets, *Adv. Neural Inf. Process. Syst.* 2672–2680.

[178] **S. Sabour, N. Frosst, G.E. Hinton**, 2017, Dynamic routing between capsules, in: Adv. Neural Inf. Process. Syst., : p. 3859–3869.

[179] **A. Krizhevsky, I. Sutskever, G.E. Hinton**, 2012, Imagenet classification with deep convolutional neural networks, *Adv. Neural Inf. Process. Syst.* 1097–1105.

[180] **C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, others**, 2015, Going deeper with convolutions.

[181] **K. Simonyan, A. Zisserman**, 2014, Very deep convolutional networks for large-scale image recognition *ArXiv Prepr. ArXiv1409.1556*.

[182] **R. Girshick, J. Donahue, T. Darrell, J. Malik**, 2014, Rich feature hierarchies for accurate object detection and semantic segmentation, *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 580–587.

[183] **K. He, X. Zhang, S. Ren, J. Sun**, 2016, Deep residual learning for image recognition, *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 770–778.

[184] **M.D. Zeiler, R. Fergus**, 2014, Visualizing and understanding convolutional networks, *Eur. Conf. Comput. Vis.* 818–833.

[185] **K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio**, 2014, Learning phrase representations using RNN encoder-decoder for statistical machine translation, *ArXiv Prepr. ArXiv1406.1078*.

[186] **A. Graves, J. Schmidhuber**, 2005, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, *Neural Networks*. 18 602–610.

[187] **C. Poultney, S. Chopra, Y.L. Cun, others**, 2007, Efficient learning of sparse representations with an energy-based model, in: Adv. Neural Inf. Process. Syst., : pp. 1137–1144.

[188] **P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol**, 2008, Extracting and composing robust features with denoising autoencoders, *Proc. 25th Int. Conf. Mach. Learn.* 1096–1103.

[189] **D.P. Kingma, M. Welling**, 2013, Auto-encoding variational bayes *ArXiv Prepr. ArXiv1312.6114*.

[190] **By Mike Bernico**, 2018, Deep Learning Quick Reference, *Publishing Ltd.*

[191] **D.H. Hubel, T.N. Wiesel**, 1962, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, *J. Physiol.* 160 106–154.

[192] **A. Ivakhnenko**, n.d., Cybernetic predicting devices.

[193] **A.G. Ivakhnenko**, 1971, Polynomial theory of complex systems, *IEEE Trans. Syst. Man. Cybern.* 364–378.

[194] **O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, others**, 2015, Imagenet large scale visual recognition challenge, *Int. J. Comput. Vis.* 115 211–252.

[195] **A. Karpathy**, 2015, The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[196] **P.J. Werbos**, 1990, Backpropagation through time: what it does and how to do it, *Proc. IEEE*. 78 1550–1560.

[197] **Y. Bengio, P. Simard, P. Frasconi**, 1994, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. Neural Networks*. 5 157–166.

[198] **W. James**, 2013, The principles of psychology, *Read Books Ltd.*

[199] **T. Dettmers**, 2016, Deep Learning in a Nutshell: Sequence Learning, *NVIDIA Dev. Blog*. https://devblogs.nvidia.com/deep-learning-nutshell-sequence-learning/ (En Son Ziyaret-accessed 10/04/2018)

[200] **Z.C. Lipton, J. Berkowitz, C. Elkan**, 2015, A critical review of recurrent neural networks for sequence learning, *ArXiv Prepr. ArXiv1506.00019*.

[201] **Udacity Deep Learning Team**, 2018, Long Short-Term Memory Networks. https://www.udacity.com/course/deep-learning-nanodegree--nd101, (En Son Ziyaret-accessed 12/03/2018)

[202] **S. Geman, E. Bienenstock, R. Doursat**, 1992, Neural networks and the bias/variance dilemma *Neural Comput.* 4 1–58.

[203] **D.-A. Clevert, T. Unterthiner, S. Hochreiter**, 2015, Fast and accurate deep network learning by exponential linear units (elus), *ArXiv Prepr. ArXiv1511.07289*.

[204] **K. He, X. Zhang, S. Ren, J. Sun**, 2015, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proc. IEEE Int. Conf. Comput. Vis., : p. 1026–1034.

[205] **D. Mishkin, N. Sergievskiy, J. Matas**, 2016, Systematic evaluation of CNN advances on the ImageNet *ArXiv Prepr. ArXiv1606.02228*.

[206] **D.R. Wilson, T.R. Martinez**, 2003, The general inefficiency of batch training for gradient descent learning *Neural Networks*. 16 1429–1451.

[207] **Y.B. and A.C. Ian Goodfellow**, 2016, Deep Learning, *MIT Press*. 110–120. http://www.deeplearningbook.org/contents/ml.html.

[208] **B.A. Karakuş**, n.d., Google Colaboratory. http://buyukveri.firat.edu.tr/2018/03/05/google-colaboratory-icin-google-drive-uzerinden-dosya-yukleme/ (En Son Ziyaret-accessed 10/05/2018).

[209] **D.P. Kingma, J. Ba**, 2014, Adam: A method for stochastic optimization *ArXiv Prepr. ArXiv1412.6980*.

[210] **S. Shi, Q. Wang, P. Xu, X. Chu**, 2016, Benchmarking state-of-the-art deep learning software tools, *Cloud Comput. Big Data (CCBD), 2016 7th Int. Conf.* 99–104.

[211] **Y. Sugomori**, 2016, Java Deep Learning Essentials, *Packt Publishing Ltd*.

[212] **B. Liu**, 2010, Sentiment Analysis and Subjectivity. *Handb. Nat. Lang. Process.* 2 627–666.

[213] **H. Cui, V. Mittal, M. Datar**, 2006, Comparative experiments on sentiment classification for online product reviews, in: AAAI, : pp. 1265–1270.

[214] **Y. Qu, J. Shanahan, J. Wiebe**, 2004, Exploring attitude and affect in text: Theories and

applications, in: AAAI Spring Symp. Tech. Rep. SS-04-07. AAAI Press. Menlo Park. CA.

[215] **R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A. Ng, C. Potts**, 2013, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proc. 2013 Conf. Empir. Methods Nat. Lang. Process., : pp. 1631–1642.

[216] **D. Sarkar, R. Bali, T. Sharma**, 2018, Analyzing Movie Reviews Sentiment, in: Pract. Mach. Learn. with Python, *Springer*, : p. 331–372.

[217] **J. Pennington, R. Socher, C.D. Manning**, 2014, Glove: Global Vectors for Word Representation., in: EMNLP, : pp. 1532–1543.

[218] **P. Kalaivani, K.L. Shunmuganathan**, 2013, Sentiment classification of movie reviews by supervised machine learning approaches, *Indian J. Comput. Sci. Eng.* 4 285–292.

[219] **S. V Wawre, S.N. Deshmukh**, 2016, Sentiment classification using machine learning techniques, *Int. J. Sci. Res.* 5 819–821.

[220] **C. Bhadane, H. Dalal, H. Doshi**, 2015, Sentiment analysis: measuring opinions *Procedia Comput. Sci.* 45 808–814.

[221] **N. Akkarapatty, A. Muralidharan, N.S. Raj, P. Vinod**, 2016, Dimensionality reduction techniques for text mining, *Collab. Filter. Using Data Min. Anal.* 49.

[222] **X. Ouyang, P. Zhou, C.H. Li, L. Liu**, 2015, Sentiment analysis using convolutional neural network, *Comput. Inf. Technol. Ubiquitous Comput. Commun. Dependable, Auton. Secur. Comput. Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM), 2015 IEEE Int. Conf.* 2359–2364.

[223] **D. Tang, B. Qin, T. Liu**, 2015, Document modeling with gated recurrent neural network for sentiment classification, *Proc. 2015 Conf. Empir. Methods Nat. Lang. Process.* 1422–1432.

[224] **C. Zhou, C. Sun, Z. Liu, F. Lau**, 2015, A C-LSTM neural network for text classification, *ArXiv Prepr. ArXiv1511.08630*.

[225] **Y. Kim**, 2014, Convolutional neural networks for sentence classification *ArXiv Prepr. ArXiv1408.5882*.

[226] **B. Pang, L. Lee**, 2004, A sentimental education: Sentiment analysis using subjectivity

summarization based on minimum cuts, *Proc. 42nd Annu. Meet. Assoc. Comput. Linguist.* 271.

[227] **M. Hu, B. Liu**, 2004, Mining and summarizing customer reviews, *Proc. Tenth ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.* 168–177.

[228] **T. Mikolov, K. Chen, G. Corrado, J. Dean**, 2013, Efficient estimation of word representations in vector space, *ArXiv Prepr. ArXiv1301.3781*.

[229] **R. Kato, H. Goto**, 2016, Categorization of web news documents using word2vec and deep learning *Proc. 2016 Int. Conf. Ind. Eng. Oper. Manag. Kuala Lumpur, Malaysia*.

[230] **W. Huang, J. Wang**, 2016, Character-level Convolutional Network for Text Classification Applied to Chinese Corpus, *ArXiv Prepr. ArXiv1611.04358*.

[231] **L. van der Maaten, G. Hinton**, 2008, Visualizing data using t-SNE *J. Mach. Learn. Res.* 9 2579–2605.

[232] Bokeh Görselleştirme Kütüphanesi. https://bokeh.pydata.org/en/latest/ (En Son Ziyaret-accessed 05/05/2018).

[233] **A. Gulli, S. Pal**, 2017, Deep Learning with Keras, *Packt Publishing Ltd*.

[234] **S. Lai, K. Liu, S. He, J. Zhao**, 2016, How to generate a good word embedding, *IEEE Intell. Syst.* 31 5–14.

[235] **M.W. Gardner, S.R. Dorling**, 1998, Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences, *Atmos. Environ.* 32 2627–2636.

[236] **L. Deng**, 2014, A tutorial survey of architectures, algorithms, and applications for deep learning, *APSIPA Trans. Signal Inf. Process.* 3.

[237] **N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov**, 2014, Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* 15 1929–1958.

[238] **Y. Yao, Z. Huang**, 2016, Bi-directional LSTM recurrent neural network for Chinese word segmentation, *Int. Conf. Neural Inf. Process.* 345–353.

[239] **J.P.C. Chiu, E. Nichols**, 2015, Named entity recognition with bidirectional LSTM-CNNs *ArXiv Prepr. ArXiv1511.08308*.

[240] **T. Mikolov, W. Yih, G. Zweig**, 2013, Linguistic regularities in continuous space word representations, *Proc. 2013 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.* 746–751.

[241] **S. Lai, L. Xu, K. Liu, J. Zhao**, 2015, Recurrent Convolutional Neural Networks for Text Classification, *AAAI*. 333 2267–2273.

[242] **P. Migdal**, n.d., Library for Keras for investigating architectures and parameters of sequential models. https://github.com/stared/keras-sequential-ascii (En Son Ziyaret-accessed 05/10/2018).

[243] **J.H. Lau, T. Baldwin**, 2016, An empirical evaluation of doc2vec with practical insights into document embedding generation, *ArXiv Prepr. ArXiv1607.05368*.

[244] **A.M. Dai, C. Olah, Q. V Le**, 2015, Document embedding with paragraph vectors, *ArXiv Prepr. ArXiv1507.07998*.

[245] **Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin**, 2003, A neural probabilistic language model, *J. Mach. Learn. Res.* 3 1137–1155.

[246] **J. Mitchell, M. Lapata**, 2010, Composition in distributional models of semantics, *Cogn. Sci.* 34 1388–1429.

[247] **F.M. Zanzotto, I. Korkontzelos, F. Fallucchi, S. Manandhar**, 2010, Estimating linear models for compositional distributional semantics, *Proc. 23rd Int. Conf. Comput. Linguist.* 1263–1271.

[248] **A. Yessenalina, C. Cardie**, 2011, Compositional matrix-space models for sentiment analysis, *Proc. Conf. Empir. Methods Nat. Lang. Process.* 172–182.

[249] **E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, M. Baroni**, 2013, Multi-step regression learning for compositional distributional semantics, *ArXiv Prepr. ArXiv1301.6939*.

[250] **M. Chen**, 2017, Efficient vector representation for documents through corruption, *ArXiv Prepr. ArXiv1707.02377*.

[251] **A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts**, 2011, Learning word vectors for sentiment analysis, *Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol. 1*. 142–150.

[252] **M.T. and G.A. Betül Ay Karakuş, İbrahim Rıza Hallaç**, 2017, Derin Öğrenme ile Türkçe Film Yorumlarının Duygu Sınıflandırması, *5.Ulusal Yüksek Başarımlı Hesaplama Kongeransı.*

[253] **G. Aydin**, n.d., RssTurk, Web Kazıma Aracı. https://github.com/galipaydin/rssturk (En Son Ziyaret-accessed 10/05/2018).

[254] **R. Rehurek**, n.d., Deep learning with paragraph2vec. https://radimrehurek.com/gensim/models/doc2vec.html (En Son Ziyaret-accessed 05/10/2018).

[255] **T. Mikolov, M. Karafiát, L. Burget, J. Černock\`y, S. Khudanpur**, 2010, Recurrent neural network based language model, in: Elev. Annu. Conf. Int. Speech Commun. Assoc.

[256] **R. Kiros, Y. Zhu, R.R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, S. Fidler**, 2015, Skip-thought vectors, *Adv. Neural Inf. Process. Syst.* 3294–3302.

[257] **B. Karakus, I.R. Hallac, G. Aydin**, 2015, Distributed Readability Analysis Of Turkish Elementary School Textbooks, *Int. Conf. Inf. Technol. Comput. Sci. Pattaya, Thail.*

[258] **G. Aydin, I.R. Hallac, B. Karakus**, 2015, Architecture and implementation of a scalable sensor data storage and analysis system using cloud computing and big data technologies, *J. Sensors*. 2015 11. doi:10.1155/2015/834217.

[259] **B. Karakus, G. Aydin**, 2016, Call center performance evaluation using big data analytics, *IEEE Int. Symp. Networks, Comput. Commun.* doi:10.1109/ISNCC.2016.7746116.

[260] **Semiha Makinist, Betül Ay Karakuş, İbrahim Rıza Hallaç, Muhammed Talo, Galip Aydin**, 2017, Gensim ve Simserver ile Türkçe Haber Dokümanları Arasında Benzer Dokümanların Sınıflandırılması, *5. Ulus. Yüksek Başarımlı Hesaplama Konf. 14-15 Eylül 2017, Yıldız Tek. Üniversitesi, Davutpaşa, İstanbul.*

[261] **G.A. Semiha Makinist, İbrahim Rıza Hallaç, Betül Ay Karakuş**, 2017, Preparation of Improved Turkish DataSet for Sentiment Analysis in Social Media, *ITM Web Conf. 13*. 13 6. doi:https://doi.org/10.1051/itmconf/20171301030.

[262] **G.A. Betul Ay Karakus, Muhammed Talo, İbrahim Rıza Hallac**, 2018, Evaluating

Deep Learning Models for Sentiment Classification, in: Concurr. Comput. Pract. Exp., . doi:10.1002/cpe.4783.

ÖZGEÇMİŞ

	Betül Ay Karakuş
	Araştırma Görevlisi (2011-), Fırat Üniversitesi Bilgisayar Mühendisliği Bölümü
	Lisans : İstanbul Üniversitesi, Bilgisayar Mühendisliği , 2011. **Yüksek Lisans: "**Biyomekanik Analiz Tabanlı İnsan Hareketi Tanıma Algoritmalarının Geliştirilmesi**"**, Fırat Üniversitesi, Bilgisayar Mühendisliği, 2014. **Doktora** : **"**Derin Öğrenme ve Büyük Veri Yaklaşımları ile Metin Analizi", Fırat Üniversitesi, Bilgisayar Mühendisliği, 2018.
Önceki Deneyimler	• Calleva Yazılım Ltd. , 2014-2016.
Araştırma Alanı	• Derin Öğrenme, Doğal Dil İşleme, Büyük Veri, Bilgisayar Görmesi, Dağıtık Sistemler.
Projeler	• Savunma Sanayii Müsteşarlığı Savunma Geniş Alan (SAGA) Projesi, Derin Öğrenme Büyük Veri Analizi Platformu (DEĞİRMEN) Projesi, 2017-2020, Araştırmacı. • Tubitak Teydeb 1512, Çağrı Merkezleri için Bulut tabanlı, Dağıtık Performans Analizi ve Değerlendirme Sistemi, No: 2140243, 2014-2015, Proje Yürütücüsü. • Fırat Üniversitesi, Bilimsel Araştırma Projeleri (FUBAP) No: M.F.13.11, 2012-2014, Araştırmacı.
Diğer akademik deneyimler	• Photogrammetric Computer Vision Laboratory, Ohio State University, Columbus, ABD, 2012, Araştırmacı. Konu: Video ve görüntülerden insan hareketlerinin tanınması.