T.C.
İSTANBUL SABAHATTİN ZAİM ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ BİLİM DALI

YAPAY ZEKA İLE ORTAK YÖNELİMLİ WEB SİTELERİNİN TESPİTİ

YÜKSEK LİSANS TEZİ

Hasibe Büşra DOĞRU

İstanbul
Haziran, 2020

T.C.
İSTANBUL SABAHATTİN ZAİM ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ BİLİM DALI

YAPAY ZEKA İLE ORTAK YÖNELİMLİ WEB SİTELERİNİN TESPİTİ

YÜKSEK LİSANS TEZİ

Hasibe Büşra DOĞRU

Tez Danışmanı
Dr. Yahya ŞİRİN

İstanbul
Haziran, 2020

TEZ ONAYI

Lisansüstü Eğitim Enstitüsü Müdürlüğüne,

Bu çalışma, jürimiz tarafından Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Bilimleri ve Mühendisliği Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman

Dr. Yahya ŞİRİN

Üye

Dr. Öğr. Üyesi Akhtar JAMIL

Üye

Dr. Öğr. Üyesi Abdulfetah Abdela SHOBOLE

Onay

Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

Prof. Dr. Ali GÜNEŞ

Enstitü Müdürü

BİLİMSEL ETİK BİLDİRİMİ

Yüksek lisans tezi olarak hazırladığım **"Yapay Zeka ile Ortak Yönelimli Web Sitelerinin Tespiti"** adlı çalışmanın öneri aşamasından sonuçlandığı aşamaya kadar geçen süreçte bilimsel etiğe ve akademik kurallara özenle uyduğumu, tez içindeki tüm bilgileri bilimsel ahlak ve gelenek çerçevesinde elde ettiğimi, tez yazım kurallarına uygun olarak hazırladığımı, bu çalışmamda doğrudan veya dolaylı olarak yaptığım her alıntıya kaynak gösterdiğimi ve yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu beyan ederim.

Hasibe Büşra Doğru

ÖNSÖZ

Tez konumun belirlenmesinden tezin son aşamasına kadar bilgi ve deneyimi ile bana yol gösteren değerli tez danışmanım Dr. Yahya ŞİRİN hocama, benden desteklerini hiçbir zaman esirgemeyen anneme ve babama, ayrıca çalışmam boyunca destek ve katkılarıyla yanımda olan değerli arkadaşlarım Sahra Tilki ve Büşra Nur Muslu'ya teşekkür ederim.

Hasibe Büşra DOĞRU
İstanbul - 2020

ÖZET

YAPAY ZEKA İLE ORTAK YÖNELİMLİ WEB SİTELERİNİN TESPİTİ

Hasibe Büşra Doğru

Yüksek Lisans, Bilgisayar Bilimi ve Mühendisliği

Tez Danışmanı: Dr. Yahya Şirin

Haziran – 2020, 77

Çalışmamızda iyi bir sınıflandırıcı yaparak herhangi bir web sitesinin yönelimini tespit etmek için metin sınıflandırma ile hangi tipte sınıf olduğunun bulunabilmesi konusu üzerinde durulmuştur. Farklı sınıflara ait web sitelerinden alınan İngilizce metinlerin vektörleri oluşturulmuştur. Sınıfı bilinmeyen herhangi bir web sitesinden alınan metnin hangi sınıfa ait olduğu belirlenebilir ve böylece yönelimi tespit edilebilir. Bunun için Gensim Kütüphanesi kullanılarak, Doc2Vec'in PV-DM ve PV-DBOW yöntemleri ile model eğitimleri yapılmıştır. Farklı iterasyonlarla yapılan eğitimlerin iki model için de doğruluk oranına etkileri araştırılmak istenmiştir. Buradan elde edilen vektörler üzerinde makine öğrenmesi sınıflandırma yöntemleri Random Forest ve Gauss Naive Bayes ile başarı oranları incelenmiştir. Daha sonra görüntü işlemede etkili bir yöntem olduğu bilinen derin öğrenme yöntemi CNN kullanılmak üzere, Doc2Vec ile elde edilen her bir örneğe ait vektörler resme çevrilmiştir. Makine öğrenmesi sınıflandırma yöntemleri ile elde edilen sonuçların başarı oranları karşılaştırılmış ve ortak yönelimin tespiti açısından etkileri değerlendirilmiştir. Son olarak, Doc2Vec ile oluşturulan belge vektörleri boyut azaltma yöntemleri olan PCA ve t-SNE kullanılarak 2 boyuta indirilip grafiği çizdirilmiştir. Bu çalışmadan elde edilen sonuçlara göre web sitelerinin ortak yönelimlerinin tespit edilmesi konusunda yüksek doğruluk oranı elde edilmiş olup web siteleri başarılı bir şekilde ait oldukları sınıfa göre etiketlenmişlerdir.

Anahtar Kelimeler: Makine Öğrenmesi, Derin Öğrenme, Doc2Vec, CNN, Random Forest, Gauss Naive Bayes

ABSTRACT

DETECTION OF COMMON TENDENTIOUS WEBSITES WITH ARTIFICIAL INTELLIGENCE

Hasibe Büşra Doğru

Master of Science, Computer Science and Engineering

Supervisor: Dr. Yahya Şirin

June – 2020, 77

In our study, it was focused on finding the type of class by text classification to detect the tendentious of any website by making a good classifier. Vectors of English text where taken from websites of different classes have been created. The text extracted from any website of unknown class is able to determine which class it belongs to and thus it's tendentious can be detected. Therefore, model trainings were made with the PV-DM and PV-DBOW methods of Doc2Vec by using Gensim Library. The effects of the trainings with different iterations on the accuracy rate for both methods were investigated. The machine learning classification methods on the vectors obtained from here are analyzed with Random Forest and Gauss Naive Bayes. Afterwards, vectors belonging to each sample obtained with Doc2Vec were converted into a picture with the use the deep learning method CNN, which is known to be an effective method of image processing. The success rates of the results obtained by machine learning classification methods were compared and their effects were evaluated in terms of detecting the common tendentious. Finally, document vectors created with Doc2Vec have been reduced to 2 dimensions and plotted using PCA and t-SNE, which are reduction dimension methods. According to the results obtained from this study, high accuracy rate has been reached in detecting of common tendentious of the websites and the websites have been successfully tagged by the class they belong to.

Keywords: Machine Learning, Deep Learning, Doc2Vec, CNN, Random Forest, Gauss Naive Bayes

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

KISALTMALAR

AI: Artificial Intelligence (Yapay Zeka)

ANN: Artificial Neural Network (Yapay Sinir Ağı)

BNB: Bernoulli Naive Bayes

BOW: Bag-of-Words (Kelime Torbası)

CBOW: Continuous Bag-Of-Words (Sürekli Sözcük Torbası)

CNN: Convolutional Neural Network (Konvolüsyonel Sinir Ağları)

DBOW: Distributed Bag-of-Words (Dağıtık Kelime Torbası)

DM: Distributed Memory (Dağıtık Bellek)

DNN: Deep Neural Network (Derin Sinir Ağı)

DT: Decision Tree (Karar Ağacı)

GNB: Gaussian Naive Bayes

HDNN: Hierarchical Deep Neural Network (Hiyerarşik Derin Sinir Ağı)

HTML: Hyper Text Markup Language (Köprü Metni Biçimlendirme Dili)

LDA: Latent Dirichlet Allocat (Gizli Dirichlet Tahsisi)

LLE: Locally Linear Embedding (Lokal Doğrusal Gömme)

LSTM: Long Short-Term Memory (Uzun Kısa Süreli Bellek)

MDS: Multi-Dimensional Scaling (Çok boyutlu ölçeklendirme)

MNB: Multinomial Naive Bayes

MV: Matrix Vector (Matris Vektör)

NB: Naive Bayes

NLP: Natural Language Processing (Doğal Dil İşleme)

PCA: Principal Component Analysis (Temel Bileşen Analizi)

PV: Paragraph Vector (Paragraf Vektörü)

ReLU: Rectified Linear Unit (Düzleştirilmiş Doğrusal Birim)

RF: Random Forest (Rastgele Orman)

RNN: Recurrent Neural Network (Tekrarlayan Sinir Ağı)

SVM: Support Vector Machine (Destek Vektör Makinesi)

t-SNE: t-Distributed Stochastic Neighbor Embedding (t-Dağıtık Stokastik Komşu Gömme)

URL: Uniform Resource Locator (Düzgün Kaynak Bulucu)

WWW: World Wide Web (Dünya Çapında Ağ)

BİRİNCİ BÖLÜM

1. GİRİŞ

World Wide Web'in (WWW) hızla gelişmesiyle, artık web kullanıcıları tarafından çok miktarda bilgiye erişilebiliyor. Düşük maliyet, yüksek erişilebilirlik ve yayıncılık özgürlüğü, web'in popülerliğine katkıda bulunan özellikleridir. İnternet üzerindeki verinin hızlı biçimde genişlemeye devam etmesiyle metnin hacmi arttıkça, sınıflandırma çalışmalarının geliştirilmesi daha da önem kazanmaktadır. Web'de sayfa içeriğinin sınıflandırılması, odaklanmış tarama, web dizinlerinin destekli gelişimi, konuya özgü web bağlantısı analizi, bağlamsal reklamcılık ve web'in konuyla ilgili yapısının analizi için gereklidir. Web sayfası sınıflandırması, web aramasının kalitesini artırmaya da yardımcı olabilir.

Web sayfaları metin ve multimedya bileşenlerinin yanı sıra, web sayfaları bağlantılar, HTML etiketleri ve meta veriler gibi bağlam özelliklerini içerir. Çoğu araştırma, web sayfalarının metin bileşenlerinin sınıflandırma için birincil bilgi sağladığını, diğer metin dışı bileşenlerin ise sınıflandırma performansını artırmak için kullanılabileceğini varsaymaktadır (Chakrabarti, Dom ve Indyk, 1998; Oh, Myaeng ve Lee, 2000; Craven ve Slattery, 2001; Yang, Slattery ve Ghani, 2002)

Web sitelerinde bulunun metinlerin günlük konuşma dilinde ya da herhangi bir dilde olması mümkündür. Bu yüzden metinleri Doğal Dil İşleme (Natural Language Processing – NLP) ile analiz etmek zorlaşmaktadır. Doğal dilin yapısını ve kelime anlamlarını anlamak dışında anlamlandırma ve sonuç çıkarma NLP açısından zor bir durum haline gelir.

Birçok Makine Öğrenmesi algoritması ve neredeyse tüm Derin Öğrenme mimarileri, dizeleri veya düz metinleri ham formlarında işleyemez. Sınıflandırma, regresyon vb. gibi her türlü işi geniş kapsamda gerçekleştirmek için girdi olarak sayı gerektirirler. Metin biçiminde mevcut olan çok büyük miktarda veriden, bilgi çıkarılması ve uygulamalar oluşturulması bazen zorunludur. Metin uygulamalarının gerçek dünyadaki bazı uygulamaları; duygu analizi, belge veya haber sınıflandırması veya Google tarafından kümeleme vb. örnek olarak gösterilebilir.

Word Embeddings (Kelime Gömmeleri) metinlerin sayılara dönüştürülmesi ve aynı metnin farklı sayısal gösterimlerini elde etmek için kullanılan önemli bir yöntemdir. Bağlamsal bilgiyi düşük boyutlu bir vektörde depolayan bir kelime temsilidir. Bu yaklaşım, 2013'te Word2Vec'in tanıtımı ile, kelime gömmelerini hesaplamalı olarak verimli bir şekilde öğrenen bir model grubu ile aşırı popülerlik kazanmıştır. Word2Vec'in uzantısı olarak görülen, Paragraf Vektör de denilen Doc2Vec'in hedefi ise, bir belgenin temsili vektörünü oluşturmaktır (Le ve Mikolov 2014). Doc2Vec'te kullanılan kelime dizisi n-gram, cümle, paragraf veya belge olabilir.

Yeni bir yöntem olan bu paradigmada kelimeler (Mikolov, vd., 2013) ve belgeler (Le and Mikolov, 2014) için dağıtılmış bir gösterim kullanılmaktadır. İlginç olan, bu temsillerin önceki temsillere göre daha az insan tarafından yorumlanabilse de pratikte iyi iş görmeleridir. Özellikle, Le ve Mikolov (2014) yöntemlerinin Paragraf Vektörlerinin yoğun vektörlerdeki birçok belge anlamını yakaladığını ve film incelemelerinin sınıflandırılmasında veya web sayfalarını incelemek için kullanılabileceğini göstermektedir.

Son yıllarda sınıflandırmanın bilgi çıkarmayı kolaylaştırmak için sınıflandırılması gereken web sayfalarında bulunan yapılandırılmamış metin verilerinin artması nedeniyle daha önemli bir rolü vardır.

Bu tez çalışmasında ortak yönelimli web sitelerinin tespiti konusunda kullanılacak yapay zeka yöntemleri incelenmiştir. Kullanılan yöntemler ile web sitelerinden elde edilen metinler çeşitli aşamalardan geçirilerek ortak yönelimlerin ve birbirlerinden farklı yönelimlerin tespit edilmesi hedeflenmiştir. Belirlenen hedef doğrultusunda, web sitelerinden elde edilen metinler ile Doc2Vec yöntemi kullanılarak ortak yönelimin tespiti ve görselleştirilmesi konusunda sınıflandırmanın etkileri araştırılmaktadır.

Web sitelerinden alınan ham veriler belge olarak kaydedilip makine öğrenmesi teknikleri kullanılarak ön işlem aşamasından geçirilmiştir. Buradan elde edilen vektörlere makine öğrenmesi sınıflandırma ve derin öğrenme yöntemi uygulanır. İki yöntemin de sonuçları doğruluk oranları açısından karşılaştırılmış olup ortak yönelimin tespiti açısından etkileri değerlendirilir. Son olarak, metinlerin vektörel hali görselleştirmek üzere boyut azaltma yöntemleri uygulanıp grafiği elde edilir. Böylece

web sitelerinden alınan metinler ile oluşturulan Doc2Vec model görsel olarak da incelenebilir hale gelmiş olur.

Bu tez kapsamında iyi bir sınıflandırıcı yaparak herhangi bir web sitesini metin sınıflandırmadan geçirerek hangi tipte sınıf olduğunun bulunabilmesi konusu üzerinde durulmuştur. Çalışmadaki metin türleri web sitelerinden belli sınıflara göre alınarak oluşturulmuştur. Farklı dildeki metinler bu tez kapsamına dahil edilmemiştir ve çalışma sadece İngilizce dilindeki metinler üzerinde yapılmıştır.

Bu çalışma kapsamında, tez şu şekilde oluşmaktadır: Bölüm 1'de tez çalışması ile ilgili amaç, kapsam ve katkılar konusunda genel bilgiler verilmiştir. Bölüm 2'de literatürdeki benzer yöntemlerle ilgili çalışmalar açıklanarak ele alınmıştır. Bölüm 3'te Yapay Zeka ile birlikte alt başlıkları olan Makine Öğrenmesi ve Derin Öğrenme konuları teorik açıdan anlatılıp, uygulamanın aşamaları hakkında ayrıntılı bilgi verilmiştir. Bölüm 4'te uygulamada kullanılan deneysel çalışmalardan bahsedilip, veri setinin oluşturulmasından başlayıp görselleştirmesine aşamasına kadar ayrıntılı bilgi verilmiştir. Bölüm 5'te ise bulgular yorumlanarak oluşturulan model ve sınıflandırma yöntemleri karşılaştırılmaktadır. Bu bölümde son olarak modelin zayıf ve güçlü yönlerinden bahsedilip gelecek çalışmalar hakkında öneriler sunulmuştur.

İKİNCİ BÖLÜM

2. LİTERATÜR TARAMASI

2.1 Word2Vec

Kelime gömme oluşturmak için önde gelen yöntemlerden biri Word2Vec'tir (Mikolov, vd, 2013). Mikolov vd. (2013), sürekli kelime gösterimlerini öğrenmek için tek katmanlı bir sinir ağı mimarisinin kullanıldığı Word2Vec modeli olarak bilinen, skip-gram ve sürekli sözcük çantası (CBOW) modelleri önermektedir.

Skip-gram modelinde, çevreleyen bağlam sözcüklerini tahmin etmek için bir kelime kullanılırken, CBOW modelinde çevreleyen bağlam kelimelerini dikkate alarak bir kelime öngörülür. Önerilen modellerde, her kelimenin iki vektörü vardır ve mimari bu iki sözcük vektörünün iç çarpımına dayanır. Önerilen yaklaşım, her bir kelimenin yerel bağlam pencereleri dikkate alınarak kelime temsillerinin öğrenildiği sığ bir pencere tabanlı yöntemdir. Temel olarak, skip-gram modeli, karşılaştırılabilir bağlamlardaki kelimelerin genellikle benzer anlamları temsil ettiğini öne süren dağıtım hipotezi ışığında bir komşuluk koruma hedefini optimize etmeye çalışmaktadır (Harris, Harris, Z. S., 1954). Bu amaç Mikolov (2013) tarafından yapılan çalışmada negatif örneklemeyle stokastik gradyan inişini kullanarak ve Mikolov vd., (2013) tarafından yapılan çalışmada hiyerarşik softmax ile optimize edilmiştir.

Word2Vec yöntemi kelimeler arasında anlamsal ve yapısal ilişkiler kurabilir. Bu yöntemin en güçlü özelliklerinden biri, sözcükleri bir vektör uzayındaki sözcük vektörlerine dönüştürürken kelimeler arasındaki benzerlikleri ve ilişkileri vektörler arasındaki mesafeye aktarabilmesidir. Bu transferin bir sonucu olarak, matematiksel vektörler olarak ifade edilen kelimeler üzerinde vektör cebirinde kullanılan işlemleri (örneğin toplama, çıkarma, mesafe bulma gibi) kullanarak kelimeler arasındaki bazı ilişkileri elde etmek ve kullanmak mümkündür.

Su vd. (2014) tarafından yapılan bir çalışmada, Word2Vec ve SVM, Çince yorum metinlerinin sınıflandırılmasında bir arada kullanılmaktadır. Deneylerin bir parçası olarak, hazır giyim ürünleri ile ilgili binlerce Çinli yorumu tarayarak veri setini oluşturmuşlardır. İlk olarak, semantik özelliklerin çıkarılmasındaki performansını ölçmek amacıyla aynı ürün özelliğini ifade eden eş anlamlıları kümelemek için

Word2Vec'i uygulamışlardır. Ardından, yorum metinlerini sınıflandırmak için SVM'i kullanırlar. Önerilen Word2Vec ve SVM temelli duygu sınıflandırmasının en iyi deneysel sonuçları, sözlüğe dayalı özellik seçme yöntemi ya da part-of-speech tabanlı yöntem olup olmadığına bakılmaksızın, kullanılan yöntemlerden herhangi birinde %90 doğrulukla performans göstermektedir. Su vd. (2014), göre bu performans, Word2Vec'in duygu analizi için uygunluğunu göstermektedir.

Ouyang vd. (2015) çalışmalarında, Word2vec ve CNN'i bir arada kullanmışlardır. İşl olarak duygu analizi görevi için 3 çift kıvrımlı katman ve havuz katmanından oluşan CNN mimarisi tasarlamışlardır. Daha sonra beşli etiket içeren film inceleme alıntılarının bir parçası olan halka açık bir veri kümesinde test edilmiştir: negatif, biraz negatif, neural, biraz pozitif ve pozitif. Ağ, bu veri kümesinde %45,4'lük test doğruluğu elde edip; bu sonucun, RNN ve Matrix-Vector RNN (MV-RNN) gibi diğer sinir ağ modellerinden daha iyi bir performans gösterdiğini söylemişlerdir.

Hughes vd. (2017) tarafından yapılan başka bir çalışmada, konvolüsyonel sinir ağları (CNN) ve Word2Vec, klinik metni otomatik olarak cümle düzeyinde sınıflandırmak için kullanılmıştır. Yazarlar, ağı sağlık bilgilerinin geniş bir şekilde sınıflandırılmasını sağlayan bir veri kümesi üzerinde eğitmektedir. Detaylı bir değerlendirme yoluyla, yöntemlerinin doğal dil işleme görevlerinde yaygın olarak kullanılan çeşitli yaklaşımlardan %15 daha iyi performans elde edildiğini göstermektedir.

Ertuğrul, Onal ve Acartürk (2017) çalışmasında, regresyonun Türk tweet'lerini kullanarak duyarlılık analizinde güven puanları üzerindeki etkisi incelenmiştir. Sözcük özellikleri, ifadeler ve duyarlılık puanları dahil olmak üzere el ile özellikleri çıkarırlar. Yazarlar ayrıca regresyon ve sınıflandırma için tweet'ler üzerinde kelime gömme yöntemi kullanmaktadır. Bulgular, regresyon kullanmanın duygu sınıflandırma doğruluğunu biraz geliştirdiğini göstermektedir. Dahası, kelime gömme işlemlerinin elle çıkarılmış özelliklerle birleştirilmesi, özellik boyutunu azaltmış ve alternatif özellik kombinasyonlarından daha iyi performans elde edilmiştir.

Çoban (2017) tarafından yapılan bir çalışmada, Türk müziği tür sınıflandırma sürecinde ses ve lirik özellikler kullanılmıştır. Metinsel özellikler, Word2Vec ve geleneksel kelimeler torbası (BOW) gibi çeşitli özellik çıkarma modellerinin kullanımıyla şarkı sözlerinden çıkarılır. Deneyler, SVM sınıflandırıcı algoritması kullanılarak gerçekleştirilir; bunu, özellik seçiminin ve farklı özellik gruplarının müzik

türü sınıflandırması üzerindeki etkisinin analizi izler. Şarkı sözü temelli müzik türü sınıflandırmasını bir metin sınıflandırma görevi olarak görürken, yazar aynı zamanda terim ağırlıklandırma yönteminin etkisini de inceler. Deneysel sonuçlara göre, metinsel özellikler Türk müzik türü sınıflandırmalarında, özellikle gözetimli bir terim ağırlıklandırma yöntemi ile kullanıldığında, ses özellikleri kadar etkili olabilir. Çalışma, ses özelliklerinin tek başına kullanımının %98 başarı oranıyla sonuçlandığını gösterirken, sadece şarkı sözü özelliklerinin kullanılması, 4-gram yöntemle %94,32 doğruluk oranı verdiğini gösteriyor. Öte yandan, sözlerin ve ses özelliklerinin bir kombinasyonu %99,12 ile en yüksek başarı oranına ulaşıyor.

Arabacı vd. (2018) yaptıkları çalışmada, anlamsal olarak benzer cümleleri bulmak için Word2Vec yöntemi ile Esen ve Özkan (2017) tarafından önerilen Fisher kodlamasını yapan bir uygulama önerilmiştir. Bu yöntem, benzer cümleyi bulmak için geniş bir cümle ile test edilmiştir. Sonuçlar, önerilen yöntemin cümle benzerliği sorununa etkili bir çözüm sağladığını göstermektedir.

2.2. Doc2Vec

Mikolov vd. (2013) kelime gömme (Word2vec) öğrenme yaklaşımını takip ederek, Le ve Mikolov (2014) Paragraf Vektörü adı verilen başka bir algoritma önermektedir. Ayrıca Doc2vec olarak da bilinir, bu isim Gensim (Rehurek ve Sojka, 2010) paketinden gelir. Bu, belgeler ve cümleler için dağıtılmış sunumları öğrenmek için gözetimsiz bir yöntemdir. Word2vec yöntemine çok benzerdir. Aradaki fark, Doc2vec yönteminde her cümle veya paragraf için bir vektör bulunmasıdır. Doc2vec yöntemi, belgenin konusunu yakalamayı hedefleyen standart dil modeline bir bellek vektörü eklediği dağıtık bellek (Distributed Memory, DM) modelini kullanır.

Le ve Mikolov (2014) bu yöntemi Stanford Sentiment Treebank ve IMDB veri kümeleri üzerinde bir duyarlılık analizine uygular. Denemelerinde, Doc2vec'in duyarlılık analizi için kullanılabilecek film inceleme metinlerini yerleştirmeyi öğrenebildiklerini göstermektedir. Buna göre, yöntemleri, duyarlılık analizi sonuçlarını önceki en iyi sonuçtan %1,3 oranında iyileştirmektedir.

Dai, Olah ve Le (2014) tarafından yapılan bir başka çalışma da, Doc2vec performansını Latent Dirichlet Allocation (LDA) gibi diğer belge modelleme algoritmalarıyla karşılaştırmıştır. Vikipedi veri setindeki ve arXiv veri setindeki

modelleri kıyaslarlar. Bu çalışma ile elde ettikleri sonuca göre Doc2vec algoritmasının diğer yöntemlerden daha iyi performans gösterdiğini belirtiyorlar.

Wieting vd. (2016), Paraphrase Veritabanından geniş çaplı bir eğitim seti olan paraphrase çiftleri temel alınarak hazırlanan belge gömmelerinin daha doğrudan bir şekilde öğrenilmesini önermiştir (Ganitkevitch vd., 2013). Bir paragraf cümlesi, sözcük gömme işlemi ve bir cümle gömme için kelime gömme oluşturma yöntemine bakıldığında, sinir ağı modelinin amaç fonksiyonu, çift gömme için cümle gömme kosinüsünün benzerliğini en üst düzeye çıkaracak şekilde gömme kelimesini optimize etmektir. Yazarlar, kelime gömme işlemlerini birleştirmenin birkaç yöntemini keşfetmiş ve basit ortalamanın en iyi performansı sağladığını bulmuştur.

Lau ve Baldwi (2016) yaptıkları çalışmada Doc2vec tarafından öğrenilen belge gömme işlemlerinin kalitesi, iki temel yöntemle, word2vec sözcük vektörü ortalama alma ve bir n-gram model, ve iki rakip belge gömme yöntemi olan dm ve dbow karşılaştırılması ile ölçülmüştür. Genel olarak, doc2vec'in iyi çalıştığını ve dbow'un dm'den daha iyi bir model olduğu görülmektedir. Genel amaçlı uygulamalar için en iyi doc2vec hiper-parametre ayarları konusundaki önerilere ampirik olarak ulaşılmış ve büyük dış kurumlar kullanılarak eğitildiğinde bile doc2vec'in sağlam bir şekilde performans gösterdiğini ve önceden eğitilmiş kelime yerleştirmelerinden faydalandığı sonucuna ulaşılmıştır.

Hashimoto vd. (2016) tarafından önerilen konu saptama yöntemi, belgeler arasındaki anlamsal benzerlikleri yakalamak için bir sinir ağı tabanlı vektör uzayı modelini kullanmaktadır. Öncelikle vektör uzayındaki belgeler temsil ediliyor ve belgeleri önceden tanımlanmış sayıda kümeye ayrılıyor. Kümelerin ağırlık merkezleri gizli konular olarak ele alınmıştır. Daha sonra her belgeyi gizli konuların bir karışımı olarak temsil etmişlerdir. Değerlendirme amacıyla hem yeni konu tespit yöntemlerini hem de temel konu modelini (Latent Dirichlet Allocation, LDA) kullanarak aktif öğrenme stratejisini kullanışlardır. Deneysel kanıtlar, sinir ağı tabanlı konu tespit yönteminin, temel yönteme kıyasla gelişmiş bir verim ve yük performansı elde edilmiştir. Ek olarak, beş incelemeden dördünde, önerilen yöntemin son incelemede uygun çalışmaların %95'ini koruyarak manuel açıklama maliyetini büyük ölçüde düşürdüğünü göstermiştir.

Karvelis vd. (2018) tarafından yapılan bir çalışmada, her iki bileşen için farklı aday çözümleri araştırılmış ve karşılaştırılmıştır. İlk aşamada hem dağıtılmış doc2vec hem de geleneksel BOW bileşenleri kullanılırken, ikincisi için çoklu etiket sınıflandırma alanından iki farklı dönüşüm yaklaşımı karşılaştırılmıştır. Karşılaştırma için, MIT Kütüphaneleri Dspace deposundan bir doktora özeti (yaklaşık 19000 doküman) koleksiyonu, farklı kombinasyonların yüksek kalitede çözümler sunabileceğini düşünülmektedir. Sonuçlar, doc2vec temsillerinin, geleneksel BoW ile karşılaştırıldığında daha iyi temsil bilgisi sunduğunu ve verilen konular arasındaki korelasyonu dikkate alan çok dilli bir sınıflandırma şeması ile birleştirildiğinde oldukça yüksek performans sağladığını göstermektedir.

Safalı vd. (2019) Do2Vec modeli ile birlikte derin öğrenme yöntemleri kullanarak Türkçe sempozyumlarda yayınlanan çalışmaların sınıflandırılması üzerine bir çalışma yapmışlardır. Sınıflandırma için RNN ve LSTM sinir ağı kullanmışlardır. Kullanılan veri setinde 9 farklı sınıftan oluşan 90.000 adet akademik çalışma bulunur. Yapılan sınıflandırmada PV-DBOW ve PV-DM yöntemleri karşılaştırılmış ve en başarılı sonuç PV-DBOW yönteminde LSTM ile sınıflandırma sonucunda %93,54 olarak elde edilmiştir.

2.3 Metin Sınıflandırma

Kelime ve belge gömme yöntemleri ile birlikte derin öğrenme ve makine öğrenmesi sınıflandırıcıları kullanılarak farklı diller için metin sınıflandırma çalışmaları bulunmaktadır. Literatürde son yıllarda popüler olan metin gömme yöntemlerini kullanan birçok çalışmada geleneksel makine öğrenme yöntemlerine ek olarak, CNN, LSTM ve benzer birçok sinir ağı kullanılmıştır. Çalışmaların çoğu İngilizce metinler için yapılmış olmasına rağmen, son yıllarda diğer diller için yapılan çalışmaların sayısı artmıştır.

Şen ve Erdoğan (2014) 52 milyon Wikipedia veri kümesi ve Boğaziçi Üniversitesi tarafından oluşturulan metin veri kümesi olmak üzere iki veri seti kullanmışlardır. Çalışmada, Word2vec yönteminin skip-gram modeli seçilmiştir. Kelime vektörleri iki algoritma kullanılarak elde edilir: negatif örnekleme ve hiyerarşik softmax. Çalışmada bu iki yöntem karşılaştırılmış ve negatif örneklemenin daha başarılı olduğu

bulunmuştur. Ek olarak, vektör boyutunun semantik ve sözdizimsel doğruluk üzerindeki etkisi ölçülür ve en yüksek başarı 200-400 aralığında elde edilir.

Severyn ve Moschitti (2015), Semeval-15 corpus'ta mesaj ve tümce düzeyinde duygu analizini uygulamak için kelime gömme yöntemleri ve CNN kullanmışlardır. Deneysel analiz sırasında; ağ parametreleri için rastgele değerler, 50 milyon tweet üzerinde eğitilmiş Word2vec vektörleri kullanılmıştır. Deneyler sonucunda; tümce düzeyinde duygu analizi için önerilen yöntem en iyi, cümle düzeyinde duygu analizi ise duyarlılık analizi için ikinci en iyi yöntem olarak görülmektedir.

Yapay sinir ağları büyük verilerle çalışmayı kolaylaştırır; ayrıca, veri boyutu arttıkça sınıflandırma performansının arttığını kanıtlayan çalışmalar da vardır. Bu çalışmalardan biri Hu vd. (2015) büyük ölçekli verilerde belge düzeyinde duygu analizi için HDNN mimarisini önermiştir. Kelime frekansları, bağlamsal pencere ve POS etiketleme özelliklerinin kombinasyonu, yapay sinir ağına girdi olarak verilir. Deneylerde, Amazon'un elektronik ürün incelemeleri, IMDB'den film incelemeleri ve TripAdvisor'daki otel yorumları HDNN, SVM ve NB sınıflandırıcıları ile sınıflandırılmıştır. Deneyler, her veri kümesinin farklı boyutları için tekrarlanır. DNN, tüm veri kümeleri için en yüksek sınıflandırma başarısını elde etmiştir. Verilerin boyutu arttıkça, 3 veri seti için HDNN'nin sınıflandırma başarısında bir artış vardır. DNN'nin hem büyük ölçekli veri sorunlarını hem de etki alanı bağımlılığı sorununu çözdüğü sonucuna varılmıştır.

Yang ve Xia (2016), çalışmalarında, son katmanı doğrusal bir sınıflandırıcı olan bir CNN mimarisini, Word2vec yöntemini kullanarak belgeleri duygularına göre sınıflandırmak için kullanıyorlar. 4 farklı veri seti (3 dengeli, 1 dengesiz sınıf dağılımı) içeren Çin otel incelemeleri derlemini kullanırlar. Belgeleri sınıflandırmak; bir CNN mimarisi, SVM ve NB sınıflandırıcılar kullanılır. Deneylerin sonunda, dengesiz veri seti dahil tüm veri setlerinde NB ile en kötü sonuçlar elde edilirken, en iyi sonuçlar CNN ile gözlenmektedir.

Hassan ve Mahmood (2017) CNN ve LSTM mimarilerinin birlikte kullanıldığı ConvLstm adlı yapay bir sinir ağı önerdi. Önerilen bu mimaride, CNN girdi verilerinden özellikler çıkarmak için kullanılır; ve önemli bilgileri hatırlamak ve uzun vadeli bağımlılıkları yakalamak için havuzlama katmanı yerine LSTM kullanılır. Çalışmada; deneyler, IMDB ve Stanford Sentiment Treebank üzerinde önceden

eğitilmiş Word2vec vektörleri kullanılarak duyarlılık analizi için gerçekleştirilir. Elde edilen sonuçların, bu veri kümelerini kullanan önceki çalışmalara göre daha başarılı olduğu bulunmuştur. Deneyler sonucunda; daha az parametre ile eğitilen bu ağ mimarisi, diğer modellere göre daha yüksek başarı elde ederek diğer yöntemlere alternatif olarak kabul edilmiştir.

Literatürde İngilizce metinler dışında farklı diller üzerinde de geleneksel temsil yöntemleri ve kelime gömme yöntemleri ile temsil edilen makine öğrenmesi ve derin öğrenme yöntemlerini kullanarak duygu ve belge sınıflandırması uygulayan çalışmalar bulunmaktadır.

Çince için yapılan bir başka çalışma Huang vd. (2017) Çin mikro-blog verilerinde Word2vec kullanarak kelime vektörleri elde eden ve sınıflandırıcılar olarak CNN, LSTM, tek katmanlı CNN ve ağ yapısı ve iki LSTM katmanlı SVM uygulalarlar. Duygu sınıflandırmasının sonuçlarına göre; önerilen yöntem (tek katmanlı CNN + 2 katmanlı LSTM) %87,2 doğruluğa sahipken, SVM, CNN, CNN-LSTM ve LSTM sırasıyla %86, %85,6, %84,2 ve %83,8 doğruluklara sahiptir. Sonuç olarak; CNN veya LSTM'in hibrid modellerden daha az başarılı olduğu sonucuna varılmıştır.

Vo vd. (2017) 2 farklı Vietnam Twitter veri seti üzerinde çalışmıştır. CNN, LSTM, çok katmanlı CNN-LSTM ve SVM sınıflandırıcılar kullanılır. SVM, kelime torbası kullanılarak elde edilen özelliklerle uygulanır; diğer sınıflandırıcılar için, önerilen yöntemin gömme katmanındaki özellikler kullanılır. Sınıflandırma performansları karşılaştırıldığında, en yüksek başarı CNN ve LSTM birlikte kullanılarak elde edilir.

Amasyalı vd. (2018), metin sınıflandırması için sözcük, anlamsal ve karakter temsillerini karşılaştırmışlardır. 3 sınıf ve 17.289 tweet içeren bir veri seti oluşturdular. Metin gösterimi yöntemleri için BOW, n-gram ve Fasttext kullanılır. BOW için hem frekans hem de ikili gösterim yöntemleri kullanılır. Sınıflandırıcı olarak SVM, RF, CNN ve LSTM uygulanır. Sonuç olarak; yeni nesil yaklaşımların geleneksel yöntemlerden biraz daha başarılı olabileceği ve her ikisi için de karakter temelli sunumların daha yüksek sınıflandırma doğruluğu olduğu gözlenmiştir.

ÜÇÜNCÜ BÖLÜM

3. YAPAY ZEKA

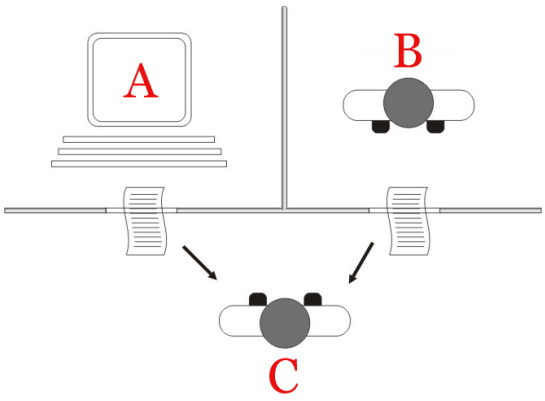
Günümüzde yazılımları geliştirerek zeki hale getirmeye yönelik çalışmalar kullanıcıları gereksiz ya da tekrar gerektiren işlerle uğraştırmadan verimi artırmalarını sağlamak amacıyla fazlasıyla önem kazanmıştır. Burada amaç, yazılımların öğrenmesini ve öğrendiklerinden yola çıkarak davranışlarını belirlemesini ya da değiştirmesini sağlamaktadır.

Zeki olan ya da olmayan doğal yazılım sistemlerinin yapabildiğine benzer olarak davranabilen bir bilgisayar ya da robotun kabiliyetine Yapay Zeka (Artificial Intelligence, AI) denir. Yapay Zekada 4 temel konu ön plandadır:

İnsan gibi davranabilmek (Bir yazılımın insan gibi davranabilmesi); Bu durum 1950 yılında Alan Turing tarafından geliştirilen Turing testi ile açıklanır (Turing, 1950). Turing testi bir makinenin bir insanınkine eşdeğer veya ayırt edilemez davranış sergilemeye yeteneğinin bir testidir. Turing, bir insanın bir insan ile insan gibi tepkiler üretmek için tasarlanmış bir makine arasındaki doğal dil konuşmalarını yargılamasını önermiştir. Örneğin, bir duvarın arkasındaki iki kişi ile yazışıyorsunuz. Bir tanesinde gerçek bir insan (B) var diğerinde ise bir yapay zeka yazılımı (A) var.

Burada bilinmesi gereken soru; bu yazışmalardan yola çıkarak, cevapların yapay zeka yazılımına mı yoksa bir insana mı ait olduğunu tespit edebilir misiniz?

Yazılıma her türlü soru sorabilirsiniz. Bu konuşmanın sonucunda eğer değerlendirici (C) makineyi insandan ayırt edemezse, makine testi geçmiş demektir. Test sonuçları makinenin sorulara doğru cevaplar verme yeteneğine bağlı değildir, cevaplarının bir insanın vereceği cevaplara ne kadar yakın olduğuna bağlıdır.



Şekil 3.1: Turing Testi Diagramı

İnsan gibi düşünebilmek (Bir yazılımın insan gibi düşünebilmesi); Bilişsel bilim (cognitive science) insan gibi düşünmeyi hedefleyen bir bilimdir. Bunun içerisinde psikoloji, dilbilimi, sosyoloji, davranış bilimi, matematik, mantık ve felsefe gibi birçok bilim var. Sistemin problemleri insanın çözdüğü gibi çözmesi asıl amaçtır.

Rasyonel bir şekilde düşünebilmek (Bir yazılımın rasyonel bir şekilde düşünebilmesi); Mantık kullanmak ve mantıksal çıkarımlar yapmak. Tümevarım ve Tümden gelim kavramlarının kullanılması. İspat edilebilir bir sonuca ulaşmak.

Rasyonel bir şekilde hareket edebilmek (Bir yazılımın rasyonel bir şekilde hareket edebilmesi); Mantık kullanarak tespit edilen aksiyonları gerçekleştirmek.

3.1 Makine Öğrenmesi

Makine öğrenmesi, yapay zekanın insan öğrenme yeteneğine benzer, ancak insan beyni yerine bir bilgisayarla önemli bir parçasıdır. Ayrıca, hesaplanamayan ve istatistiksel algoritmaları kullanarak çözülemeyen sorunları çözebilir ve bilgisayarlara alınan muazzam miktarda veriyi öğrenmesini öğretir. Dahası, normal algoritmaları kullanamayan ve değişkenleri arasında net bir ilişki olmayan zor sorunları temsil edebilen bir yöntemdir. Ancak, bu yöntemlerin belirli bir haritada eğitilebilmesi için girdiler ve hedefler gereklidir; tahmin edilen bir çıktı vermek için modele bağlıdır, daha sonra gerçek hedefle karşılaştırılır ve hedef ile tahmin edilen çıktı arasındaki kayıp fonksiyonunu en aza indirir; en iyi model elde edilene kadar. Makine öğrenmesi, sınıflandırmalar, örüntü tanıma, spam filtreleme, zaman serisi tahmini ve tahmin gibi bir dizi gerçek yaşam durumunda kullanılır. Günümüzde yaygın olarak kullanılan, en popüler iki makine öğrenmesi türü gözetimli ve gözetimsiz öğrenmedir.

3.1.1 Gözetimli Öğrenme

Gözetimli öğrenmede bilinen girdiler için çıktılar tahmin edilmeye çalışılır. En yaygın tür geri yayılımdır (backpropagation). Gözetimli öğrenme, hatalardan öğrenmedir. Sistem tahminlerini hedef çıktı ile karşılaştırır ve hatalarından ders alarak hedefe yakınlaşır. Ağırlık girişler üzerinden uygulanır, ardından girdiler bir sonraki düğümden geçer, orada güncellenen ağırlıklar toplanır ve yoğunlaştırılır veya zayıflatılır. Başlangıçta sistem rastgele ağırlıklarla başlar, doğru çıktı hakkında ilk tahminde bulunur, tahmin değeri ile doğru çıktı arasında hata olarak bilinen farkı alır. Hata oranını en aza indirmek için ağırlıklar değiştirilir ve bu tekrarlamalı tahmin süreci, veriler çıktı katmanına ulaşana kadar devam eder. Bu katmanda gerçek çıktı ile karşılaştırma yapılır ve veriler eşit ise ağırlıklar üzerinde değişiklik yapılmaz, aksi halde hata sistem üzerinden yayılır ve hedef elde edilinceye veya neredeyse yaklaştırılıncaya kadar ağırlıklar güncellenir.

3.1.2 Gözetimsiz Öğrenme

Gözetimsiz öğrenme, açıklamadan ziyade tahmin ederek gözetimli öğrenmeden farklılık gösterir. Bu öğrenme yönteminde sinir ağının gerçek çıktı hakkında hiçbir fikri yoktur. Bu mimaride hiçbir çıktı yoktur. Gözetimsiz öğrenme, benzer verileri kümelemek için ideal bir işlemdir. Basitçe, gözetimsiz öğrenme eğitim verileri olmadan öğrenirken gözetimli öğrenme, eğitim verilerine dayalıdır. Sorunun türüne bağlı olarak, bu iki yöntem de en iyi performansı elde etmek için karıştırılabilir.

3.1.3 Kelime Gömme

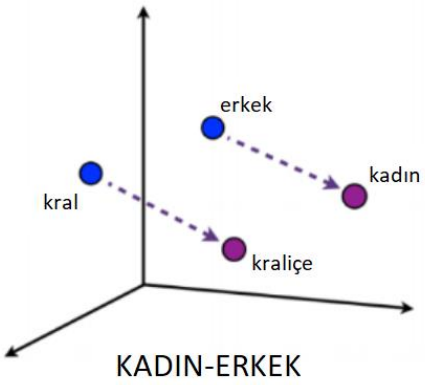
Bu tezde uygulanan metin gösterimi yöntemleri sinir ağı mimarisine dayanan kelime gömme yöntemi olan Word2Vec'ten türetilmiş Doc2Vec yöntemi ile yapılacaktır. Yapay sinir ağları gibi kategorik değişkenlerle doğrudan çalışamayan yapılar için belgeleri sayısal olarak temsil etmenin bir yolu da vektörleri kullanmaktır. Belgeler sözcük torbası kullanılarak temsil edildiğinde, belgelerdeki sözcüklerin sırası kaybolur ve belgeleri temsil eden vektörler yüksek boyutlu ve seyrek olur. Bu nedenle, bu sorunlardan kaçınmak için kelime gömme yöntemleri önerilir. Gömme yöntemlerinin

temel amacı, kelimeler arasındaki anlamsal ilişkileri kaybetmeden, düşük boyutlu vektörleri olan kelimeleri temsil etmektir.

3.1.3.1 Word2Vec

Mikolov ve arkadaşları tarafından geliştirilmiş, kelimelerin dilsel bağlamlarını yeniden inşa etmek amacıyla eğitilmiş sığ ve iki katmanlı sinir ağlarıdır. Bir tahmin modeli olan Word2vec, kelimeleri vektör uzayında konumlandırır ve ortak bağlamları olan sözcükleri, uzayda birbirine yakın olacak şekilde yerleştirir (Mikolov vd. 2013).

Genel olarak kelimeleri kullanarak model oluşturmak istediğimizde, basitçe onları etiketleme yöntemini kullanırız. Bununla birlikte böyle bir yöntemle kelimeler anlamlarını kaybeder. Örneğin,

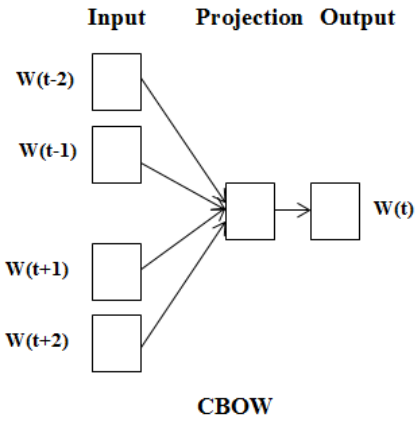


Şekil 3.2: Kral ile erkek, kraliçe ile kadın arasındaki benzerlik

Word2vec, iki farklı teknik kullanarak kelimeler arasındaki ilişkileri kullanan kelimelerin vektör temsillerini tanımlar: Sürekli Sözcük Torbası (CBOW) ve Skip-Gram. Bu yöntemlerin her ikisi de sinir ağının her bir kelimenin kelime temsilini nasıl öğrendiğini açıklar. Öğrenme kelime gösterimleri temelde gözetimsiz bir algoritma olduğundan, verilen girdi için mimariye bağlı olarak modeli eğitmek için etiketler oluşturulmalıdır. Kelime gösterimi üreten başka derin veya tekrarlayan sinir ağı mimarileri olsa da, Word2vec diğer modellere göre hızlı bir şekilde öğrenir. Diğer yöntemlerle ilgili temel sorun, bu modelleri eğitmek için gereken nispeten daha uzun zamandır.

Sürekli Sözcük Torbası modeli (CBOW), hedef kelimeyi çevreleyen bağlamsal kelimelerle, örneğin; wi_2, wi_1, $wi + 1$, $wi + 2$ giriş sözcüklerini alarak hedef kelimesini tahmin eder (Şekil 3.3). Bu model sinir ağı dil modeline benzer, doğrusal olmayan gizli katman kullanmak yerine, tüm kelimeler için paylaşılan doğrusal projeksiyon katmanını kullanır. Buna sözcük torbası denir, çünkü sözcüklerin sırası, projeksiyon katmanında önemli değildir. Eğer kelimeler birbirine yakınsa, anlamlarının bir şekilde benzer olduğu ve algoritmanın uzak kelimelere daha az ağırlık verdiği anlamına gelir. Eğitim sırasında, hata geri yayılır ve benzer kelimeler benzer bağlamlarda göründüğü için, benzer kelimelerin vektörleri, doğru kelimeyi tahmin etmeleri için benzer yönlere doğru güncellenir. CBOW model şeması, Şekil 3.3'te gösterilmiştir.

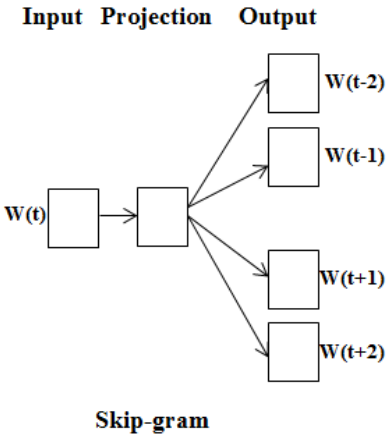
Word2vec, birbirine genel olarak benzeyen Continous Bag of Words (CBOW) ve Skip-Gram adı verilen iki model mimarisinden birini kullanır ve böylece sözcüklerin dağınık bir gösterimini üretir. Skip-gram modelinde, bir dizi cümle verildiğinde, model her cümlenin kelimelerini özetler veya komşularını tahmin etmek için o anki kelimesini kullanmaya çalışır. CBOW modelinde ise, hedef kelimeyi tahmin etmek için komşuların her biri kullanılır. Skip-gram, CBOW'a göre daha yavaştır, ancak seyrek kelimeler için daha iyi sonuç elde edilir (Mikolov vd. 2013).



Şekil 3.3: CBOW Model Mimarisi.

Şu anki kelime $w(t)$ ve bağlamsal kelimeler $w(t-2) \ldots w(t+2)$'dir (Mikolov vd. 2013).

Skip-gram modelinde ise, bir dizi cümle verildiğinde, model her cümlenin kelimelerini özetler veya komşularını tahmin etmek için o anki kelimesini kullanmaya çalışır. Skip-gram modelinin eğitim amacı, bir cümleyi ya da belgedeki çevreleyen kelimeleri tahmin etmede yararlı olan kelime temsillerini bulmaktır (Mikolov, vd., 2013).



Şekil 3.4: Skip-Gram Model Mimarisi

Skip-gram modeli CBOW modelinin tam tersidir; giriş katmanındaki bağlam kelimelerini kullanmak yerine hedef kelimeleri kullanırız. Gizli katman aynı kalır ve bağlam sözcüklerini çıktı katmanında buluruz.

3.1.3.2 Doc2Vec

Paragraf vektörü, kelime gömmelerine (word embeddings) dayanan belge gömülmelerini öğrenmek için Word2vec'in bir uzantısı olarak görülebilir. Mikolov vd. (2013) kelime gömme öğrenme yaklaşımını takip eden Le ve Mikolov (2014), Paragraf Vektörü adı verilen başka bir algoritma önermektedir. Bu yöntem Doc2vec olarak da bilinir, bu ad Gensim (Rehurek ve Sojka, 2010) paketinden gelmektedir.

Doc2vec modeli algoritmasını Word2Vec'den alır. Word2Vec'de kelimeleri etiketlemeye gerek yoktur, çünkü her kelimenin kelime haznesinde kendi anlamsallığı vardır. Ancak, Doc2Vec durumunda, kelimenin veya cümlenin kaç tane anlam taşıdığını, bununla beraber algoritmanın bunu tek bir varlık olarak tanımlayabileceğini

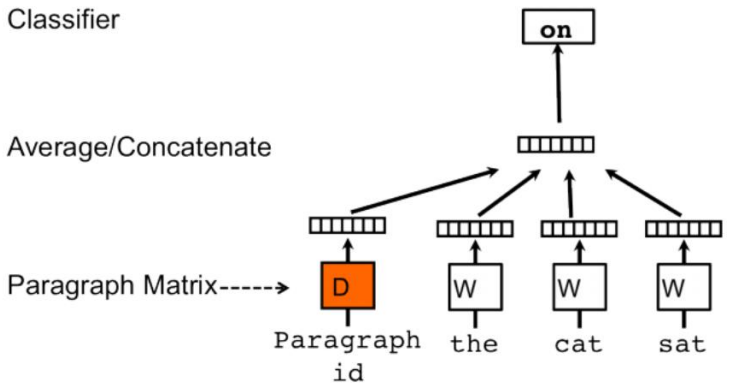
belirtmeye ihtiyaç vardır. Bu nedenle, aktarılan anlam seviyesine bağlı olarak cümleye veya paragrafa etiket veya etiketler belirlenmektedir.

Paragraf vektöründe, vektörün kendisi her bağlamda yerleştirilerek, bağlamdaki tüm kelimelerin anlamını kavramaya çalışır. Böylece, paragraf vektörü, eğitilen bağlamdaki tüm kelimelerin anlamsallığını içerir.

Bir paragraf vektörü, yüksek düzeyde, mevcut bağlamda neyin eksik olduğunu hatırlayan veya paragrafın konusunu hatırlatan yeni bir belirteçtir. Taktik düzeyde, örneğin fizikle ilgili kelimeleri içeren bir belgeden alınan bir cümlenin, bilimsel kelimeleri kullanma olasılığının daha yüksek olduğunu kabul etmek için tasarlanmıştır. Bu tür belge düzeyinde içerik Word2Vec'in bir parçası değildir. Doc2Vec, Word2Vec üzerine kuruludur ve Doc2Vec, Word2Vec'in kelime vektörleri matrisini tuttuğu gibi bir paragraf vektör matrisi tutar.

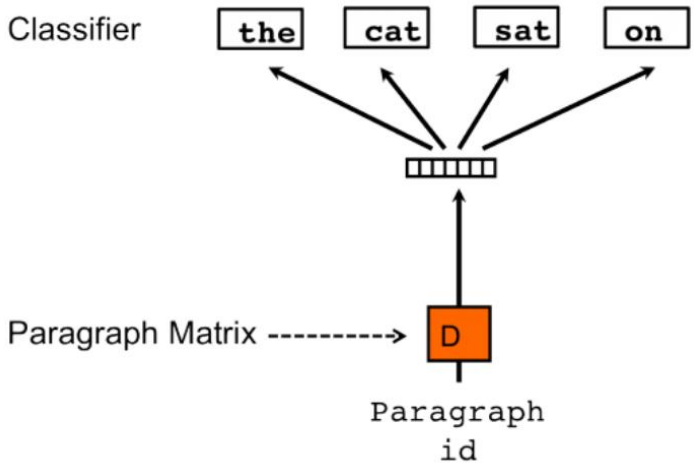
Doc2Vec, paragraf vektörünü kullanmak için paragraf vektörlerinin dağıtık bellek modeli (PV-DM) ve paragraf vektörünün dağıtık kelime torba modeli (PV-DBOW) adında iki yaklaşım sunar. Bu yaklaşımlar, Word2Vec'te kullanılan CBOW ve skip-gram yöntemlerine çok benzerler.

Paragraf vektörlerinin dağıtık bellek modelinde (PV-DM), paragraf kimliğinin (paragraf_id) sıralı bir kelime dizisine başka bir kelime olarak eklenmesidir. PV-DM, cümle içindeki diğer kelimelere ve paragraf kimliğinin içeriğine göre sıralı bir kelimeyi tahmin etmeye çalışır. Giriş kelimesi vektörlerinin sınıflandırılması işleminin bir parçası olarak ortalaması alınır, toplanır veya birleştirilir.



Şekil 3.5: Paragraf Vektörlerinin Dağıtık Bellek Modeli (PV-DM)

Paragraf vektörünün dağıtık kelime torba modeli (PV-DBOW), Word2Vec'te skip-gram'ın çalışma şekline benzer. Problem ters yönden ele alınır. PV-DBOW, verilen bir paragraf kimliğini alır ve sözcük sıralaması üzerinde herhangi bir kısıtlama olmadan penceredeki kelimeleri tahmin etmek için kullanır.



Şekil 3.6: Paragraf Vektörünün Dağıtık Kelime Torbası Modeli (PV-DBOW)

PV-DM'nin çoğu görev için iyi çalıştığını öne sürülüyor olsa da araştırmacılar tarafından yapılan deneysel çalışmalarda, PV-DBOW'un daha basit bir model olmasına rağmen PV-DM modeline göre daha iyi çalıştığını göstermektedir. Gensim Doc2Vec kütüphanesi yazarları da PV-DBOW'un PV-DM'yi geride bıraktığını bulmuştur (Rehurek ve Sojka, 2010). Bunu Word2Vec ile karşılaştırdığımızda, Word2Vec'deki her sözcük kendi anlamsallığını korur. Böylece tüm vektörlerin toplanması veya ortalamalarının alınması, tüm anlamsallığı koruyabilen bir vektör ile sonuçlanacaktır. Buna bakılarak, örneğin "transport + water" vektörlerini topladığımızda sonuç neredeyse "ship" veya "boat" kelimelerine eşittir. Bu, vektörlerin toplanmasının anlamsallığı topladığı anlamına gelir.

Şu cümleye baktığımızda:

The man went ___ for a walk.

Belli ki boşluğu "out" kelimesiyle doldurmak istiyoruz, fakat aynı zamanda "outside" kelimesini de kullanıyor olabiliriz. Word2Vec algoritmaları bu fikirden ilham almıştır. Yanındaki boşlukları doldururken bütün kelimeleri istersiniz, çünkü birbirlerine aittirler. Bu nedenle “out” ve “outside” kelimeleri birbirine yakınlaşırken, “bread” gibi bir sözcük daha uzakta olacaktır.

Paragraf vektörlerine gelince, Word2vec'teki ile tamamen aynı şeyi yaparlar, ancak toplamı kullanmak yerine, vektör kelimelerinin ortalamalarını alırlar ve paragraf vektörü ile birleştirirler.

Hem Doc2vec hem de Word2vec, genel bir metin bloğunu, Word2Vec'in bir kelimeyi vektöre dönüştürme biçimine benzer şekilde bir vektöre dönüştürür. Paragraf vektörleri geleneksel olarak metinde düzenlendiği için paragraflara atıfta bulunmaları gerekmez. Teorik olarak ifadelere, cümlelere, paragraflara ve hatta daha büyük metin bloklarına uygulanabilirler.

Paragraf vektörü yayınlanmadan önce, insanlar cümle vektörleri olarak ortalama sözcük vektörlerini kullanmışlardır.

Ayrıca, Doc2Vec'in (bu nedenle, Word2Vec) amaçlanan göreve bağlı olarak önceki bölümde belirtilen her iki kelime vektörü gösterim tekniği Skip-gram ve CBOW ile kullanılabileceğini not etmek çok önemlidir.

3.1.4 Metin İşleme

Yapay öğrenmenin kullanıldığı alanlardan biri olan Doğal Dil İşleme, kurallı yapıya sahip dillerin anlaşılır ve yeniden üretilebilir olması açısından işlenerek çözümlenmesini sağlar. Bu çözümlemenin insana getireceği kolaylıklar birçok konuyu kapsar. Örneğin, yazılı belgelerin otomatik çevirisinde, otomatik konuşma ve komut anlamada, metin özetlemede ya da bilgi sağlamada birçok kolaylık elde edilmektedir.

Doğal dil işleme ile bilgisayarların doğal dili öğrenmesi sağlanıyor ve bununla birlikte çeşitli uygulamalar geliştirilerek insanların hayatı kolaylaştırılıyor. Bu uygulamalara örnek olarak:

- Arama motorları
- Sanal asistanlar
- Spam tespiti

- Otomatik Çeviri
- Duygu Analizi

Birçok uygulama alanı olan doğal dil işleme temek olarak bakıldığında iki gruba ayrılmaktadır. Bunlar, metin işleme (text processing) ve ses işleme (speech processing).

Metin işleme; çeşitli yöntemler kullanarak işlenmemiş dokümanları sınıflandırıp anlamlı hale getirmek için analiz etme işlemidir. Bu çalışmada birçok metin işleme yöntemi kullanılmaktadır.

3.1.3.1 Metin Ön İşleme

Metin ön işleme, herhangi bir doğal dil işleme sisteminin önemli bir parçasıdır, çünkü bu aşamada bilgi alma ve makine çeviri sistemlerinde de olduğu gibi, tanımlanan karakterler, kelimeler ve cümleler, morfolojik analizörler ve part-of-speech etiketleri gibi analiz ve etiketleme bileşenleri yoluyla diğer tüm işlem aşamalarına iletilen temel ünitelerdir. Bu işlemler metin belgelerinin önceden işlendiği bir faaliyet topluluğudur. Metin verileri, sayı formatları, tarih formatları, edatlar, isimler gibi bazı özel biçimler içerdiği için bu özel biçimler ve belge içerisindeki en yaygın kelimeler ortadan kaldırılarak metin işleme kolaylaştırılabilir.

Belgelerdeki birçok kelime cümle içinde kelimeleri birleştirmek için kullanıldığı çok sık yinelenir, ancak esasen anlamsızdır. Gereksiz kelimelerin, metinsel belgelerin içeriğine katkı sağlamadığı yaygın olarak anlaşılmaktadır. Yüksek oluşum sıklıklarından dolayı belgelerin içeriğini anlamada engel teşkil etmektedir.

Gereksiz kelimelerin (Stop-words) kaldırılması; "and", "are", "this" gibi genel kelimelerin veri setinden çıkarılmasıdır. Bunlar çok sık kullanılan kelimelerdir. Belgelerin sınıflandırılmasında yararlı değildir. Bu yüzden kaldırılmaları gerekir. Bununla birlikte, bu tür gereksiz kelimeler listesinin geliştirilmesi metin kaynakları arasında zordur ve tutarsızdır. Bu işlem aynı zamanda metin verilerini azaltır ve sistem performansını iyileştirir.

Köklerine ayırma (Stemming); bir kelimenin değişken biçimlerini, kök ile ortak bir temsil haline getirme işlemidir. Örneğin, “presentation”, “presented”, “presenting” kelimelerinin tümü “present” olan ortak bir temsiliyete indirgenebilir. Köklerine

ayırma için farklı algoritmalar vardır. İngilizce için de etkili olduğu bilinen en yaygın algoritma, Porter algoritmasıdır.

Tablo 3.1: Porter Stemmer ile bir Stemming örneği.

Orjinal Kelime	Kök Kelime
connect	connect
connected	connect
connection	connect
connections	connect
connects	connect

Küçük harfe çevirme (Lowercasing); en basit ve en etkili metin ön işleme yöntemlerinden biridir. Çoğu doğal dil işleme sorunlarına uygulanabilir ve veri kümesinin çok büyük olmadığı durumlarda ve beklenen çıktının tutarlılığına önemli ölçüde yardımcı olmaktadır.

3.1.3.2 Vektör Uzayı Modeli

Vektör Uzay Modeli, bilgisayarlar tarafından anlaşılabilecek belgelerin resmi metin gösterimi için başarılı bir modeldir. Bu cebir tabanlı modelde, metin belgeleri, uygulamaya bağlı olarak, bir cümle veya bir kelime olabilen, bazı farklı terimlerin katı olan, boyutları olan vektörler şeklinde temsil edilmektedir. Bu güçlü temsil şekli, makinelerin vektör işlemlerini kullanarak bu belgeleri sorgulamasını sağlar. Matematiksel bir yapı biçiminde temsil edilen bu vektörlerin bir kümesi vektör uzayı olarak adlandırılır (Abdelwahab ve Elmaghraby, 2016).

3.1.3.3 Boyut Azaltma

Makine öğrenmesi sınıflandırma problemlerinde, genellikle son sınıflamanın yapıldığı temelde çok fazla faktör vardır. Bu faktörler temel olarak özellikler denilen

değişkenlerdir. Özelliklerin sayısı arttıkça, eğitim setini görselleştirmek ve üzerinde çalışmak zorlaşır. Bazen, bu özelliklerin çoğu ilişkilidir ve dolayısıyla gereksizdir. Bu, boyut azaltma algoritmalarının devreye girdiği yerdir. Boyut azaltma, bir dizi temel değişken elde ederek, incelenen rastgele değişken sayısını azaltma işlemidir. Yani, X boyutunda bir veri kümesine sahipsek, onu Y boyutlarının bir alt kümesine dönüştürebiliriz.

Bu tekniklere neden ihtiyacımız var?

- Boyutlar azalırsa, verileri saklamak için gereken alan da azalır.
- Verileri görselleştirmemize yardımcı olur. Verileri daha yüksek boyutlarda görselleştirmek çok zordur, bu nedenle alanımızı 2 boyuta veya 3 boyuta düşürmek, kalıpları daha net bir şekilde çizmemize ve gözlemlememize izin verebilir.
- Modelimizin hızını artırarak hesaplama süresi azaltılabilir.
- Bazı algoritmalar büyük boyutlara sahip olduğunda iyi performans göstermeyebilir. Bu yüzden algoritmanın yararlı olması için bu boyutların azaltılması gerekir.

3.1.4.3.1 Boyut Azaltma Bileşenleri

Boyut azaltmanın iki bileşeni; özellik çıkarma ve özellik seçimi vardır. Boyut azaltma yöntemleri iki ana başlık altında incelenir. Bunlar; doğrusal boyut azaltma ve doğrusal olmayan boyut azaltma yöntemleridir.

3.1.4.3.1.1 Özellik Çıkarma

İlk ham veri kümesinin işlenmesi için daha yönetilebilir gruplara indirgendiği boyut azaltma işlemidir. Bu büyük veri kümelerinin bir özelliği, işlenmesi için çok fazla bilgi işlem kaynağı gerektiren çok sayıda değişkendir. Özellik çıkarma, değişkenleri özelliklerde bir araya getiren ve/veya birleştiren, orijinal veri setini doğru ve tam olarak tanımlayan, işlenmesi gereken veri miktarını etkin bir şekilde azaltan yöntemlerin adıdır.

Özellik çıkarma işlemi, önemli veya ilgili bilgileri kaybetmeden işleme için gereken kaynak sayısını azaltmanız gerektiğinde yararlıdır. Özellik çıkarma, belirli bir analiz

için fazladan veri miktarını da azaltabilir. Ayrıca, verilerin ve makinenin değişken kombinasyonlar (özellikler) oluşturma çabalarının azaltılması, yapay öğrenme sürecinde öğrenme ve genelleme adımlarının hızını kolaylaştırır.

3.1.4.3.1.2 Özellik Seçimi

Makine öğrenimi ve istatistiklerinde, değişken seçimi, özellik seçimi veya değişken altküme seçimi olarak da bilinen özellik seçimi, model yapımında kullanılmak üzere ilgili özelliklerin bir alt kümesini (değişkenler, öngörücüler) seçme işlemidir. Özellik seçimi teknikleri dört nedenden dolayı kullanılır:

- Araştırmacılar/kullanıcılar tarafından modellerin yorumlanmalarını kolaylaştırmak için basitleştirilmesi (James, vd., 2013),
- Eğitim sürelerini kısaltma,
- Boyutsallığın zoruluklarını önlemek,
- Overfittingi azaltarak geliştirilmiş genelleme yapmak (Bermingham, vd., 2015) (resmi olarak, varyansın azaltılması (James, vd., 2013).

Bir özellik seçim tekniği kullanılırken temel öneme sahip veriler verilerin yedekli veya alakasız olan bazı özellikler içermesidir ve bu nedenle fazla bilgi kaybı olmadan kaldırılabilirler (Bermingham, vd., 2015). Gereksiz ve konu dışı kavramları birbirinden farklıdır, çünkü konu ile ilgili bir özellik, güçlü bir şekilde ilişkili olduğu ilgili başka bir özelliğin varlığında gereksiz olabilir (Guyon ve André, 2003).

Özellik seçme teknikleri, özellik çıkarımından ayırt edilmelidir. Özellik çıkarma, orijinal özelliklerin işlevlerinden yeni özellikler oluşturur, özellik seçimi ise özelliklerin bir alt kümesini döndürür. Özellik seçme teknikleri, birçok özelliğin bulunduğu alanlarda ve nispeten az sayıda örneklemde (veya veri noktalarında) kullanılır. Özellik seçimi binlerce özelliğin ve yüzlerce örneğin bulunduğu yazılı metinlerin ya da DNA verilerinin analizini içerir.

3.1.4.3.2 Doğrusal Boyut Azaltma Yöntemleri

En yaygın ve iyi bilinen boyut azaltma yöntemleri doğrusal dönüşümler uygulayanlardır ve en çok tercih edilen metotlardan biri Temel Bileşen Analizidir (Principal Component Analysis, PCA). Bu çalışmada da PCA kullanılarak boyut azaltma işlemi gerçekleştirilmiştir.

3.1.4.3.2.1 Principal Component Analysis (PCA)

PCA, mekanikte temel eksen teoreminin bir analoğu olarak Pearson (1901) tarafından icat edilmiştir. Daha sonra 1930'larda bağımsız bir şekilde geliştirilip, Hotelling (1933) tarafından isimlendirilmiştir.

PCA, büyük bir değişken kümesini, büyük kümedeki bilgilerin çoğunu içeren daha küçük bir öğeye dönüştürerek, genellikle büyük veri kümelerinin boyutunu azaltmak için kullanılan bir boyut azaltma yöntemidir.

Bir veri kümesinin değişken sayısını azaltmak doğal olarak doğruluk oranı için yapılır, ancak boyut azaltmadaki püf nokta basitlik için küçük bir doğruluk sağlamaktır. Çünkü, yapay öğrenme algoritmalarında daha küçük veri setlerinde, yabancı değişkenlerin işlenmesi gerekmeden verilerin analiz edilmesi çok daha kolay ve hızlıdır.

Özetlemek gerekirse, PCA fikri basittir ve mümkün olduğu kadar çok bilgiyi korurken, veri setindeki değişkenlerin sayısını azaltır.

Aşağıdaki adımları içerir:

- Verilerin kovaryans matrisini oluşturur.
- Bu matrisin özvektörlerini hesaplar.
- En büyük özdeğerlere karşılık gelen özvektörler, orijinal verinin büyük bir varyans oranını yeniden oluşturmak için kullanılır.

Dolayısıyla, daha az sayıda özvektör elde etmiş oluruz ve bu süreçte bazı veri kayıpları olmuş olabilir. Ancak, en önemli değişkenler kalan özvektörler tarafından korunmaktadır.

Bu teknik, orijinal veri kümesinden bazı özellikler çıkarır. Bu çıkarılan özellikler/değişkenler asıl bileşenler olarak adlandırılır. Bu tekniğin arkasındaki ana neden, yüksek boyutlu bir veri kümesinden düşük boyutlu bir dizi özellik çıkarmaktır. Bu yöntem 3 veya daha yüksek boyutlu verilerle uğraşırken daha faydalı olur.

PCA, gerçek özvektör tabanlı çok değişkenli analizlerin en basitidir. Çoğu zaman, işleyişi, verilerin iç yapısını, verilerdeki varyansı en iyi açıklayacak şekilde ortaya çıkarmak olarak düşünülebilir. Çok değişkenli bir veri kümesi yüksek boyutlu bir veri alanında bir koordinat kümesi olarak görselleştirilirse, PCA kullanıcıya daha

bilgilendirici bir bakış açısıyla bakıldığında bu nesnenin bir yansıması olan daha düşük boyutlu bir resim sağlayabilir. Bu, yalnızca ilk birkaç temel bileşen kullanılarak yapılır, böylece dönüştürülen verilerin boyutu azalır.

PCA'nın her adımında yapılanlar sırasıyla; standartlaştırma, kovaryans, özvektörler ve özdeğerler hesaplamasıdır.

Standartlaştırma adımının amacı, sürekli başlangıç değişkenlerinin aralığını standartlaştırarak her birinin analize eşit katkıda bulunmasını sağlamaktır.

Daha spesifik olarak, PCA'dan önce standartlaştırma yapmanın kritik olmasının nedeni, ikincisinin başlangıç değişkenlerinin varyanslarına karşı oldukça hassas olmasıdır. Diğer bir deyişle, ilk değişkenlerin aralıkları arasında büyük farklar varsa, daha büyük aralıkları olan değişkenler küçük aralıkları olanlara göre baskın çıkacaktır (Örneğin, 0 ile 100 arasında değişen bir değişken 0 ile 1 arasında değişen bir değişkene hakim olacaktır.), önyargılı sonuçlara yol açacaktır. Dolayısıyla, verileri karşılaştırılabilir ölçeklere dönüştürmek bu sorunu önleyebilir.

Matematiksel olarak, bu, ortalamanın çıkarılması ve her değişkenin her değeri için standart sapma ile bölünmesi ile yapılabilir.

$$z = \frac{değer - ortalama}{standart\ sapma} \tag{3.1}$$

Standartlaştırma yapıldıktan sonra, tüm değişkenler aynı aralığa dönüştürülür.

Kovaryans matrisi hesaplama adımın amacı, girdi veri setinin değişkenlerinin birbirleriyle ortalamadan farklılık gösterdiğini veya başka bir deyişle aralarında bir ilişki olup olmadığını anlamaktır. Çünkü bazen değişkenler, fazladan bilgi içerecek şekilde yüksek oranda ilişkilidir. Bu korelasyonları tanımlamak için kovaryans matrisini hesaplıyoruz.

Kovaryans matrisi, başlangıç değişkenlerinin tüm olası çiftleriyle ilişkili kovaryansları girmiş olan pxp simetrik bir matristir (burada p boyut sayısıdır). Örneğin, değişkenleri x, y ve z olan 3 boyutlu bir veri seti için, 3 x 3'lük kovaryans matrisi aşağıdaki gibidir.

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix} \quad (3.2)$$

Bir değişkenin kendi ile kovaryansına bakıldığında, varyansı $\big(Cov(a,a) = Var(a)\big)$ olduğundan, aslında esas köşegende her ilk değişkenin varyasyonlarına sahibiz. Ve kovaryans yer değiştirme özelliğine sahip olduğundan $\big(Cov(a,b) = Cov(b,a)\big)$, kovaryans matrisinin girişleri esas köşegene göre simetriktir, bu da üst ve alt üçgen bölümlerin eşit olduğu anlamına gelir.

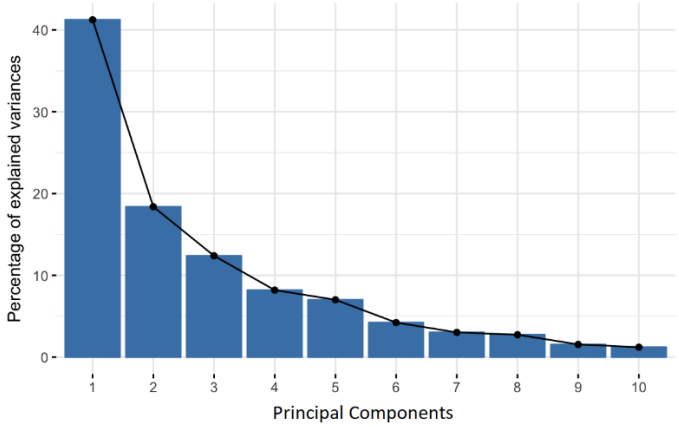
Matrisin girdileri olarak sahip olduğumuz kovaryanslar bize değişkenler arasındaki korelasyon hakkında neler söylüyor?

Burada asıl önemli olan kovaryansın işaretidir. Eğer işaret pozitifse iki değişken birlikte artar veya azalır yani birbirleriyle ilişkilidir. Eğer işaret negatifse biri azaldığında diğeri artar. Bu da korelasyon olmadığını gösterir.

Kısaca kovaryans matrisi tüm olası değişken çiftleri arasındaki korelasyonu özetleyen bir tablodur.

Temel bileşenleri tanımlamak için kovaryans matrisinin özvektörleri ve özdeğerleri hesaplanır. Özvektörler ve özdeğerler, verilerin temel bileşenlerini belirlemek için kovaryans matrisinden hesaplamamız gereken lineer cebir kavramlarıdır. Bu kavramların açıklamasına başlamadan önce, ilk olarak temel bileşenler ile ne demek istediğimizi anlayalım.

Temel bileşenler, doğrusal kombinasyonlar veya başlangıç değişkenlerin karışımları olarak oluşturulan yeni değişkenlerdir. Bu kombinasyonlar, yeni değişkenlerin ilişkisiz olduğu ve ilk değişkenler içindeki bilgilerin çoğunun birinci bileşenlere sıkıştırıldığı şekilde yapılmıştır. Burada 10 boyutlu veri 10 temel bileşen verir, ancak PCA, ilk bileşende kayda alınabilecek maksimum bilgiyi, daha sonra ikinci ve kalan bileşenlerdeki maksimum bilgiyi, aşağıdaki çizimde gösterildiği gibi elde etmeye çalışır.



Şekil 3.7: PCA tarafından azalan varyans yüzdesinin çizimi

Bilgileri temel bileşenlerde bu şekilde düzenlemek, fazla bilgiyi kaybetmeden boyutu azaltmamıza izin verir ve bu, düşük olan bileşenleri atarak kalan bileşenleri yeni değişkenler olarak kabul etmemizi sağlar.

Burada gerçekleştirilmesi gereken önemli bir nokta, temel bileşenlerin daha az yorumlanabilir olması ve bu bileşenler ilk değişkenlerin doğrusal kombinasyonları olarak oluşturulduğundan, hiçbir gerçek anlamı olmamasıdır.

Geometrik olarak konuşursak, temel bileşenler, maksimum miktarda varyansı açıklayan verinin yönünü, yani verilerin çoğu bilgisini alan çizgileri temsil eder. Buradaki varyans ve bilgi arasındaki ilişki, bir çizgi tarafından taşınan varyans ne kadar büyükse, onun içindeki veri noktalarının dağılımı o kadar büyüktür ve bir çizgi boyunca dağılım ne kadar büyükse, o kadar fazla bilgiye sahiptir. Tüm bunları basitçe söylemek gerekirse, temel bileşenleri yalnızca verileri görmek ve değerlendirmek için en iyi açıyı sağlayan yeni eksenler olarak düşünün, böylece gözlemler arasındaki farklar daha iyi görülebilir.

Özvektörler ve özdeğerler her zaman çiftler halinde gelmeleridir, böylece her özvektörün bir özdeğeri vardır. Ve sayıları verinin boyut sayısına eşittir. Örneğin, 3 boyutlu bir veri seti için 3 değişken vardır, dolayısıyla 3 özdeğere karşılık gelen 3 özvektör vardır.

Yukarıda açıklanan tüm bilginin arkasında özvektörler ve özdeğerler vardır, çünkü kovaryans matrisinin özvektörleri aslında en fazla varyansın olduğu (en fazla bilgi) ve Temel Bileşenler dediğimiz eksenlerin yönleridir. Ve özdeğerler, basit bir şekilde, her

Temel Bileşende taşınan varyans miktarını veren özvektörlere bağlı katsayılardır. Özvektörleri, özdeğerleri sırasına göre sıralayak, en yüksekten en düşüğe doğru önem sırasına göre temel bileşenler elde edilir.

Örneğin; veri setimizin x, y değişkenli yani 2 boyutlu olduğunu, kovaryans matrisinin özvektörlerinin ve özdeğerlerinin aşağıdaki gibi olduğunu varsayalım:

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$
$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323 \tag{3.3}$$

Özdeğerleri azalan sıralamada sıralarsak, $\lambda_1 > \lambda_2$ elde ederiz. Bu, birinci temel bileşene (PC1) karşılık gelen özvektörün v1 ve ikinci bileşene (PC2) karşılık gelen özvektörün λ_2 olduğu anlamına gelir.

Temel bileşenlere sahip olduktan sonra, her bir bileşenin hesaba kattığı varyans yüzdesini hesaplamak için her bileşenin özdeğerini, özdeğerlerin toplamına böleriz. Bunu yukarıdaki örneğe uygularsak, PC1 ve PC2'nin varyansını sırasıyla %96 ve %4 olarak elde ederiz.

Önceki adımda gördüğümüz gibi, özvektörleri hesaplamak ve özdeğerlerini azalan düzende sıralamak, temel bileşenleri anlamlılık sırasına göre bulmamıza izin verir. Bu adımda, yaptığımız şey, tüm bu bileşenlerin saklanıp saklanmayacağını veya daha az anlamlı olanları (düşük özdeğerlere sahip olanları) atıp atmamaya karar vermek ve geri kalanları ile özellik vektörü olarak adlandırdığımız bir vektör matrisi oluşturmaktır.

Bu yüzden, özellik vektörü, sadece tutmaya karar verdiğimiz bileşenlerin özvektörlerini sütun olarak içeren bir matristir. Bu boyut azaltmaya doğru atılan ilk adımdır, çünkü sadece p özvektörlerini (bileşenlerini) n'den uzak tutmayı seçersek, son veri setinde sadece p boyutları olacaktır.

Önceki adımdaki örneğe devam edersek, hem v_1 hem de v_2 özvektörleriyle birlikte bir özellik vektörü oluşturabiliriz:

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix} \tag{3.4}$$

Veya daha az önemli olan özvektör v_2'yi atıp, sadece v_1 ile bir özellik vektörü oluşturabiliriz:

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \quad (3.5)$$

Özvektör v_2'nin atılması, boyutu 1 oranında azaltır ve sonuçta nihai veri setinde bilgi kaybına neden olur. Ancak v_2'nin bilgilerin yalnızca %4'ünü taşıdığı göz önüne alındığında, kayıp bu nedenle önemli olmayacak ve v_1 tarafından taşınan bilgilerin %96'sına sahip olacağız.

Dolayısıyla, örnekte gördüğümüz gibi, aradığımız şeye bağlı olarak tüm bileşenleri koruyabilir ya da daha az önemli olanları atabiliriz. Çünkü verileri sadece boyut azaltmaya çalışmaksızın ilişkisiz olan yeni değişkenler (temel bileşenler) olarak tanımlamak istiyorsak, daha az önemli bileşenleri dışarıda bırakmak gerekli değildir.

Önceki adımlarda, standartlaştırma dışında verilerde herhangi bir değişiklik yapılmaz, sadece temel bileşenler seçilir ve özellik vektörü oluşturulur, ancak girdi veri seti her zaman orijinal eksende kalır (yani, ilk değişkenler).

Son adımda amaç, verileri temel bileşenler tarafından orijinal eksenlerden temsil edilen bileşenlere yeniden yönlendirmek için kovaryans matrisinin özvektörleri ile oluşturulan özellik vektörünü kullanmaktır. Temel Bileşen Analizi ismi de buradan gelmektedir. Bu veri seti, ayarlanan standart orijinal verilerin transpozesi ile özellik vektörünün transpozesi çarpılarak elde edilir.

$$SonVeriSeti = \text{Özellik Vektörü}^T \times StandartOrijinalVeriSeti^T \quad (3.6)$$

3.1.4.3.3 Doğrusal Olmayan Boyut Azaltma Yöntemleri

Doğrusal olmayan dönüşüm yöntemleri, veriler doğrusal bir alt alana dayanmadığında kullanılır. Yüksek boyutlu bir yapıda, en alakalı bilgilerin az sayıda düşük boyutlu manifoldlarda yoğunlaştığını söyleyen manifold hipotezine dayanmaktadır. Doğrusal bir alt alan düz bir kağıtsa, yuvarlanmış bir kağıt, doğrusal olmayan bir manifoldun

basit bir örneğidir. Doğrusal olmayan boyut azaltma yöntemlerinden biri t-Dağıtılmış Stokastik Komşu Gömme (t-SNE) yöntemidir. Özellikle yüksek boyutlu veri kümelerinin görüntülenmesi için çok uygun olan boyut azaltma tekniğidir.

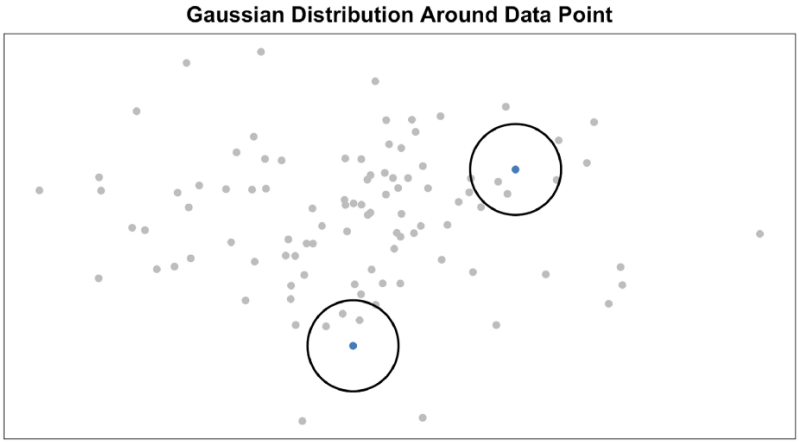
3.1.4.3.2.1 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Dağıtık Stokastik (Rasgele) Komşu Gömme (t-SNE); Öncelikle veri araştırması ve yüksek boyutlu verilerin görselleştirilmesi için kullanılan denetimsiz, doğrusal olmayan bir tekniktir. Yüksek boyutlu uzayda bulunan veri noktalarının çiftlerinin ilişkili olma ihtimalini hesaplar ve daha sonra benzer bir dağılım üreten düşük boyutlu bir gömme seçer. Daha basit bir ifadeyle, t-SNE size verilerin yüksek boyutlu bir alanda nasıl düzenlendiğine dair bir his veya sezgi verir. 2008 yılında Laurens van der Maaten ve Geoffrey Hinton tarafından geliştirilmiştir (Maaten ve Hinton, 2008).

t-SNE, bilgisayar güvenliği araştırması (Gashi, Stankovic ve Leita, 2009), müzik analizi (Hamel ve Eck, 2010), kanser araştırması (Jamieson, vd., 2010), biyoinformatik (Wallach ve Liliean, 2009) ve biyomedikal sinyal işleme gibi birçok uygulamada görselleştirme amacıyla kullanılmıştır (Birjandtalab, Pouyan, Nourani, 2016). Yapay bir sinir ağı tarafından öğrenilen üst düzey sunumları görselleştirmek için de sıklıkla kullanılır.

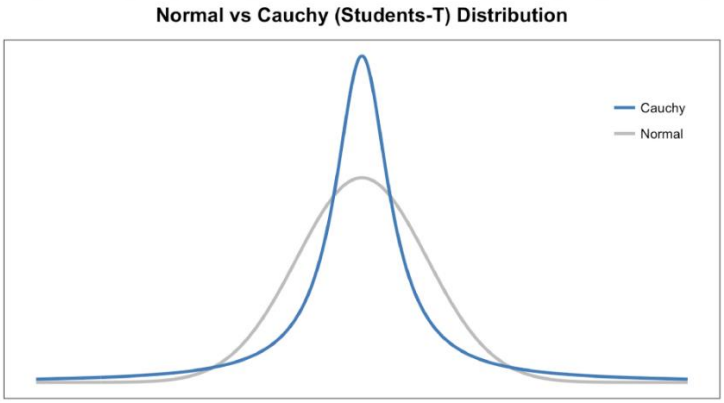
T-SNE algoritması, yüksek ve düşük boyutlu alandaki örnek çiftleri arasındaki benzerlik ölçüsünü hesaplar. Daha sonra bir maliyet işlevi kullanarak bu iki benzerlik önlemini optimize etmeye çalışır. Bu 3 temel adımda yapılır.

1. adımda, yüksek boyutlu uzayda noktalar arasındaki benzerlikler ölçülür. 2 boyutlu uzayda dağılmış bir sürü veri noktasını düşünün (Şekil 3.8). Her veri noktası (xi) için bu nokta üzerinde bir Gauss dağılımı merkezlenir. Sonra bu Gauss dağılımındaki tüm noktaların (xj) yoğunluğu ölçülür. Daha sonra ise tüm noktalar yeniden normalize edilir. Bu bize tüm noktalar için bir dizi olasılık (Pij) verir. Bu olasılıklar benzerliklerle orantılıdır. Bunun anlamı, eğer x1 ve x2 veri noktaları bu gauss çemberinin altında eşit değerlere sahipse o zaman bunların oranları ve benzerlikleri aynıdır ve dolayısıyla bu yüksek boyutlu alanın yapısında yerel benzerlikler vardır (Maaten ve Hinton, 2008).



Şekil 3.8: Yüksek boyutlu uzaydaki ikili benzerliklerin ölçülmesi

2. adım da 1'e benzerdir, ancak bir Gauss dağılımı kullanmak yerine, aynı zamanda Cauchy dağılımı olarak da bilinen bir serbestlik derecesine sahip bir Student t-dağılımı kullanılır (Şekil 3.9). Bu bize düşük boyutlu uzayda ikinci bir olasılıklar kümesi (Qij) verir. Görüldüğü gibi, Öğrenci t-dağılımı normal dağılımdan daha ağır kuyruklara sahiptir. Ağır kuyruklar, uzak mesafelerin daha iyi modellenmesine izin verir.



Şekil 3.9: Normal ve Öğrenci t-Dağılımın Karşılaştırılması

3. son adımda, bu olasılık kümesinin düşük boyutlu alandan (Qij) yüksek boyutlu alanın (Pij) mümkün olan en iyi şekilde yansıtması istenir. İki harita yapısının benzer olmasını isteriz. İki boyutlu uzayların olasılık dağılımları arasındaki fark, Kullback-Liebler sapma (KL) kullanarak ölçülür. KL, Pij ve Qij değerlerini etkili bir şekilde karşılaştıran asimetrik bir yaklaşımdır.

Laurens van der Maaten, birçok örnek göstermektedir. İklim araştırması, bilgisayar güvenliği, biyoinformatik, kanser araştırması vb. alanlarda t-SNE kullanımından bahsedilir.

3.1.4.3.3.2 PCA ve t-SNE

Unutulmaması gereken ilk şey, PCA'nın 1933'te geliştirildiği ve t-SNE'nin 2008'de geliştirildiğidir. 1933'ten bu yana veri bilimi dünyasında, özellikle bilgi işlem ve veri boyutunda çok şey değişmiştir. İkincisi, PCA, varyansı maksimuma çıkarmak ve büyük çift yönlü mesafeleri korumak için doğrusal boyut küçültme tekniğidir. Başka bir deyişle, farklı olan şeyler birbirinden uzaklaşır. Bu, özellikle doğrusal olmayan manifold yapılarıyla uğraşırken zayıf görselleştirmeye yol açabilir.

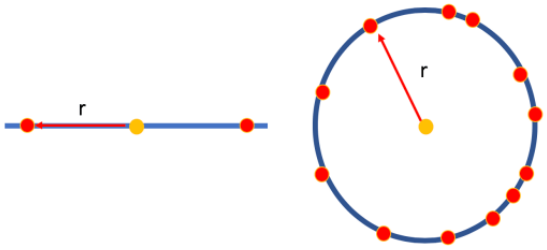
Boyut azaltma söz konusu olduğunda, en ünlü ve yaygın olarak kullanılan algoritmalardan biri PCA'dır. Bunun sebebi, PCA hızlıdır ve kullanımı basittir. Temel olarak, PCA veri kümesinin genel varyansını en iyi koruyan düşük boyutlu gömmeler oluşturur. Ancak PCA'nın yanı sıra t-SNE yöntemine de ihtiyaç duyarız. Çünkü, PCA iyi bir boyut azaltma yöntemi olsa da bazı dezavantajları vardır. Bunlardan biri, doğrusal bir izdüşüme sahiptir, yani doğrusal olmayan bağımsızlıkları yakalayamaz. Örneğin, PCA aşağıdaki yapıyı açamaz (Şekil 3.10).

Şekil 3.10: PCA'nın başarısız olacağı klasik bir örnek Swiss rulosudur

Çünkü doğrusal bir izdüşüm temelde gölge düşürmeye benzer. Açmamıza izin verecek olan bu İsviçre rulosuna bakabileceğimiz hiçbir yön bulunmaz. PCA'nın aksine, t-

SNE, doğrusal izdüşümlerle sınırlı değildir, bu da onu her türlü veri kümesine uygun hale getirir.

Temel olarak t-SNE yönteminin altında yatan yaklaşım, noktalar arasındaki mesafelerin neredeyse aynı kalması için yüksek boyutlu uzayda noktaları daha düşük bir boyuta eşlemeye çalışmaktır. T-SNE pratikte iyi çalışır. Bunun birçok sebebi vardır. En önemlilerinden biri kalabalık sorununu çok iyi çözebilmesidir. Kalabalık sorunu temel olarak boyutsallığın lanetinden gelmektedir. Yüksek boyutlu uzayda, kürenin yüzeyi yarıçapı ile düşük boyutlu uzayda bir küreye kıyasla çok daha hızlı büyür. Bu, yüksek boyutlu uzayların belirli bir noktadan orta mesafede birçok sayıda noktaya sahip olabileceği anlamına gelir.



Şekil 3.11: Daha yüksek boyutlu uzayların orta mesafede nasıl daha fazla noktaya sahip olduğunu gösteren bir örnek

Bu noktalar daha düşük boyutlu bir alana eşlendiğinde çok sayıda orta mesafe noktası, alt boyutlu alanda orta mesafeden toplanmaya çalışacaktır. Sorun şu ki, orta mesafedeki alt boyutlu alanda çok daha az yer vardır. Böylece tüm noktalar alt boyutta " ezilir " ve kalabalıklaşır. t-SNE, kalabalıklaşmayı önlemek için optimizasyonu orta mesafeli noktalara " yayarak " yapar.

t-SNE'nin bir diğer önemli yararı da "stokastik komşular" kullanmasıdır. Bu, noktaların diğer noktaların komşuları arasında net bir çizgi olmadığı anlamına gelir. Bu açık sınırların olmaması büyük bir avantaj olabilir, çünkü t-SNE'nin hem küresel hem de yerel yapıyı doğal olarak dikkate almasına izin verir. Yerel yapı, küresel yapıdan daha önemlidir, ancak uzaktaki noktalar tamamen göz ardı edilmez ve "iyi dengelenmiş" boyutsal bir azalmaya izin verir.

3.1.5 Sınıflandırma

Sınıflandırma, verilen veri noktalarının sınıfını tahmin etme işlemidir. Sınıflar bazen hedef / etiket veya kategori olarak adlandırılır. Örneğin, e-posta servis sağlayıcılarında spam algılama bir sınıflandırma sorunu olarak tanımlanabilir. Spam ya da spam değil, 2 sınıf olduğundan bu ikili sınıflandırmadır. Sınıflandırıcı, verilen girdi değişkenlerinin sınıfla nasıl bir ilişki içinde olduğunu anlamak için bazı eğitim verilerini kullanır. Bu durumda, eğitim verisi olarak bilinen spam ve spam olmayan e-postaların kullanılması gerekir. Sınıflandırıcı doğru bir şekilde eğitildiğinde, bilinmeyen bir e-postayı tespit etmek için kullanılabilir.

Sınıflandırma, hedeflerin girdi verileriyle de sağlandığı gözetimli öğrenme kategorisine aittir. Kredi onayı, tıbbi teşhis, hedef pazarlama vb. gibi alanlarda birçok uygulama vardır.

Tembel öğrenenler (Lazy learner) ve istekli öğrenenler (Eager learners) olarak sınıflandırmada iki tür öğrenim vardır. Lazy Learner sadece eğitim verilerini saklar ve bir test verisi görünene kadar bekler. Bu olduğunda, sınıflandırma, saklanan eğitim verilerindeki en ilgili verilere dayanarak gerçekleştirilir. Eager learners'a kıyasla, lazy learner'ların eğitim süreleri daha az, ancak tahminleri daha fazladır. Örnek; k-En Yakın Komşu, Durum Temelli Akıl Yürütme.

Eager Learners, sınıflandırma için veri almadan önce verilen eğitim verilerine dayalı bir sınıflandırma modeli oluşturur. Tüm örnek alanını kapsayan tek bir hipotez uygulayabilmelidir. Model yapımı nedeniyle, Eager Learners'ta öğrenme süresi uzun sürer ancak tahmin süresi daha az zaman alır.

Örnek; Karar Ağacı, Naive Bayes, Yapay Sinir Ağları.

3.1.5.1 Gauss Naive Bayes

Bayes teoremi olasılık teorisinde incelenen önemli bir konudur. Bu teorem, olasılık dağılımındaki rastgele değişken için koşullu olasılıklar ile marjinal olasılıklar arasındaki ilişkiyi gösterir. Bu formla, Bayes teoremi tüm istatistikçiler tarafından kabul edilen bir ilişkiyi açıklar. Naive Bayes (NB), hedef değişken ile bağımsız değişkenler arasındaki ilişkiyi analiz eden tahminci ve tanımlayıcı bir sınıflandırma algoritmasıdır (Ceci, 2005).

Olasılık teorisinde incelenen bir olay olarak, koşullu bir olayın (yani, B olayı biliniyorsa A olayı) olasılık değeri, olay B'nin olasılık değerinden (A olayı bilindiğinde B olayı) koşullu olarak farklıdır. Bununla birlikte, bu iki farklı durum arasında özel bir ilişki vardır ve bu ilişkiye Bayes Teoremi denir (ilk olarak İngiliz istatistikçi Thomas Bayes 1702-1761 tarafından açıklanmıştır) (Stigler, 1982).

Son yıllarda, veri analizindeki artış nedeniyle Bayes yaklaşımının kullanımı artmaktadır. Thomas Bayes'in 1763'te ortaya koyduğu Bayes Teoremi bu yöntemin temelini oluşturmaktadır. Jeffrey's de Finette, Savage ve Lindley gibi birçok araştırmacı Bayes analizinin geliştirildiğini keşfetti. Ancak, öznellikçi modele göre, olasılık gözlemcinin öznel belirsizliğidir. Bu nedenle olasılık değeri özneldir ve yeni kanıtların çalışmanın temel taşı olarak kabul edildiğinde değiştirilebileceğine inanılmaktadır (Jatana ve Sharma, 2014).

Özellikler arasındaki bazı ilişkilendirmeler ve bağımlılıklar gösterilemese de, Bayes kararı birçok sınıflandırma probleminde oldukça tatmin edici olduğu kanıtlanmıştır. Naive Bayes, model öğrenimi sırasında her bir çıktının öğrenme setinde kaç kez göründüğünü hesaplar. Bu değere önceki olasılık denir. Örneğin, bir banka kredi kartı uygulamalarını risk sınıflarının "iyi" ve "kötü" olarak gruplandırmak istemektedir. Çıktı toplam 5 durumda 2 kez gelirse, risk açısından iyi olma olasılığı 0,4'tür. Bu nedenle, kredi kartı için başvuran kişi hakkında hiçbir şey bilinmiyorsa, o kişi risk sınıfında iyi olma olasılığı 0.4'tür, Naive Bayes aynı zamanda her bağımsız / bağımlı değişken, değişkenlerin kombinasyonunun sıklığını bulur. Bu frekanslar önceden bir araya gelerek tahminde kullanılmıştır (Ceci, 2005).

Belirsizlik, insan yaşamında çok yer kapladı ve bu belirsizlik, geçmiş ya da gerçekleşmemiş koşullara bağlı olarak bilgi eksikliğidir. Günlük yaşamda, entelektüel ve sezgilerin kullanımı bilimsel amaçlar için kullanılabilir. Bayesci yaklaşımın bu tür olaylarına ilişkin bilimsel çalışmalar, alternatif bir olasılık kullanımı olarak ortaya çıkmaktadır. Naive Bayes sınıflandırma algoritmasını sınıflandırma ana fikri genellikle sonsal olasılıkları hesaplamak için kullanılan ve iki rastgele olayın koşullu olasılıklarını ilişkilendiren bir hipotezdir. Sonsal olasılık prensibine dayanan bir teoremdir (Panda ve Patra, 2007). Oluşturulacak en basit sınıflandırma modellerinden biri, değişkenler arasındaki bağımsızlık varsayımını kullanan Naive Bayes sınıflandırma algoritmasıdır, bu nedenle özelliklerin olasılığının başka bir niteliğin olasılığını etkilemediğini tanımlar. Naive Bayes sınıflandırması, sınıfın en yüksek

olasılığa sahip örneklerini öngören olasılık yöntemlerinden biridir; Naive Bayes'te artımlı olarak adlandırılan bir çevrimiçi öğrenme durumu vardır ve her eğitim örneği bir hipotezin doğru olma olasılığını artırır veya azaltır. Ön bilgiler gözlemlenen verilerle birleştirilebilir (Mukherjee ve Sharma, 2012).

Naive Bayes yaklaşımı, olasılıkları hesaplamak için kullanılan iki rastgele olgunun koşullu olasılıklarıdır. Maksimum olasılık ilkesine dayanan bir teoremdir. Bu durumda, Bayes Teoremi, rasgele bir olay elde etmek için A ve B olasılıklarının doğruluğunu hesaplamak için kullanılabilir Naive Bayes sınıflandırması, sınıfın en yüksek olasılığa sahip örneklerini öngören olasılık yöntemlerinden biridir ve akış olarak çalışır. A ve B rastgele olaylar olsun.

$$P(A|B) = \frac{P(B|A)P(B)}{P(A)} \tag{3.7}$$

Burada, $P(A)$ ve $P(B)$ olayın birbirinden bağımsız olma olasılığıdır. $P(A|B)$, B koşulunda A'nın, $P(B|A)$ ise A olayının doğru olması koşulunda B'nin olma olasılığıdır.

Naive Bayes sınıflandırması, sınıfın en yüksek olasılığa sahip örneklerini tahmin eden olasılık yöntemlerinden biridir ve akış olarak çalışır. A ve B'nin rastgele olaylar olduğunu varsayın.

$$P(A|D) = \frac{P(D|A)P(D)}{P(A)} \tag{3.8}$$

Naive Bayes formülünde, n x m vektörü olarak temsil edilen veri kümesi, burada n öznitelikler ve m sınıf sayısıdır. D = $D = \{d_1, d_2, .., d_n\}$ sınıflandırılması gereken belirli bir veridir. Yöntem, D'nin D koşullandırılmış maksimum posterior olasılığa sahip olan A_k sınıfına gittiğini hesaplar.

$$P(A_k|D) = \frac{P(D|A_k)P(A_k)}{P(D)} \tag{3.9}$$

Bu sadece sınıfın maksimum olasılığını elde etmek içindir. Sınıfı en önemli hale getirmek için, önceki olasılık A_k sınıfındaki tüm eğitim verilerinin, eğitim veri setinin toplamına bölünmesidir. A_k sınıfı olasılığı bilinmiyorsa, bu sınıfların olasılıklarının eşit olduğu düşünülü. Eğitim verileri gerekli olasılığı genellikle elde etmek için kullanılır.

Özetle, Bayes'e dayanan Naive Bayes algoritması eğitim verilerini kullanarak her bir sınıf için etiket değerine bakarak olasılığını hesaplar. Daha sonradan etiketlenmemiş veriler için özelliklerde kullanıldığında sınıf tahmini yaparken daha önce gözlenmiş olasılıklardan yararlanır. Birçok algoritmada güçlü olmayan etkiler göz ardı edilse de, Bayes ile yapılan tahminleri değiştirirken tüm kanıtlar dikkate alınır. Çünkü tek başına bir özelliğin çok fazla etkisi olmasa da bir arada kullanımında etkisi artabilir. Naive Bayes algoritmasında da veri kümesinde bulunun özelliklerin hepsi önemli olarak varsayılır. Bu algoritma çok yönlü ve doğru sonuçlar vermesi nedeniyle sınıflandırıcının güçlü olduğu kabul edilir (Lantz, 2013). Temel olarak, Naive Bayes algoritmasında 3 farklı model bulunur. Bunlar; Multinomial Naive Bayes (MNB), Bernoulli Naive Bayes (BNB) ve Gaussian Naive Bayes (GNB)'tir. MNB, sınıflandırma yaparken genellikle kategorik bir verinin sıklığına bakar. MNM sınıflandırma ile BNB sınıflandırma benzemektedir ancak BNB algoritmasında boolean değişkenlerle tahmin yapılır (evet-hayır gibi). GNB ise, kategorik verinin yanı sıra sayısal verinin de gauss dağılımı ile sınıflandırılmasını imkan sunar ve bu çalışmada kullanılmasının tercih sebebi de budur.

Gauss (Normal dağılım) ile çalışmak en kolayıdır, çünkü eğitim verilerinden yalnızca ortalama ve standart sapmayı tahmin etmek gerekir. Her sınıf için girdi değerlerinin (x) ortalama ve standart sapmasını hesaplayabiliriz.

$$ortalama\ (\mu) = \frac{\sum x_i}{N} \tag{3.10}$$

Burada N, örnek sayısı ve x_i, eğitim verisindeki her bir giriş değişkeni için değerdir.

$$standart\ sapma\ (\sigma) = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}} \tag{3.11}$$

Burada N, örnek sayısıdır, x_i i'inci örnektir ve μ ortalama değeridir. Her bir örneğin ortalamadan farkının karesi alınıp toplanır. Daha sonra toplam örnek sayısına bölünür. Bunun da karekökü alınarak standart sapma elde edilmiş olur.

Tahminler yaparken bu parametreler Gauss Olasılık Yoğunluk Fonksiyonuna (Gaussian Probability Density Function) değişken için yeni bir girişle eklenebilir ve karşılığında, söz konusu sınıf için bu yeni giriş değerinin olasılığı hakkında bir tahmin sağlanır.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{3.12}$$

$f(x)$ Gauss Olasılık Yoğunluk Fonksiyonudur. Buradaki μ ve σ yukarıda hesapladığımız ortalama ve standart sapmadır. π ise sayısal sabittir, e sayısal sabit veya Euler'in güce yükseltilmiş sayısıdır ve x, giriş değişkeni için giriş değeridir.

3.1.5.2 Random Forest

Rastgele Orman (RF), birbiriyle ilişkili olmayan birkaç karar ağacından oluşan bir sınıflandırma yöntemidir. Tüm karar ağaçları öğrenme süreci boyunca belirli bir tür rastgelelik altında büyümüştür. Bir sınıflandırma işlemi için, o ormandaki her ağaç karar verebilir ve yüksek oyu olan sınıf nihai sınıflandırmaya karar vermektedir.

Rastgele Orman terimi 1999 yılında Leo Breiman tarafından belirlenmiştir (Breiman, 2001). Breiman, karar ağaçlarının çeşitli yöntemlerini, örneğin torbalama veya artırma yollarını araştırmıştır.

Rastgele Ormanın bazı özellikleri bulunmaktadır.

Sınıflandırıcı çok hızlı eğitim yapar: Bu avantaj, tek bir karar ağacının kısa eğitim veya kurulum süresinden ve rastgele bir ormanın eğitim süresinin ağaç sayısıyla doğrusal olarak artmasından kaynaklanır.

Bir test örneğinin değerlendirilmesi, her ağaçta ayrı ayrı gerçekleşir, bu nedenle çok hızlı değerlendirilir.

Büyük ölçekli veriler için çok etkilidir (birçok sınıf, birçok eğitim örneği, birçok özellik).

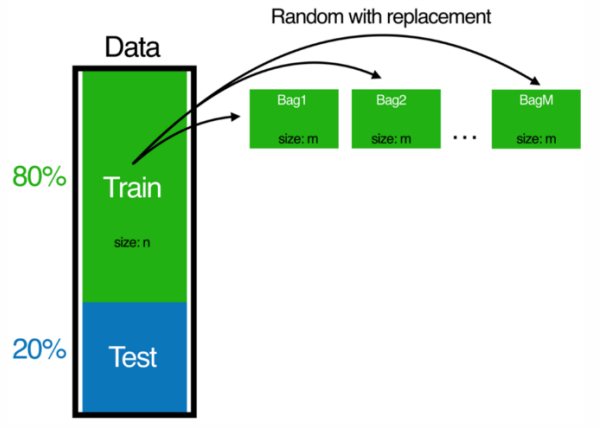
Farklı seçenekleri anlamak ve kullanmak için, nasıl hesaplanacağı hakkında daha fazla bilgi edinmek bize yardımcı olacaktır. Çoğu seçenek rastgele ormanlar tarafından oluşturulan iki veri nesnesine bağlıdır.

Rastgele ormanlar öğrenmeden önce, torbalamayı (bagging) anlamamız gerekir. Torbalama basit ve çok güçlü bir topluluk yöntemidir. Modelimizin varyansını azaltmak için kullanılabilecek genel bir prosedürdür. Daha yüksek bir varyans, modelinizin gereğinden fazla takıldığı anlamına gelir. Karar ağaçları gibi bazı algoritmalar genellikle yüksek varyanstan muzdariptir. Başka bir deyişle, karar ağaçları eğitildikleri verilere son derece duyarlıdır. Temel veriler biraz değiştirilirse, sonuçta ortaya çıkan karar ağacı çok farklı olabilir ve sonuç olarak modelimizin tahminleri büyük ölçüde değişecektir. Torbalama, yüksek sapma sorununa bir çözüm sunar. Ortalama birkaç karar ağacı alarak aşırı duyarlılığı sistematik olarak azaltabilir. Torbalama, bootstrap örneklemesi kullanır ve son olarak nihai tahminleri elde etmek için ortalamaları toplayarak bireysel modelleri toplar. Bootstrap örneklemesi, eğitim veri kümesinden rastgele değiştirilen örnekleme satırları anlamına gelir.

Bu nedenle torbalama ile, birden fazla kez tek bir eğitim örneği çizmek mümkündür. Bu, bazı satırların birden çok kez temsil edildiği ve bazılarının olmadığı eğitim setinin değiştirilmiş bir versiyonuyla sonuçlanır. Bu, başladığımız verilere benzer yeni veriler oluşturmamıza da olanak tanır. Bunu yaparak, birçok farklı ama benzer modele uydurabiliriz.

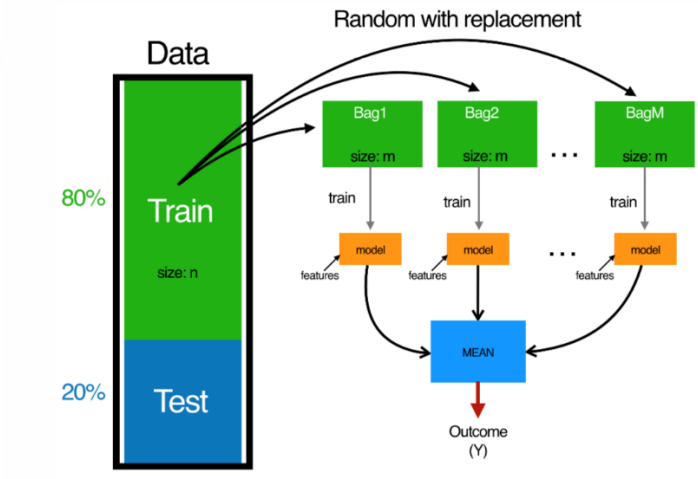
Torbalamanın çalışma şekli aşağıdaki gibidir:

Adım 1: Örnekler orijinal veri kümesinden değiştirilerek çizilir; burada örnekler, eğitim setindeki toplam örnek sayısı olan n'den küçük veya ona eşit bir sayıdır.



Şekil 3.12: Torbalamayı Anlama 1. Adım

Adım 2: Yeni oluşturulan bootstrap örnekleri üzerinde karar ağaçlar eğitilir. Adım1 ve Adım2 istenilen sayıda tekrarlanır. Genellikle ağaç sayısı arttıkça model daha iyidir. Ama çok sayıda ağaç, modeli karmaşık hale getirebilir ve sonuç olarak model ilk etapta var olmayan verilerdeki ilişkileri görmeye başladığında overfittinge yol açabilir.

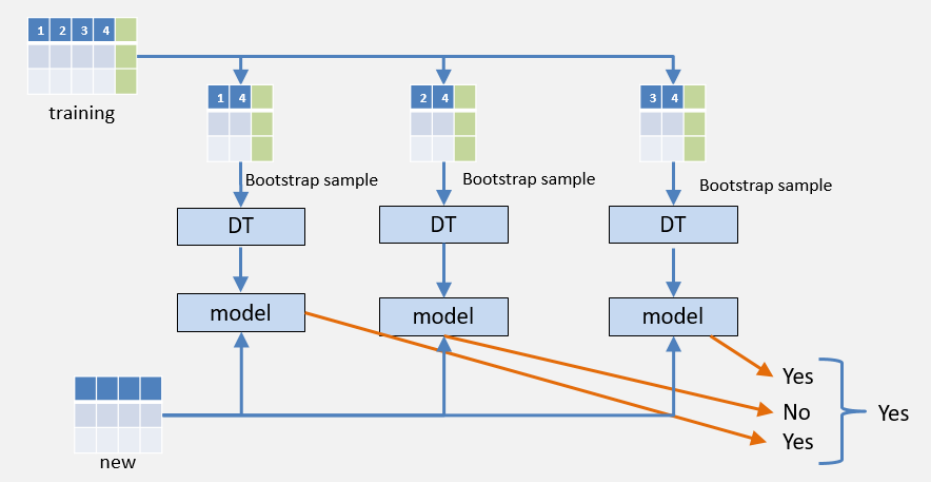


Şekil 3.13: Torbalamayı Anlama 2. Adım

Torbalı ağaçlar yaklaşımını kullanarak bir tahmin oluşturmak için, karar ağaçlarının her birinden bir tahmin oluşturmak ve ardından son bir tahmin elde etmek için tahminlerin ortalamasını almak gerekir. Torbalı veya grup tahmini, örneklenmiş overfitting ağaçların ortalama tahminidir. Torbalı ağaçlar modeli konseye çok benzer. Genellikle, bir konseyin karar alması gerektiğinde, oy çokluğu ile alınır. Daha fazla oy

alan seçenek (A seçeneği 100 oy ve B seçeneği 90 oy aldı), sonuçta konseyin nihai kararıdır. Benzer şekilde, torbalamada, bir sınıflandırma problemini çözmeye çalıştığımızda, temel olarak tüm karar ağaçlarında çoğunluk oyu alınır. Ve gerileme durumunda, tüm karar ağacı tahminlerimizin ortalamasını alırız. Çeşitli karar ağaçlarının kolektif bilgisi, tipik olarak herhangi bir ağacın bilgisini yener. Torbalı ağaçlar bu nedenle daha iyi tahmin performansı sunar.

Rastgele orman, torbalamadan sadece bir şekilde farklıdır. Öğrenme sürecinin her bir bölümünde, özelliklerin rastgele bir alt kümesini inceleyen değiştirilmiş bir ağaç öğrenme algoritması kullanır. Ağaçlar arasındaki korelasyondan kaçınmak için bunu yapar. Varsayalım ki, diğer bazı orta derecede güçlü tahmin edicilerle birlikte veri setinde çok güçlü bir tahmin edici var, o zaman torbalanmış ağaçların toplanmasında, karar ağaçlarının çoğu veya tamamı ilk bölüm için çok güçlü tahmin ediciyi kullanacak. Torbalı tüm ağaçlar benzer görünecektir. Dolayısıyla, torbalanmış ağaçlardan yapılan tüm tahminler birbiriyle oldukça ilişkili olacaktır. İlişkili tahmin ediciler, tahmin doğruluğunun iyileştirilmesine yardımcı olamaz. Rastgele bir özellik alt kümesi alarak Rastgele Ormanlar sistematik olarak korelasyondan kaçınır ve modelin performansını artırır. Aşağıdaki örnek Rastgele Orman algoritmasının nasıl çalıştığını gösterir.



Şekil 3.14: Rastgele Ormanları Anlama

Bir sınıflandırma sorununu çözmeye çalıştığımız bir duruma bakalım. Yukarıdaki görüntüden de anlaşılacağı gibi, eğitim verilerimiz dört özelliğe sahiptir: Özellik1,

Özellik 2, Özellik 3 ve Özellik 4. Örneğin, Karar Ağacı (Decision Tree: DT) 1, özellikler 1 ve 4 üzerinde eğitilecektir. DT2, özellikler 2 ve 4 üzerinde eğitilecek ve son olarak DT3, özellikler 3 ve 4 üzerinde eğitilecektir. Bu nedenle, her biri farklı bir özellik alt kümesi üzerinde eğitilmiş 3 farklı modelimiz olacaktır. Sonunda yeni test verilerimizi bu modellerin her birine aktaracağız ve benzersiz bir tahmin alacağız. En fazla oyu alan tahmin, rastgele orman algoritmasının nihai kararı olacaktır. Örneğin, DT1 ve DT3 test verilerimizin belirli bir örneği için pozitif bir sınıf öngörürken, DT2 negatif bir sınıf öngörmüştür. Pozitif sınıf oyların çoğunluğunu aldığından (2), rastgele ormanımız bu örneği olumlu olarak sınıflandırır. Yine, Rastgele Orman algoritmasının çeşitli modelleri eğitmek için rastgele bir özellik alt kümesini kullanırken, her model veri kümesinin yalnızca belirli alt kümesini görür.

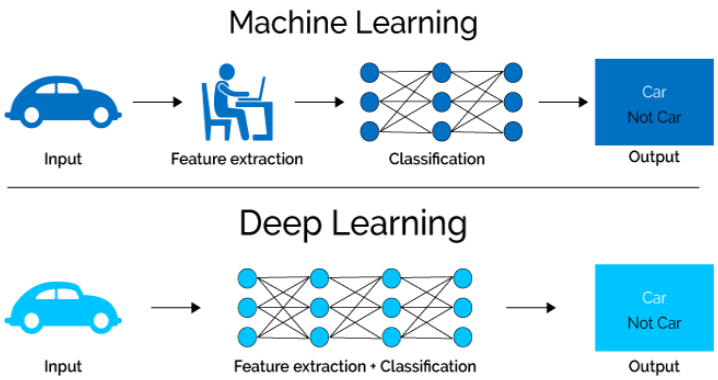
Rastgele orman, en yaygın kullanılan topluluk öğrenme algoritmalarından biridir. Neden bu kadar etkili? Bunun nedeni, orijinal veri kümesinin birden çok örneğini kullanarak, son modelin varyansını azaltmamızdır. Düşük varyansın düşük overfitting anlamına geldiğini unutmamak gerekir. Overfitting, modelimiz veri kümesindeki küçük varyasyonları açıklamaya çalıştığında olur çünkü veri kümemiz, modellemeye çalıştığımız olgunun olası tüm örneklerinin popülasyonunun sadece küçük bir örneğidir. Eğitim setimizin bazı durumlar içerebilir: gürültü, aykırı değerler ve aşırı veya az temsil edilen örnekler gibi. Eğitim setimizin değiştirilmesi ile birden fazla rastgele örnek oluşturarak bunların etkisi azaltılır.

3.1.6 Derin Öğrenme

Şu anda büyük bir hızla büyüyen yapay zekaya katkıda bulunanlardan biri de derin öğrenmedir. Yaşadığımız dünyayı değiştirirken derin öğrenmenin temellerini anlamak bize fayda sağlayacaktır.

Derin öğrenme yapay sinir ağları adı verilen beynin yapısı ve işlevinden esinlenen algoritmalarla uğraşan makine öğreniminin bir alt alanıdır. Başka bir deyişle, beynimizin işleyişini yansıtır. Derin öğrenme algoritmaları, sinir sisteminin her bir nöronun birbirine bağlandığı ve bilgiyi ilettiği yerde nasıl yapılandırıldığına benzer. Derin öğrenme modelleri katmanlar halinde çalışır ve tipik bir modelin en az üç katmanı vardır. Her katman bir öncekinden gelen bilgileri kabul eder ve bir sonrakine

aktarır. Derin öğrenme modelleri veri miktarı ile iyi performans gösterirken eski makine öğrenme modelleri doygunluk noktasından sonra gelişmeyi durdurur.



Şekil 3.15: Makine Öğrenme ile Derin Öğrenme Karşılaştırılması

Makine öğrenimi ile derin öğrenme modeli arasındaki farklardan biri özellik çıkarma alanındadır. Özellik çıkarma, makine öğreniminde insan tarafından yapılırken derin öğrenme modeli kendi kendine ortaya çıkar.

Derin öğrenmede önceden eğitilmiş çoğu model için temel oluşturan bazı önemli sinir ağı tipleri bulunmaktadır. Konvolüsyonel Sinir Ağları (CNN) şu anda derin öğrenme topluluğundaki en önemli modellerden biridir. CNN modelleri farklı uygulamalarda; özellikle görüntü ve video işleme projelerinde yaygın olarak kullanılmaktadır.

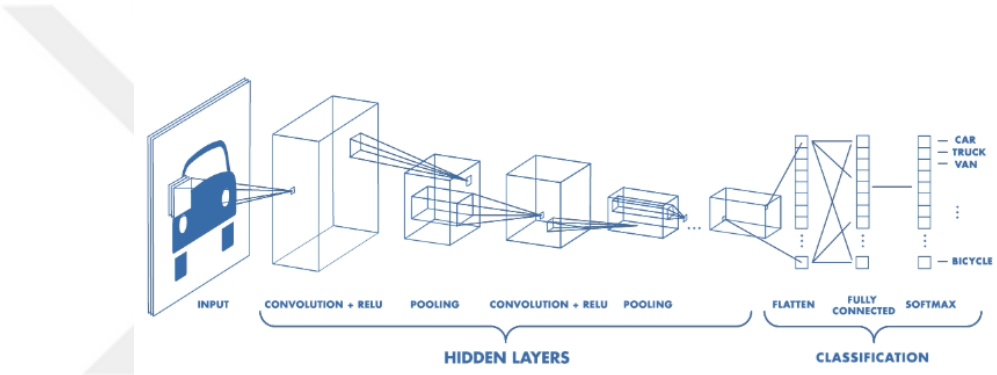
3.1.6.1 CNN

Konvolüsyonel sinir ağı (CNN), bilgisayarlı görme için LeCun ve arkadaşları (1989) tarafından ileri sürülen ileri beslemeli bir yapay sinir ağıdır. Sonraki yıllarda farklı doğal dil işleme görevleri için kullanılmıştır; cümle modelleme (Kalchbrenner vd., 2014), part-of-speech parçalama, adlandırılmış varlık tanıma, semantik rol etiketleme (Collobert vd., 2011), cümle sınıflandırması (Kim, 2014).

Yapay zeka ve sinir ağı, makinelerin öğrenmesiyle insanların öğrendikleri gibi ilgilenir. CNN en önemli sinir ağlarından biridir ve o da bu şekilde davranır (LeCun vd, 1998). CNN, gözetimli öğrenme algoritması kullanan özel bir yapay sinir ağı türüdür. Gözetimli öğrenme algoritması, algoritmanın verilen etiketli örnek girişlerinden öğrendiği anlamına gelir. Eğitim setimizde verdiğimiz her görüntü için

bir sınıfımız vardır. Bu sınıf, kuş, kedi, vb. gibi genel bir sınıflandırma öğesi etiketi olabilir veya kedi türlerinin daha ayrıntılı bir sınıflandırması olabilir. Eğitim setimizde belirli kedi türlerinden örnekler verirsek, algoritma bu ayrıntıları öğrenir. Etiketli eğitim setinde sadece kedi, kuş ve köpek sınıfları verirsek, algoritma sadece bu kategorileri gruplamayı öğrenir.

Bir CNN; giriş katmanı, çıkış katmanı ve ilk ve son katmanlar arasında çeşitli gizli katmanlar içerir. Giriş ve çıkış arasındaki katman sayısı CNN algoritmasında gerçekleştirilen göreve göre değiştirilebilir. Şekil 3.16, örnek bir CNN yapısının göstermektedir.



Şekil 3.16: CNN yapısına örnek.

Bu yapıda, görsel özellikler yerel olarak eğitilmiş filtrelerin giriş görüntüsü üzerinde kaydırılmasıyla çıkarılır. Oluşturulan özellik haritalarının boyutu havuz oluşturma işlemi ile azaltılır ve daha sonra bu haritalar bir sonraki konvolüsyon katmanına aktarılır. Bu işlem özellikler çıkarılana kadar tekrarlanır. Bundan sonra, bu özellikler üzerinde karar vermek için bir sınıflandırıcı kullanılır (Ravi vd., 2017). Genellikle CNN, konvolüsyonel katmanlar, havuzlama katmanları ve tam bağlantılı katmanlardan oluşur (Schmidhuber, 2015). Bu yapıda, özellik çıkarıcı kısmında konvolüsyon ve havuzlama katmanları bulunur, tam bağlı ağ ise bir sınıflandırıcı olarak kullanılır. Bu şekilde oluşturulan AlexNet (Krizhevsky vd., 2012), OverFeat (Sermanet vd., 2013), GoogLeNet (Szegedy vd., 2015) gibi birçok popüler yapı vardır.

3.1.6.1.1 Konvolüsyon Katmanı

Konvolüsyon, filtreler kullanılarak girdi görüntüsüne karmaşık fonksiyonlar uygulanarak gerçekleştirilir (Ravi vd., 2017). Konvolüsyonel filtreler sabit boyutlu çekirdeklerdir. Bu çekirdekler, uygulamaya göre yerel olarak eğitilir ve konvolüsyon işlemi sırasında tüm giriş görüntüsünün üzerinde kaydırılır ve her pozisyonun bir aktivasyon değeri üretilir. Bu yapıda, her bir nöron, bir önceki tabakadaki belirli bir alanda sınırlı sayıda nöron tarafından bağlanır. Bu alana alıcı alan denir. Her bir konvolüsyonel katman üzerinde çok sayıda konvolüsyonel filtre vardır ve her bir filtre, girdi görüntüleri üzerindeki belirli bir örüntüyü tanımak üzere eğitilmiştir. Filtre, görüntünün üzerinde kayarken aynı ağırlık ve sapma değerlerine sahiptir. Buna ağırlık paylaşım mekanizması denir. Bu şekilde, aynı desenler ilgili filtre tarafından tüm girdi görüntüsünde taranabilir (Chandrakumar ve Kathirvel, 2016; Ravi vd., 2017; Sankar vd., 2016). Bir konvolüsyon katmanının filtre sayısı dikkatle belirlenmelidir. Birkaç filtre kullanılırsa, birkaç desen tanındığı için performans düşebilir. Çok sayıda filtre kullanılırsa, parametre sayısı artar ve ağın eğitimi yavaşlar. Dikkate alınması gereken diğer bir konu, filtrelerin boyutudur. Genellikle ilk katmanlarda büyük filtreler, diğer katmanlarda daha küçük filtreler kullanılır. Özellikle ilk katmanlarda, görüntü boyutunu etkili bir şekilde azaltmak için uzun adımlarla işaretleme gerekebilir.

mxm giriş boyutu, cxc filtrenin boyutları, i girişi temsil eder, w ve b paylaşılan ağırlık ve sapma değerleridir. Bu değerlerle $\sigma_{0,0}$. nöron çıkışı 3.1 (3.1) 'de olduğu gibi hesaplanabilir. Burada f, Rektifiye Lineer Ünite (ReLu), sigmoid, vb. gibi bir aktivasyon fonksiyonudur (Nielsen, 2015).

$$\sigma_{0,0} = f(b + \sum_{t=0}^{c}\sum_{r=0}^{c} w_{t,r}{}^{i}{}_{0+t,0+r}) \tag{3.13}$$

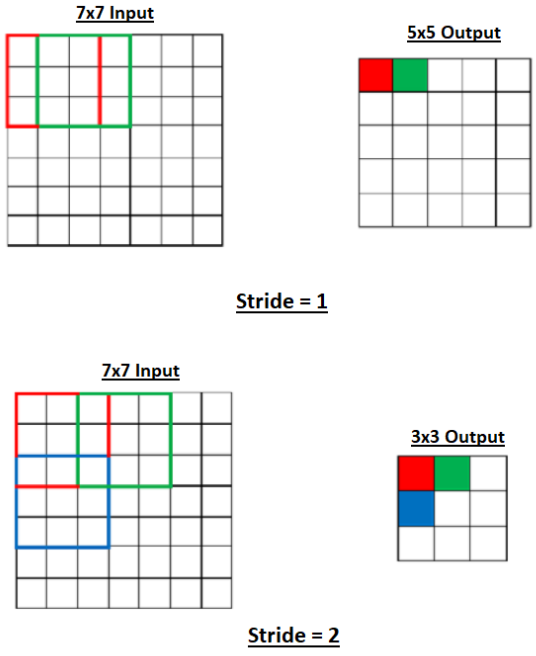
Filtre görüntünün üzerinde kaydığında ve aktivasyon hesaplandığında, kayma her adımda birden çok piksel olabilir. Bu şekilde, özellik haritasının boyutu küçültülecektir. Ayrıca, giriş görüntüsü konvolüsyon işleminden önce 0 değerli piksellerle genişletilebilir.

3.1.6.1.2 Aktivasyon Katmanı

Konvolüsyon işleminden sonra, sonuçlar bir aktivasyon fonksiyonundan geçirilir. Aktivasyon fonksiyonları; çoğunlukla doğrusal olmayan fonksiyonlardır ve doğrusal olmayan gerçek dünya problemlerinde yapay sinir ağlarının kullanımını sağlarlar. En yaygın kullanılan aktivasyon fonksiyonları ReLU, sigmoid, hiperbolik tanjant ve Softmax'tir.

3.1.6.1.3 Stride (Adım) ve Padding (Dolgu) İşlemleri

Stride, filtrelerin giriş verileri üzerinde kaç birim hareket ettiğini ifade eder. Şekil 3.17, 7x7 giriş veri boyutuna 3x3 filtre uygulanmasını göstermektedir. Stride = 1 üstte, Stride = 2 ise şeklin alt kısmında verilmiştir. Filtre, Stride = 1 olduğunda 1 birimi x ve y yönünde hareket ettirir, Stride = 2 olduğunda 2 birimi x ve y yönünde hareket ettirir.



Şekil 3.17: Stride İşlemi

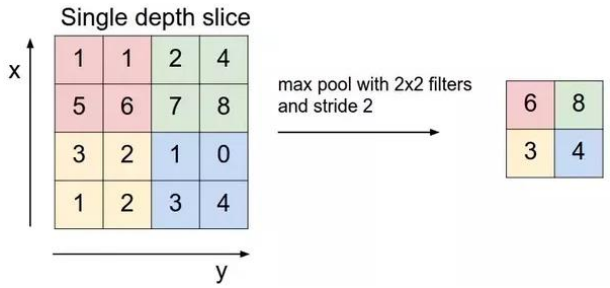
Padding, giriş verilerinin boyutunun küçülmesini önler ve böylece sonraki katmanlara iletilen özellik sayısını artırır. Stride ve padding işlemlerinden sonraki çıktı boyutu X denklemi ile ifade edilir.

$$çıktı = \frac{girdi - filtre + 2 * padding}{stride} + 1 \quad (3.14)$$

3.1.6.1.4 Havuzlama Katmanı

Özellik haritaları konvolüsyon işlemi ile üretildikten ve bir aktivasyon fonksiyonundan geçtikten sonra, havuzlama işlemi uygulanır. Bu işlemin amacı, minimum veri kaybı ile özellik haritalarının boyutunu azaltmaktır. Özellik haritalarının boyutunu azaltmak, daha az parametre ile daha hızlı bir öğrenme sürecine olanak tanır. Bu işlem ayrıca, görüntünün üzerinde filtrelerin kaydırılmasıyla da gerçekleştirilir. Ancak bu sefer bir filtre bir havuzlama işlemi uygular.

En popüler havuzlama işlemleri maksimum havuzlama, ortalama havuzlama ve L2 havuzlamadır (Nielsen, 2015). Maksimum havuzlama maksimum değeri seçer, ortalama havuzlama ortalamayı hesaplarken, L2 havuzlaması çekirdek boyutlu giriş alanındaki giriş değerlerinin L2 normunu alır. Havuzlama süreci, eğitimi hızlandırmanın yanı sıra, desenin resimdeki konumundan bağımsız olarak tanınmasına izin verir (Nielsen, 2015).



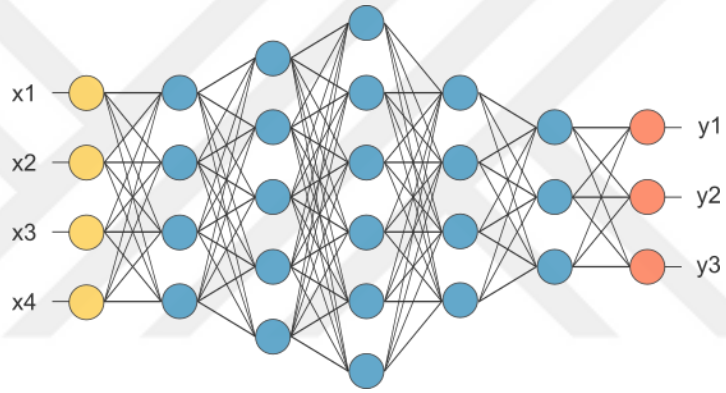
Şekil 3. 18: Max Pooling Örneği

3.1.6.1.5 Tam Bağlantılı Katman

Konvolüsyon ve havuzlama katmanları da dahil olmak üzere, özellik çıkarıcısının son katmanındaki nöronlar, tek boyutlu bir vektöre dönüştürülür. Bu işlemden sonra gelen katmanlar tam bağlantılı katmanlardır. Tam bağlantılı katman nöronları aşağıdaki hesaplamayı yapar:

$$f_{c_1} = f(b + \sum_{q=1}^{M} w_{1,q} * o_q \quad (3.15)$$

Burada f aktivasyon fonksiyonudur ve M bir vektörde hizalanmış önceki katman nöronlarının sayısıdır, $o_q$dördüncü nöronun değeri, f_{c_1} tamamen bağlı ağın ilk gizli katmanındaki ilk nöron ve $w_{1,q}$ $o_q$ve f_{c_1} nöronları arasındaki ağırlık değeridir. Tam bağlı katmanlardan oluşan tam bağlı sinir ağı yapısı genellikle derin öğrenme yapılarında sınıflandırıcı olarak kullanılır. Bu yapıyı genellikle sınıflandırma amaçları için bir SoftMax çıktı katmanı izler.



Şekil 3.19: Tam Bağlantılı Katman Örneği

Yukarıdaki şemada, özellik harita matrisi vektör $(x1, x2, x3, ...)$ olarak dönüştürülecektir. Tamamen bağlı katmanlarla, bir model oluşturmak için bu özellikleri bir araya getirdik. Son olarak, çıktıları kedi, köpek, araba, kamyon vb. Olarak sınıflandırmak için SoftMax veya sigmoid gibi bir aktivasyon fonksiyonumuz var.

SoftMax işlevi, çoklu sınıflar için genel lojistik regresyon şeklidir. N gerçek sayıların bir girdi vektörünü alır ve N olasılıklardan oluşan bir olasılık dağılımı çıkarır. Hesaplanan tüm olasılık değerleri (0,1) aralığındadır ve bu olasılık değerlerinin toplamı 1'e eşittir. Daha yüksek giriş değerleri daha yüksek olasılık değerlerine yol açar. Çalışmaların çoğunda, konvolüsyonel sinir ağları, sınıflandırma amacıyla bir SoftMax katmanı ile sonuçlanır. (3.16) da tanımlandığı gibidir.

$$class_j = \frac{\exp\,(sf_j)}{\sum_q \exp\,(sf_q)} \tag{3.16}$$

Burada $class_j$, dördüncü çıkışın değeri, sf_j, softmax aktivasyon fonksiyonunun dördüncü giriş değeridir ve sf_q, SoftMax katmanındaki her bir nöronu gösterir.

3.1.6.1.6 Çıkış Katmanı

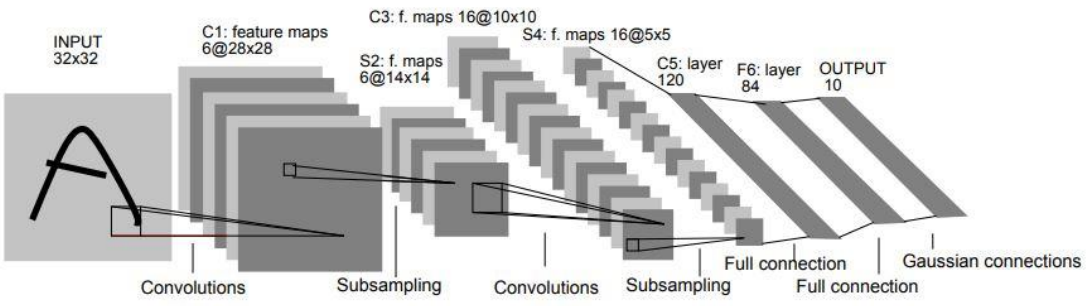
Çıktı katmanı ayrıca çok katmanlı algılayıcının tam bağlı katmanlarıdır. Bu katmanın girdisi havuzlama katmanından gelen çıktıdır. Sınıf olasılığı, bu girdilerin çeşitli aktivasyon fonksiyonları vasıtasıyla ağırlıklandırılmasıyla elde edilir.

3.1.6.1.7 Temel CNN Mimarileri

Görüntü sınıflandırması için CNN'lerin başarısına büyük katkıda bulunan bazı temel mimariler bulunmaktadır. Bunlar; LeNet, AlexNet, OverFeat, GoogLeNet ve VGGNet.

3.1.6.1.7.1 LeNet

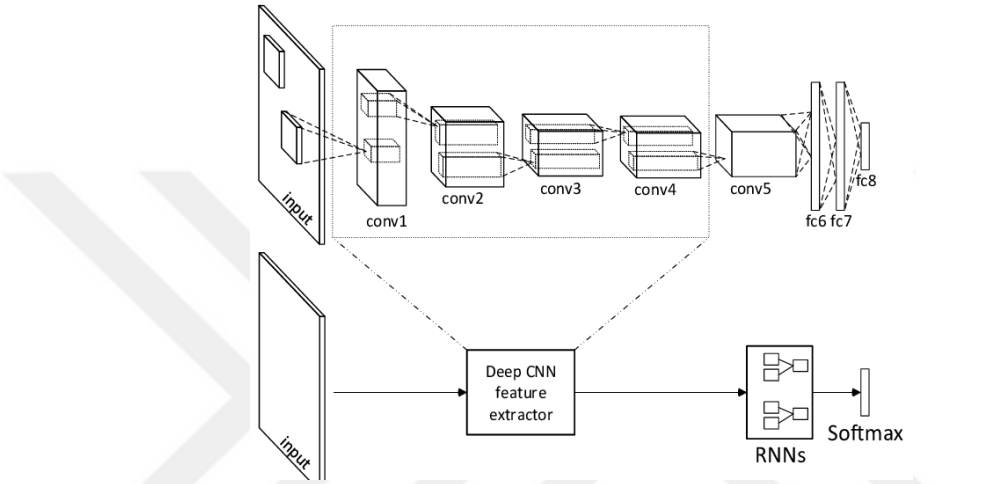
LeNet (LeCun vd., 1998) ilk yayınlanan CNN yapısıdır. İki kıvrımlı ve havuzlu katman çifti içerir ve MNIST veri kümesini sınıflandırmak için kullanılır (LeCun vd., 1995). LeNet yapısı Şekil 3.20'te verilmiştir. Bilgisayarlar, o zamanlar daha büyük veri kümeleri için daha büyük ağları eğitecek kadar güçlü olmadığından, uzun süre test edilemediler.



Şekil 3.20: LeNet Yapısı

3.1.6.1.7.2 AlexNet

AlexNet (Krizhevsky vd., 2012), ILSVRC-2012 yarışmasının ImageNet veri kümesinde %16,4'lük en yüksek hata oranı ile kazanan modelidir. AlexNet, Şekil 3.21'te verilen iki paralel özellikli tam bağlı bir ağ yapısından oluşur. Her modül 5 konvolüsyonel katman ve 3 havuzlama katmanı içerir. AlexNet'in yüksek performansı, konvolüsyonel yapılara olan ilgiyi artırmıştır.



Şekil 3.21: AlexNet Yapısı

3.1.6.1.7.3 OverFeat

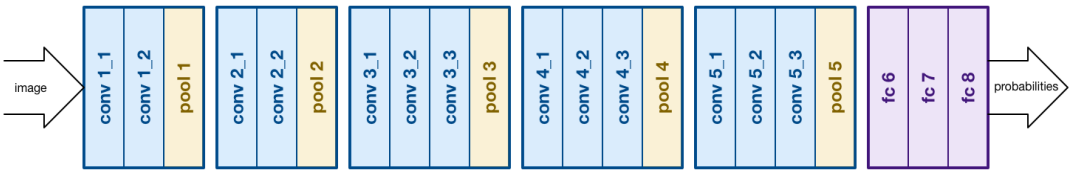
OverFeat (Sermanet vd., 2013), 5 kıvrımlı katman, 3 havuzlama katmanı ve 3 tam bağlı katman içerir. Transfer öğrenme uygulamalarında yaygın olarak kullanılmaktadır.

3.1.6.1.7.4 GoogLeNet

GoogLeNet, LeNet'e benzer bir yapıya sahiptir; bununla birlikte, farklı boyutlardaki filtrelerle konvolüsyon işlemleri ve tek bir yapıda havuzlama işlemi sağlayan başlangıç modülleri içerir. GoogLeNet, başlangıç modüllerinin dahil edilmesiyle çok daha az parametre ile önemli ölçüde daha yüksek performans değerleri elde etmektedir.

3.1.6.1.7.5 VGGNet

VGGNet, 16 konvolüsyonel katman, 5 havuzlama katmanı ve 3 tam bağlı katmandan oluşur. Bu yapının tüm konvolüsyon katmanları 3x3 konvolüsyon operasyonları uygular. Çok başarılı performans değerlerine sahip olmasına rağmen, çok sayıda parametreye sahip olması kullanımı zorlaştırır.



Şekil 3.22: VGGNet Yapısı

DÖRDÜNCÜ BÖLÜM

4. DENEYLER

4.1 Veri Seti

Bu bölümde, yönelimi tespit etmek üzere kullanılacak olan 4 farklı sınıfa ait web sitesinden elde edilmiş metinlerden veri seti oluşturulmaktadır. Bu sınıflar; dini, sağlık, spor ve üniversite olarak belirlenmiştir. Web sitelerindeki metinlere erişim yapılırken Java kütüphanesi olan JSoup kullanılmıştır. JSoup, veriye erişim için güçlü bir API sağlayan açık kaynaklı bir projedir. HTML'i URL'lerden, dosyalardan ve dizelerden ayrıştırmak için kullanılabilir. Ayrıca, HTML öğelerini veya niteliklerini değiştirebilir.

Veri setini oluştururken ilk olarak, belirlenen sınıflara ait İngilizce metin içeren web sitelerinin linklerine erişim sağlanmıştır. Linkler sitelerin alt linkleridir ve her bir web sitesi için 150 link olarak sınırlandırılmıştır. Buradan erişilen metinler Doc2Vec yönteminde kullanılmak üzere 100'er kelimelik dosyalara ayrılmıştır. Bir paragrafın ortalama 100 kelimeden oluşacağı düşünülerek bu işleme karar verilmiştir. Bunun sonucunda, 4 sınıf için her biri 10.000 olmak üzere toplamda 40.000 adet belge elde edilmiştir.

4.2 Veri Ön İşleme

Veri setlerine ön işlem uygulanması metin çalışmalarındaki en önemli aşamalardan biridir. Bu yüzden Doc2Vec modeli oluşturulmadan önce veri setine ön işlem uygulanmıştır. Öncelikle veri setinde alfabetik olmayan tüm karakterler, noktalama işaretleri ve boş satırlar temizlenmiştir, ardından tüm karakterler küçük harfe dönüştürülmüştür.

Yaygın olarak kullanılan diğer bir ön işlem yöntemi de, herhangi bir bilgilendirici değeri olmayan, yaygın olarak kullanılan gereksiz sözcükleri (örneğin; a, and, or, this, they) kaldırmaktır. Bu yöntem için, 210 adet İngilizce kelimeden oluşan bir liste oluşturuldu ve bunlar veri setinden ayıklanmıştır.

Tablo 4.1: Örnek dosya metninin temizleme aşamasından önceki hali.

Vacancies, those open to all applicants and others which are only open to ANU staff. To find out how we will handle your information during the job application process, see our.

Tablo 4.2: Örnek dosya metninin temizleme aşamasından sonraki hali.

vacancies open applicants open anu staff find handle information job application process see

Ön işlem aşamasının son adımında ise "Stemming" adı verilen ve doğal dil işleme alanındaki çalışmalarda metin, kelime ve belge hazırlamak için kullanılan yöntem ile veri setindeki türetilmiş kelimeler köklerine indirgenmiştir. Bu işlemde Python dilinde yazılan NLTK kütüphanesi kullanılmıştır.

Tablo 4.3: Örnek kelimelerin kök veya kök formlarına ayrılmış halleri.

weekend discover campus drinks parking accessibility australian	weekend **discov** **campu** **drink** **park** **access** australian

Ön işlem aşaması tamamlandıktan sonra veri setimiz Doc2Vec modelini oluşturmaya hazır hale gelmiştir.

4.3 Doc2Vec Modeli

Bu aşamada elde edilen veri setinden Doc2Vec modeli oluşturmak için Python dilindeki Gensim kütüphanesinden yararlanılmıştır. Gensim kütüphanesi vektör uzayı modelleme, başlık modelleme özellikleri olan ve bunlar üzerine uzun metinler için verimli olarak çalışabilen bir kütüphanedir. Bu kütüphane kullanılırken Numpy ve SciPy kütüphanelerine de ihtiyaç duyulmaktadır. Bu kütüphaneler Python'ın temel kütüphaneleridir.

Eğitimi yapılmış olan modelin içindeki doküman vektörlerine erişilir. Paragraf vektörü çerçevesinde, kelimeler ve paragraflar benzersiz vektörlerle temsil edilir ve her paragrafın kendine özgü bir vektörü vardır. Her paragraf için paragraf bağlamından benzersiz bir vektör oluşturduktan sonra, bu vektörler paragrafların özellikleri olarak değerlendirilir.

Doc2Vec modeli oluşturulurken bazı önemli parametreler kullanılmaktadır. Bunlardan biri özellik vektörleri boyutudur. Özellik vektörleri boyutu 1024 olarak belirlenmiştir. Diğer önemli parametrelerden bazıları ise epoch ve iterasyondur. Bunlar birbirlerine sıkça karıştırılan kavramlardır. Bir epoch, tüm veri kümesinin yalnızca bir kez sinir ağı üzerinden ileri ve geri geçirildiği zamandır. Bir epoch zamanı bilgisayar için büyük olduğundan birkaç küçük gruba bölünür. Birden fazla epoch kullanılmasının sebebi tüm veri setini bir sinir ağından geçirmek yeterli değildir. Aynı sinir ağına defalarca iletmemiz gerekir. Bu yüzden, tek bir geçişle ya da epoch'la ağırlıkları güncellemek mümkün değildir. Epoch sayısının kaç olması gerektiğinin tam olarak bir yanıtı yoktur. Farklı veri kümelerine göre değişir. Epoch sayısı verilerin ne kadar çeşitli olduğuna bağlıdır.

Bir epoch, algoritmanın tüm veri kümesini kaç kez gördüğünü açıklar. Dolayısıyla, algoritma veri kümesindeki tüm örnekleri her gördüğünde, bir epoch tamamlanmıştır.

Iterasyon, bir veri kümesinin algoritmadan kaç kez geçtiğini açıklar. Sinir ağlarında bu, ileri geçiş ve geri geçiş anlamına gelir. Böylece, bir grup veriyi sinir ağları üzerinden her ilettiğinizde, bir iterasyon tamamlanmış demektir. Modelimiz eğitilirken iterasyon değerleri 15, 25, 50, 100 olarak ayrı ayrı bakılıp iterasyonun doğruluk oranına etkisini araştırılmıştır.

Doc2Vec model eğitimi yaparken kullandığımız bazı parametreler bulunmaktadır. Bu parametrelerin önemli olanları aşağıdaki tabloda verilmiştir (Tablo 4.4). Tablodaki

parametrelerin bazıları Gensim kütüphanesinde default olarak gelen değerleri değiştirilmeden kullanılmıştır.

Tablo 4.4: Doc2Vec Model Eğitiminde Kullanılan Parametreler

Parametre	Değer
dm	1 = dm, 0 = dbow
vector_size	1024
window	3
alpha	0.025
min_alpha	0.025
min_count	5

Dm parametresi, eğitim algoritmasını tanımlar. Eğer dm = 1 ise, dağıtık bellek (PV-DM), eğer dm = 0 ise, dağıtık kelime torbası (PV-DBOW) kullanılır. Vector_size, özellik vektörlerinin boyutsallığını belirtir. Burada vector_size'ın 1024 kullanılmasının sebebi bu çalışma kapsamında daha az boyutların vektör uzayında ayrım yapabilmeye yeterli gelmemiş olmasıdır. Max_size, cümle içerisinde geçerli ve tahmin edilen kelime arasındaki maksimum mesafedir. alpha, başlangıç öğrenme oranıdır. Min_alpha, eğitim ilerledikçe öğrenme oranı doğrusal olarak bu değere düşecektir. Min_count, toplam frekansı belirlenen değerden daha düşük kelimeleri yok sayar.

4.4 Boyut Azaltma ve Görselleştirme

Günümüzdeki sorunlardan biri de, çoğu veri kümesinin çok sayıda değişkene sahip olmasıdır. Başka bir deyişle, veriler yüksek sayıda boyuta sahiptir. Bu yüzden verileri görsel olarak keşfetmek zorlayıcı hale gelebilir ve çoğu zaman manuel olarak yapmak bile neredeyse imkansızdır. Ancak, bu tür görsel araştırmalar veri ile ilgili herhangi bir problemde inanılmaz derecede önemlidir. Bu nedenle, yüksek boyutlu veri kümelerinin nasıl görselleştirileceğini anlamak çok önemlidir. Boyut azaltma olarak

bilinen teknikler kullanılarak gerçekleştirilebilir. Bu çalışmada yüksek boyutları azaltmamıza izin verecek iki teknik kullanıldı: PCA ve t-SNE.

PCA, birbirinden farklı veri noktalarını daha düşük bir boyut temsilinde birbirinden ayırmaya çalışır. Ancak, düşük boyutlu, doğrusal olmayan manifoldda yüksek boyutlu verileri temsil etmek için, benzer veri noktalarının PCA'nın yaptığı gibi birbirine yakın olarak temsil edilmesi önemlidir. Bu, t-SNE tarafından verimli bir şekilde yapılır. Böylece, veri setindeki daha zorlayıcı manifoldların yapısını verimli bir şekilde yakalayabiliriz.

t-SNE, verilerin yerel ve küresel yapısını koruyabilir. Bu, kabaca, yüksek boyutlu veri kümesinde birbirine yakın olan noktaların, düşük boyutta birbirine yakın olma eğiliminde olacağı anlamına gelir. Öte yandan, PCA, verilerdeki varyansın çoğunu açıklayan yeni boyutlar bulur. Bu yüzden, t-SNE'den farklı olarak yerel komşular hakkında göreceli olarak çok az önem verir.

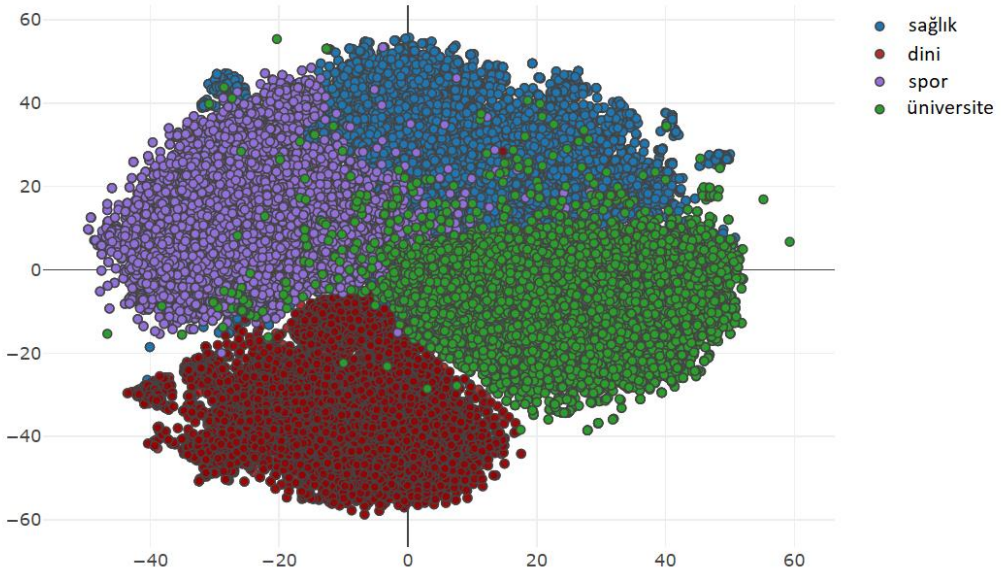
t-SNE birçok hesaplama içerir, çünkü her veri noktası için ikili koşullu olasılıkları hesaplar ve daha yüksek ve düşük boyutlardaki olasılık farkının toplamını en aza indirmeye çalışır.

t-SNE veri noktalarının sayısında zaman ve mekan karmaşıklığına sahiptir. Bu, onu 10.000'den fazla gözlemden oluşan veri setlerine uygularken özellikle yavaş, işlemsel olarak oldukça ağır yapar.

Yukarıdaki sorunların çözümü olarak, hem PCA hem de t-SNE'yi birlikte kullanmak gerekir. Veri kümesinde binlerce özelliğe sahip olduğumuzdan, ilk önce boyutları makul bir sayıda özelliğe indirgemek için PCA kullandık ve ardından boyutlandırmayı daha da azaltmak için t-SNE'yi çalıştırdık. Bunlar için Python dilinde yazılmış Scikit-Learn kütüphanesi (Brownlee, 2014) kullanılmıştır.

Scikit-Learn kütüphanesi, gözetimli ve gözetimsiz öğrenme algoritmaları içeren popüler bir pakettir. Popüler olmasının sebeplerinden bazıları temel yöntemlerin çoğunluğunu içermesi ve veri analizi uygulamalarında baştan sona kullanılabilir olmasıdır. Eksik verileri doldurmak, öznitelik seçmek, sonuç değerlendirmek gibi birçok içerik ile başka bir pakete ihtiyaç duyulmamaktadır.

Bu doğrultuda, veri setimizde bulunan 1024 boyutundaki örnekler PCA ile 50 boyuta düşürüldü. Daha sonra t-SNE ile 2 boyutlu hale getirilerek görselleştirmek için hazır hale getirildi (Şekil 4.1).



Şekil 4.1: Dört sınıfa ait Doc2Vec modelinin 2 boyutlu grafiği

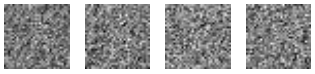
2 boyuta indirgenen özellik vektörlerini görselleştirmek için Python dilinde yazılmış Plotly kütüphanesi kullanıldı. Bu görselleştirmeleri incelendiğinde, birbirleriyle ilişkili belgelerin bir arada bulunduğu ve yönelim sınıflarının birbirinden ayrıldığı gözlemlenmiştir.

4.5 CNN Modeli

Doc2Vec eğitim modeli oluşturulduktan sonra her sınıfa ait 10.000 örnekten oluşan, toplam 40.000 örnekten 40.000x1024 boyutlu bir model elde ettik. CNN'in görüntü işleme alanında yüksek başarı elde ettiği yapılan çalışmalarla bilinmektedir. Çalışmamızda CNN'de kullanmak üzere vektörlerimizi görüntüye çevirmemiz gerekir. Görüntüler piksel değerlerinden meydana gelir. Bu değerler 0, 255 aralığında işaretsiz tamsayılardır. Ancak, Doc2Vec modelimizdeki vektörler -1, 1 aralığında sayı değerleri içermektedir. Bu yüzden vektörleri bu aralığa dönüştürmemiz gerekir. Vektörleri 0, 255 piksel değerlere çevirmek için kullandığımız formül aşağıdadır.

$$(vektor - \min(vektor)) * (1/(\max(vektor) - \min(vektor)) * 255) \quad (3.17)$$

Vektörlerimiz istediğimiz aralığa çevirdikten sonra modelde bulunan 1024 boyutlu her bir örnek vektörü 32x32 boyutunda resimlere dönüştürdük (Şekil 4.2). Bunun sonucunda ortaya çıkan resimler görsel olarak anlam ifade eden resimler değiller. Ancak CNN'de girdi olarak kullanılmak üzere 4 sınıfa ait örneklerden görüntüler elde etmiş olduk.



Şekil 4.1: 32x32 boyutlu örnek resimler

CNN modeli, arka planda TensorFlow kullanan yüksek seviyeli Keras API'sı ile oluşturuldu. Sıralı modeli Keras'tan içe aktarıp, Conv2D, MaxPooling, Flatten ve Dense katmanları eklendi.

CNN'ler, verilere sırayla uygulanan çeşitli işlem bloklarından veya modüllerden oluşur. Bu modüllerin sıralaması ve parametre ayarları, bir ağın sınıflandırma performansı üzerinde önemli bir etkiye sahiptir. Sonuç olarak, belirli bir uygulamanın CNN mimarisi dikkatlice ayarlanmalıdır. Bu çalışma için seçilen CNN mimarisi Tablo 4.5'te gösterilmektedir. Mimaride, 4 adet 2D konvolüsyon katmanı bulunmaktadır. İlk Conv2D katmanı 16 çıkış boyutunda 3x3 filtreye sahiptir. Diğer Conv2D katmanları aynı filtreye sahiptir ancak çıkış boyutları ikiye katlanarak gitmektedir (sırasıyla 32, 64, 128). Her bir konvolüsyon katmanından sonra bir MaxPooling katmanı gelir. MaxPooling [16], bir piksel grubunun grup içindeki maksimum piksel yoğunluğuyla değiştirildiği bir işlemi ifade eder. Bu çalışmada maksimum 2x2 piksel filtre ile maksimum havuzlama yapılmıştır. Son konvolüsyon katmanının ardından Flatten katmanı gelmekte ve böylece tam bağlı katmana geçilerek çıktılar düzleştirilmiştir. Daha sonra gelen Dense katmanı 4096 çıkış boyutludur ve ReLu aktivasyon fonksiyonu ile etklinleştirilir. Son Dense katmanı çıktı katmanıdır. Burada bulunan 4 sayısı, sınıf sayısını temsil eder. Bu katmanda; sınıflara ait örneklerin olasılığını ifade eden SoftMax aktivasyon fonksiyonu kullanılır.

Tablo 4.5: Kullanılan CNN Mimarisi Katmanları

Input (32x32 RGB Image)
Convolution - 16 (3x3 Filter)
MaxPooling - (2x2 Filter)
Convolution - 32 (3x3)
MaxPooling - (2x2 Filter)
Convolution - 64 (3x3)
MaxPooling - (2x2 Filter)
Convolution - 128 (3x3)
MaxPooling - (2x2 Filter)
Flatten
Dense – 4096 (Aktivasyon Fonk. = ‘ReLu’)
Dense – 4096 (Aktivasyon Fonk. = ‘ReLu’)
Dense (4096, Aktivasyon Fonk. = ‘SoftMax’)

4.6 Değerlendirme ve Sonuç

Öncelikle, Doc2Vec model eğitimi ile vektörel olarak ifade edilen belgeler, sınıflandırma işlemleri için %80 eğitim ve %20 test kümesi olarak ayrılmıştır. Sınıflandırma için seçilen CNN mimarisinden ve makine öğrenmesi sınıflandırma yöntemleri olan Random Forest ile Gauss Naive Bayes’ten yararlanılmıştır. Random Forest ve Gauss Naive Bayes sınıflandırmaları yaparken bir veri analiz platformu olan Knime (Konstanz Information Miner) yazılımı kullanılmıştır.

Sınıflandırma algoritmaları kullanılan çalışmalarda başarı kriterleri bulunmaktadır. Bunlardan bazıları; Doğruluk (Accuracy), Kesinlik (Precision), Duyarlılık (Recall) ve F1 Ölçütü. İçlerinden sadece doğruluğa bakmak büyük bir yanılgıdır. Buna ek olarak kesinlik, duyarlılık ve f1 ölüçütüne de bakmak gerekir.

Öncelikle bilmemiz gereken TP, TN, FP, FN metriklerinin ve Karmaşıklık Matrisinin (Confusion Matrix) ne olduğunu inceleyelim.

Karmaşıklık Matrisi; Makine öğrenimi ve özellikle istatistiksel sınıflandırma sorunu alanında, hata matrisi olarak da bilinen bir karışıklık matrisi. Karışıklık matrisi, gerçek değerlerinin bilindiği bir test verisi grubundaki bir sınıflandırma modelinin (veya "sınıflandırıcı") performansını tanımlamak için sıklıkla kullanılan bir tablodur. Bir algoritmanın performansının görüntülenmesini sağlar.

Sınıflar arasındaki karışıklığın kolayca tanımlanmasını sağlar, örneğin bir sınıf genellikle diğeri gibi yanlış etiketlenebilir. Çoğu performans ölçüsü karmaşıklık matrisinden hesaplanır.

Pozitif	Negatif		
TP	FP	Pozitif	Prediction (Tahmin)
FN	TN	Negatif	

Tablo 4.6: Karmaşıklık Matrisi

TP (True Positive – Doğru Pozitif): Örneğin; Hastaya hasta demek.

TN (True Negative – Doğru Negatif): Örneğin; Hasta olmayana hasta değil demek.

FP (False Positive – Yanlış Pozitif): Örneğin; Hasta olmayana hasta demek.

FN (False Negative – Yanlış Negatif): Örneğin; Hasta olana hasta değil demek.

Doğruluk, sınıflandırma modellerinin değerlendirilmesi için bir metriktir. Modelimizin doğru yaptığı tahminlerin bir kısmıdır. Doğruluk, pozitif ve negatifler açısından aşağıdaki gibi hesaplanabilir.

$$Doğruluk = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Kesinlik değerini elde etmek için doğru sınıflandırılmış pozitif örneklerin toplam sayısını, tahmin edilen pozitif örneklerin toplam sayısına böleriz. Yüksek kesinlik,

pozitif olarak etiketlenmiş bir örneğin gerçekten pozitif olduğunu gösterir (az sayıda FP).

$$Kesinlik = \frac{TP}{TP + FP} \tag{4.2}$$

Duyarlılık, doğru sınıflandırılmış pozitif örneklerin toplam sayısının toplam pozitif örnek sayısına bölünmesi olarak tanımlanabilir. Yüksek duyarlılık, sınıfın doğru tanındığını gösterir (az sayıda FN).

$$Duyarlılık = \frac{TP}{TP + FN} \tag{4.3}$$

F1 ölçütü, kesinlik ve duyarlılığın ağırlıklı ortalamasıdır. Bu nedenle, bu ölçüt hem FP'leri hem de FN'leri dikkate alır. Sezgisel olarak doğruluk kadar anlaşılması kolay değildir, ancak F1, özellikle eşit olmayan bir sınıf dağılımınız varsa, doğruluktan genellikle daha kullanışlıdır. Yanlış pozitifler ve yanlış negatifler benzer değere sahipse doğruluk en iyi sonucu verir. Yanlış pozitiflerin ve yanlış negatiflerin değeri çok farklı ise hem kesinliğe hem de duyarlılığa bakmak daha iyidir.

$$F_1 = 2x\frac{Kesinlik\ x\ Duyarlılık}{Kesinlik + Duyarlılık} \tag{4.4}$$

Dört farklı sınıfa ait web sitelerinden elde ettiğimiz metinler Doc2Vec eğitimi sonucu vektörel olarak ifade edilen belgeler haline geldikten sonra, CNN, Random Forest ve Gauss Naive Bayes ile sınıflandırılmışlardır.

Sınıflandırmaların doğruluk oranları PV-DBOW ve PV-DM yöntemleri açısından farklı Doc2Vec eğitimleri ile karşılaştırılmıştır (Tablo 4.7).

Tablo 4.7: Convolutional Neural Network (CNN), Random Forest (RF), Gauss Naive Bayes (GNB) sınıflandırma yöntemlerinin doğruluk oranları

	PV – DBOW			**PV – DM**		
Doc2Vec İterasyon	**RF**	**GNB**	**CNN**	**RF**	**GNB**	**CNN**
15	%95,55	%95,81	**%96,10**	%80,35	%76,00	%79,55
50	%93,35	%95,20	%96,07	%86,70	%82,75	%87,01
75	%92,38	%94,84	%94,62	%87,65	%84,40	%88,32
100	%91,94	%95,30	%94,22	%87,04	%83,81	%88,84

Yapılan sınıflandırma sonuçları doğruluk oranlarına göre incelendiğinde, PV-DM'e kıyasla PV-DBOW'un daha yüksek sonuç verdiği görülmektedir. PV-DBOW yönteminde iterasyon değerinin artması doğruluk oranını azaltırken, diğer yöntemde doğruluk oranının artmasına sebep olmuştur. Üç sınıflandırma yöntemine ait değerlere bakıldığında en yüksek doğruluk oranı 15 iterasyonlu PV-DBOW yöntemi ile yapılan, CNN sınıflandırmasının olduğu gözlemlenmektedir. Sınıflandırma yöntemlerinin PV-DBOW için yüksek doğruluk oranının göre sıralaması; CNN, Gauss Naive Bayes, Random Forest iken, PV-DM için yüksek doğruluk oranının sıralaması; CNN, Random Forest, Gauss Naive Bayes'tir. Her iki belge gömme yönteminde de en başarılı sonuç CNN'den elde edilmiştir ancak makine öğrenmesi sınıflandırma yöntemlerinde sıralama değişmiştir.

Temel olarak, doğruluk bize bir modelin doğru şekilde eğitilip eğitilmediğini ve genel olarak nasıl performans gösterebileceğini hemen söyleyebilir. Ancak, soruna uygulanması hakkında ayrıntılı bilgi vermez. Bu yüzden daha iyi bir cevap alabilmek için kesinlik , duyarlılık ve f1 ölçütünü bilmemiz gerekir. Bu yüzden tüm sınıflandırma işlemleri için diğer başarı ölçütlerine de bakılmıştır.

Tablo 4.8: Random Forest Başarı Ölçütleri

	PV – DBOW				PV – DM			
Doc2Vec İterasyon	**Accuracy**	**Precision**	**Recall**	**F1 Measure**	**Accuracy**	**Precision**	**Recall**	**F1 Measure**
15	%95,55	%95,56	%95,56	%95,56	%80,35	%80,32	%80,06	%80,35
50	%93,35	%93,38	%93,35	%93,36	%86,70	%86,70	%86,73	%86,71
75	%92,38	%92,31	%92,35	%92,38	%87,65	%87,63	%87,66	%87,63
100	%92,92	%91,93	%91,94	%91,93	%87,04	%87,08	%87,05	%87,05

Tablo 4.9: Gauss Naive Bayes Başarı Ölçütleri

	PV – DBOW				PV – DM			
Doc2Vec İterasyon	**Accuracy**	**Precision**	**Recall**	**F1 Measure**	**Accuracy**	**Precision**	**Recall**	**F1 Measure**
15	%95,81	%95,83	%95,80	%95,80	%76,00	%76,33	%76,05	%75,95
50	%95,20	%95,18	%95,15	%95,16	%82,75	%83,13	%82,83	%82,83
75	%94,84	%94,88	%94,88	%94,85	%84,40	%84,78	%84,38	%84,45
100	%95,30	%95,33	%95,35	%95,30	%83,81	%84,23	%83,80	%83,88

Tablo 4.10: CNN Başarı Ölçütleri

	PV – DBOW				PV – DM			
Doc2Vec İterasyon	**Accuracy**	**Precision**	**Recall**	**F1 Measure**	**Accuracy**	**Precision**	**Recall**	**F1 Measure**
15	%96,10	%96,00	%96,50	%96,00	%78,55	%78,75	%78,50	%78,25
50	%95,00	%94,75	%94,50	%94,25	%89,00	%88,75	%89,25	%89
75	%94,60	%94,50	%94,00	%94,25	%88,25	%88,25	%88,00	%88,25
100	%94,20	%94,25	%94,25	%94,25	%89,00	%89,00	%89,00	%89,00

PV-DBOW sonuçlarına bakıldığında, kesinlik bakımından CNN ile en iyi sonuç elde edilmiş olup, diğer algoritmalar bu ölçüte göre Gauss Naive Bayes ve Random Forest olarak sıralanır. Ancak, bu ölçütün duyarlılık ölçütüyle ele alınması gerekir aksi halde tek başına yanlış sonuçlara götürebilir. Tablolara bakıldığında duyarlılık ölçütüne göre

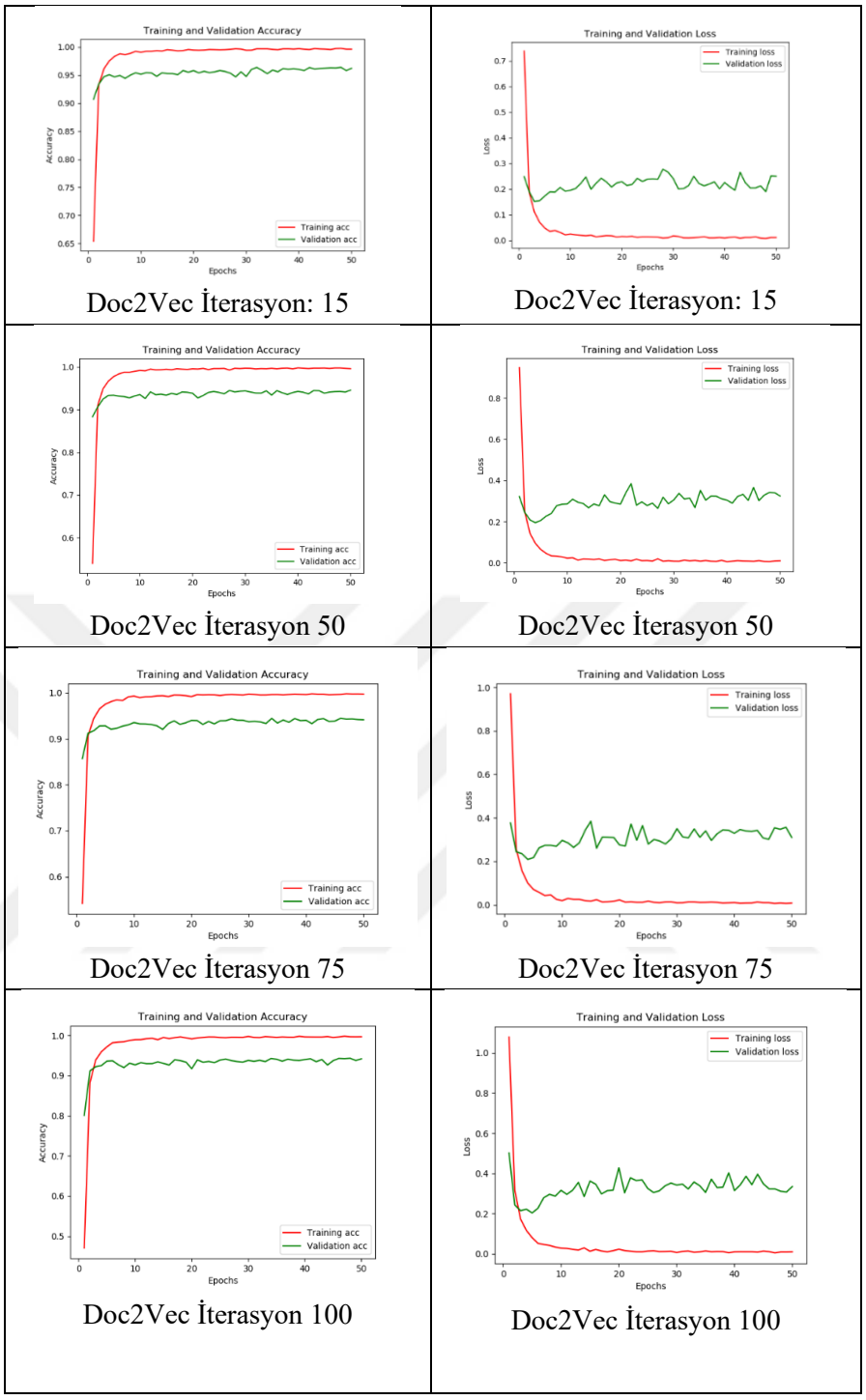
en iyi sonuç CNN olmak üzere sırasıyla Gauss Naive Bayes ve Random Forest'tır. Her iki ölçütün de birbiriyle aynı sıralamayı ortaya koyduğu görülmektedir.

PV-DM sonuçlarına bakıldığında, ise kesinlik ve duyarlılık sonuçlarında diğer yönteme göre sıralama değişmektedir. Ancak üç algoritmadan en iyi sonuç yine CNN'den elde edilmiştir. Sıralamanın devamında ise diğer yöntemin tersine bu iki başarı ölçütü için Random Forest, Gauss Naive Bayes'ten daha iyi sonuç vermiştir.

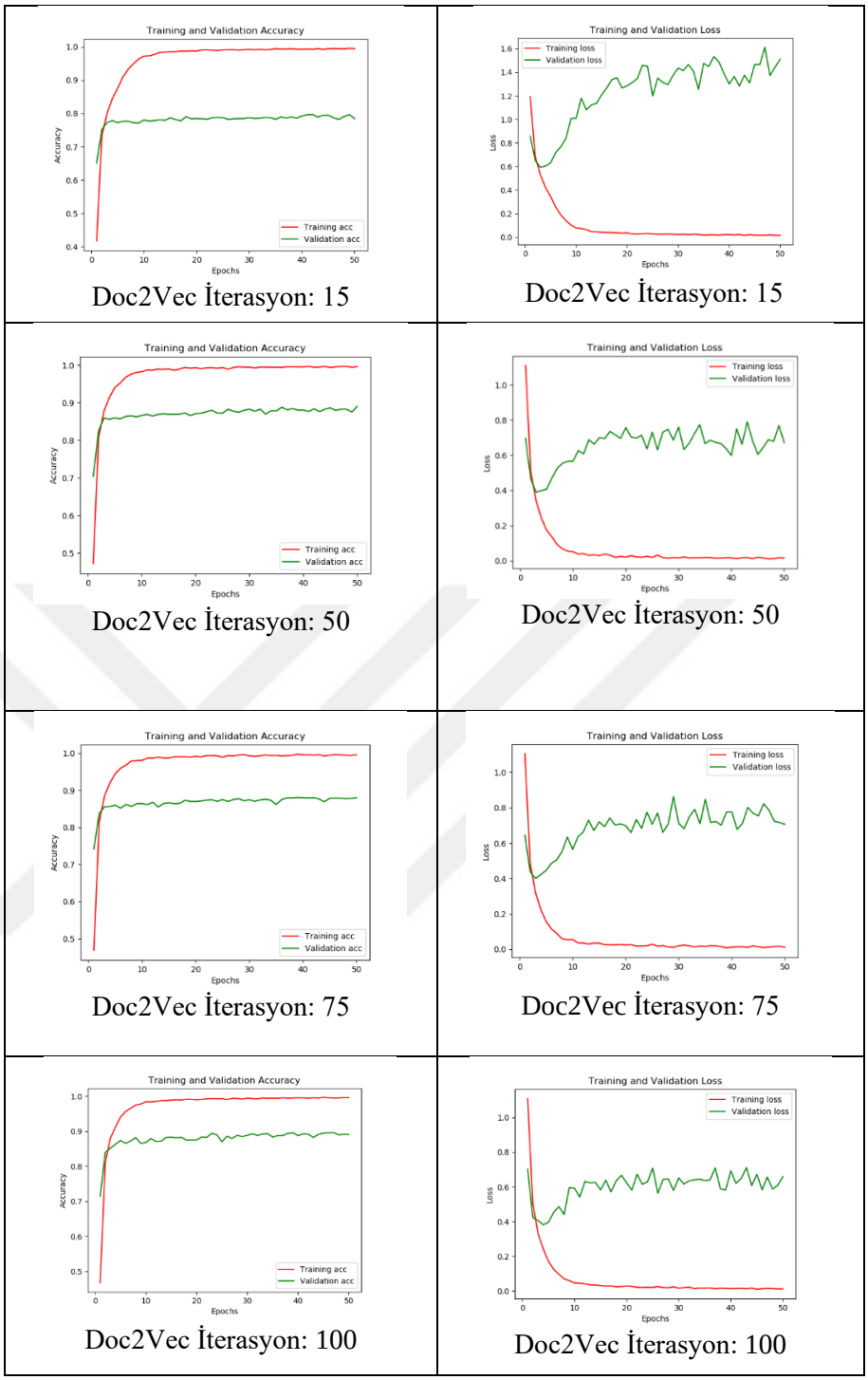
Algoritmaların sıralaması kesinlik ve duyarlılık ölçütlerinde farklı çıkabilirdi. İki kriterin de beraber değerlendirilmesi için F1 ölçütüne yani harmonik ortalamasına bakılması gerekir. F1 ölçütün diğer ölçütlerin başarı sıralamasıyla aynı sonuçları verdiği görülmektedir.

Tüm başarı kriterleri değerlendirildiğin, kesinlik, duyarlılık ve f1 ölçütündeki sıralamaların doğruluk kriter sıralamasıyla birebir aynı olduğu görülmektedir.

Buna ek olarak aşağıda CNN eğitim modeli sonuçlarına göre eğitim ve test doğruluğu ile kaybına ait grafikler verilmiştir.



Şekil 4.3: PV-DBOW yönteminin farklı Doc2Vec iterasyon değerleri ile oluşturulan doğruluk ve kayıp grafikleri.



Şekil 4.4: PV-DM yönteminin farklı Doc2Vec iterasyon değerleri ile oluşturulan doğruluk ve kayıp grafikleri.

BEŞİNCİ BÖLÜM

5. SONUÇ

Bu tez çalışmasında web sitesinin metin sınıflandırma ile hangi tipte sınıfa ait olduğunun bulunabilmesi için Doc2Vec yöntemleri olan PV-DBOW ve PV-DM uygulamalı olarak ele alınıp performansları Random Forest, Gauss Naive Bayes ve CNN ile karşılaştırılmıştır.

Ortak yönelimin tespiti için din, sağlık, spor ve üniversite sınıflarına ait web sitelerinden JSoup kütüphanesi ile İngilizce metinler elde edilmiş ve Doc2Vec modeli oluşturulmak üzere belgeler halinde kaydedilmiştir. Veri seti her sınıf için 10.000 belge olmak üzere toplamda 40.000 belge içermektedir. Veri seti %80 eğitim ve %20 test olarak sınıflandırmada kullanılmak üzere ayrılmıştır. Metin sınıflandırmanın temel adımlarından biri olan metin ön işleme aşaması ham verilerin sınıflandırmada verimli bir şekilde kullanılması açısından uygulanmıştır. Veri seti üzerinde sayısal hesaplamalarla analizler yapabilmek için metni vektör olarak temsil etmek zorunludur. Bu yüzden, öncelikle veri setindeki gereksiz veri ve gürültü giderilip daha sonra gerekli vektörel ifade etme işlemleri yapılmıştır. Bu işlem Doc2Vec'in iki farklı yöntemi ile farklı iterasyonlarla model oluşturularak yapılmıştır. Bu aşama sonrası 1024 boyutunda vektörel matris elde edilmiştir.

Eğitimi sonrası elde edilen 1024 boyutlu vektör matrisini görselleştirmek üzere, boyut azaltma yöntemleri olan PCA ve t-SNE ile 2 boyutlu hale getirilip grafiği çizilmiştir. Grafik incelendiğinde, benzer yönelime ait belgelerin birbirine daha yakın olduğu görülmüştür.

Çalışmada ortak yönelimi tespit etmek için kullanılan sınıflandırmalardan biri olan CNN'de girdi olarak kullanılmak üzere 1024 boyutlu olan örnek vektörleri önce 0-255 piksel aralığına daha sonra 32x32 boyutunda resme çevrilmiştir. Böylece 4 farklı sınıfa ait toplamda 40.000 adet resimden oluşan veri seti elde edilir. Eğitim turu 50 olarak belirlenen sınıflandırma işlemi sonucu başarı kriterleri ile birlikte doğruluk ve kayıp grafikleri oluşturulur.

Doc2Vec model eğitimi 1024 vektörel boyutunda ve farklı iterasyonlar kullanılarak "Gensim" kütüphanesi ile oluşturulmuştur. İterasyonlar; 15, 50, 75, 100'dür. Buradan

elde edilen vektörler üzerinde makine öğrenmesi sınıflandırma yöntemleri olan Random Forest ve Gauss Naive Bayes ile "Knime" platformu üzerinde başarı oranları hesaplanmıştır.

Değerlendirme aşamasında, CNN, Gauss Naive Bayes ve Random Forest başarı kriterleri olan doğruluk, kesinlik, duyarlılık ve f1 ölçütü açısından karşılaştırılmışlardır. Karşılaştırma sonucu en iyi sonuç 15 iterasyonlu PV-DBOW yöntemiyle yapılan CNN algoritmasında %96,10 doğruluk oranı ile elde edilmiştir. Doğruluk oranı en iyi sonuca göre sırasıyla Gauss Naive Bayes, Random Forest'tır. PV-DM yönteminde en iyi sonuç yine derin öğrenme yöntemi olan CNN olarak değiştmemiştir ancak PV-DBOW'a kıyasla Random Forest'ın Gauss Naive Bayes'e göre daha yüksek sonuç verdiği görülmektedir. Başarı kiterlerininden sadece doğruluk oranına bakmak bazen yeterli olmayabilir, bu yüzden diğer başarı kriterlerine de bakmak gerekir. Sınıflandırmaların kesinlik, duyarlılık ve f1 ölçütü değerlerine bakıldığında, bunlar da doğruluk oranındaki sonuçları desteklemişlerdir. Başarı değerlerine bakarken dikkat çeken başka bir durum ise, PV-DBOW yönteminde iterasyon değeri arttıkça doğruluk oranı azalırken, PV-DM'de sonuçların artmasıdır.

Bu çalışmadan elde edilen sonuçlara göre web sitelerinin ortak yönelimlerinin tespit edilmesi konusunda metin sınıflandırma yöntemleri ile yüksek doğruluk oranı elde edilmiş olup web siteleri başarılı bir şekilde ait oldukları sınıfa göre etiketlenmişlerdir. Derin öğrenme tabanlı sınıflandırıcılar, geleneksel sınıflandırıcılara göre gömme vektörlerinin kullanımı ile daha yakın veya biraz daha yüksek sınıflandırma başarılarına sahiptir. Bu nedenle, derin öğrenmeye dayalı sınıflandırıcılar ortak yönelimli web sitelerinin tespitinde kullanılan geleneksel yöntemlere bir alternatif olarak görülebilir.

Bu uygulama çerçevesinde, oluşturulan veri setinin boyutu kullanılan bilgisayarın bellek miktarına bağlıdır. Daha fazla bellek ile sınıf ve örnek sayısı artırılarak çalışmanın daha büyük verilerde uygulanması sağlanabilir. Buna ek olarak, ön işlem aşaması dile göre değiştirilerek farklı diller için de uygulama yapılabilir.

KAYNAKLAR

Abdelwahab, O. ve Elmaghraby, A. (2016). UofL at SemEval-2016 Task 4: Multi domain word2vec for Twitter sentiment classification. *Proceedings of SemEval, 164-170*.

Amasyalı, M.F., Tasköprü, H., ve Çalışkan, K. (2018). Words, Meanings, Characters in Sentiment Analysis. *Innovations in Intelligent Systems and Applications Conference (ASYU)*, Adana, 1-6.

Arabacı, M. A., vd. (2018). Detecting similar sentences using word embedding. *26th Signal Processing and Communications Applications Conference (SIU)*. IEEE.

Bartholomew, D. J., vd. (2008). Analysis of Multivariate Social Science Data. *Statistics in the Social and Behavioral Sciences Series (2nd ed.)*. Taylor & Francis. ISBN 978-1584889601.

Bengio, Y., vd. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Bermingham, Mairead L.; Pong-Wong, Ricardo; Spiliopoulou, Athina; Hayward, Caroline; Rudan, Igor; Campbell, Harry; Wright, Alan F.; Wilson, James F.; Agakov, Felix; Navarro, Pau; Haley, Chris S. (2015). "Application of high-dimensional feature selection: evaluation for genomic prediction in man". Sci. Rep. 5: 10312. Bibcode:2015NatSR...510312B. doi:10.1038/srep10312. PMC 4437376. PMI D 25988841.

Birjandtalab, J., Pouyan, M. B. ve Nourani, M. (2016). Nonlinear dimension reduction for EEG-based epileptic seizure detection. *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)* (pp. 595–598). doi:10.1109/BHI.2016.7455968. ISBN 978-1-5090-2455-1.

Blei, D., Ng, A. Y. ve Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*.

Breiman, L. (2001). “Random Forests” Machine Learning, 45, 5-32.

Brownlee, J. (2014). A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library. https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/

Ceci, M. (2005). Naive bayesian learning from structural data. Ph. D. dissertation, *Dipartimento di Informatica*, University of Bari, Italy.

cf. the Special Interest Group on language GENeration, http://www.aclweb.org/siggen

Chakrabarti, S., Dom, B. E., ve Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In Proc. of the ACM SIGMOD1998, pp. 307–318, Seattle, USA.

Chandrakumar, T., Kathirvel, R. (2016). Classifying diabetic retinopathy using deep learning architecture. *Int J Eng Res Technol*, 5(6), 19-24.

Collobert, R., vd.(2011). Natural Language processing (almost) from Scratch, Journal of Machine Learning Research, 12: 2493-2537.

Craven, M. ve Slattery, S. (2001). Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43(1-2):97–119.

Çoban, Ö. (2017). Turkish music genre classification using audio and lyrics features. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 0. doi: 10.19113/sdufbed.88303.

Dai, A. M., Olah, C. ve Le, Q. V. (2014). Document embedding with paragraph vectors. *Paper presented at Proc of NIPS 2014 in Deep Learning and Representation*. Montreal, Canada. Retrieved from https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/44894.pdf.

Ertugrul, A. M., Onal, I. ve Acarturk, C. (2017). Does the strength of sentiment matter? A regression based approach on Turkish social media. In Frasincar, F., Ittoo, A., Nguyen, L. M., & Métais, E. (Eds.), *Proceedings of International Conference on Applications of Natural Language to Information Systems* (pp. 149-155). Liège, Belgium: Springer.

Esen, E. ve Özkan, S. (2017). Analysis of turkish parliament records in terms of party coherence, in *Signal Processing and Communications Applications Conference (SIU)*, 2017 25th. IEEE, 2017, (pp. 1–4).

Ganitkevitch, J., Van Durme, B. ve Callison-Burch, C. (2013). PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of*

the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013), (pp. 758–764), Atlanta, USA.

Gashi, I., vd. (2009). An Experimental Study of Diversity with Off-the-shelf AntiVirus Engines. *Proceedings of the IEEE International Symposium on Network Computing and Applications*: 4–11.

Guyon, I., Elisseeff, A. (2003). "An Introduction to Variable and Feature Selection". JMLR. 3.

Hamel, P., Eck, D. (2010). Learning Features from Music Audio with Deep Belief Networks. *Proceedings of the International Society for Music Information Retrieval Conference*: 339–344.

Harris, Z. S. (1954). Distributional structure. Word, 10(2-3), 146-162.

Hashimoto, K., vd. (2016). Topic detection using paragraph vectors to support active learning in systematic reviews.

Hassan, A., ve Mahmood, A. (2017). Deep Learning Approach for Sentiment Analysis of Short Texts, *3rd International Conference on Control, Automation and Robotics (ICCAR)*, 705-710.

Holland SM. Principal Components Analysis (PCA). University of Georgia, 2008.

Hotelling, H (1936). "Relations between two sets of variates". *Biometrika*. *28 (3/4)*: 321–377. doi:10.2307/2333955. JSTOR 2333955.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology, 24*, 417–441, and 498–520.

https://jsoup.org/

Hu, Z., vd. (2015). Review Sentiment Analysis Based on Deep Learning. *IEEE 12th International Conference on e-Business Engineering*, Beijing, 87-94.

Huang, Q., vd. (2017). Deep Sentiment Representation Based on CNN and LSTM, *International Conference on Green Informatics (ICGI)*, 30-33.

Hughes, M., Li, I., Kotoulas, S. ve Suzumura, T. (2017). Medical text classification using convolutional neural networks. In Randell, R., Cornet, R., McCowan, C., Peek, N., & Scott, P. (Eds.), *Informatics for Health: Connected Citizen-Led Wellness and Population Health* (pp. 246-250). Amsterdam, Netherlands: IOS Press.

James, G., vd. (2013). An Introduction to Statistical Learning. *Springer*. p. 204.

Jamieson, A.R., vd. (2010). Exploring Nonlinear Feature Space Dimension Reduction and Data Representation in Breast CADx with Laplacian Eigenmaps and t-SNE. *Medical Physics. 37 (1)*: 339–351. doi:10.1118/1.3267037. PMC 2807447. PMID 20175497.

Jatana, N., ve Sharma, K. (2014). Bayesian spam classification: Time efficient radix encoded fragmented database approach. *Paper presented at the Computing for Sustainable Global Development (INDIACom), 2014 International Conference on*. 939-942.

Jolliffe I.T. Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences, *ACL*.

Karvelis, P., vd. (2018). Topic recommendation using Doc2Vec. *International Joint Conference on Neural Networks (IJCNN)*.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification, *EMNLP*.

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems* (pp. 1097-1105).

Kwok, T. Y. ve Yeung, D. Y. (1997). Constructive algorithms for structure learning in feedforward neural networks for regression problems, *Neural Networks, IEEE Transactions on, Vol. 8, No. 3*, pp. 630–645.

L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579–2605, 2008.

Lantz,B. (2013). Probabilistic Learning Classification Using Naive Bayes, Machine Learning with R, Jones, J.(ed.), Chapter 4, Packt Publishing Ltd., Birmingham, UK., 99-110.

Lau, J. H. ve Baldwi, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation.

Le, Q. V. ve Mikolov, T. (2014). T. Distributed representations of sentences and documents. In International Conference on Machine Learning.

Le, Q. ve Mikolov, T. (2014). Distributed representations of sentences and documents. In Phung, D. & Li, H. (Eds.), *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (pp. 1188-1196). Beijing, China: PMLR.

LeCun, Y., vd. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1:541-551.

LeCun, Y., vd. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261, 276.

LeCun, Y., vd. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

LeCun, Y., vd. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

Lindsay, I. S. (2002). A tutorial on Principal Component Analysis. Smith, L.I. A tutorial on principal components analysis. Cornell Univ. USA 2002, 51, 52.

Mann, W. C. (1983). An overview of the PENMAN text generation system. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 261-265). AAAI. Also appears as USC/Information Sciences Instituteü RR-83-114.

Martinez, A.M. ve Kak, A.C. (2001). PCA versus LDA, IEEE Trans. *Pattern Analysis and Machine Intelligence, vol. 23, no. 2* (pp. 228-233).

McCallum, A. K., vd. (1998). Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of ICML-98, 15th International Conference on Machine Learning (Madison, WI, 1998)*, 359–367.

Mikolov, T., vd. (2013). Distributed Representations of Words and Phrases and their Compositionality. Accepted to NIPS.

Mikolov, T., vd. (2013). Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems* (pp. 3111–3119).

Mikolov, T., vd. (2013). Efficient estimation of word representations in vector space. Paper presented at the International Conference on Learning Representations: Workshops Track. Scottsdale, Arizona. *arXiv preprint arXiv:1301.3781*.

Mitchell, T. M., vd. (1997). Machine learning. WCB, Vol. 8, McGraw-Hill Boston, MA.

Mukherjee, S., ve Sharma, N. (2012). Intrusion detection using naive Bayes classifier with feature reduction. *Procedia Technology*, 4, 119-128.

Nielsen, M. A. (2015). Neural networks and deep learning (Vol. 25). USA: Determination press.

Oh, H. J., Myaeng, S. H. ve Lee, M. H. (2000). A practical hypertext categorization method using links and incrementally available class information. *In Proc. of the 23rd ACM SIGIR2000*, pp. 264–271, Athens, GR.

Ouyang, X., vd. (2015). Sentiment analysis using convolutional neu- ral network, in: Computer and Information Technology; Ubiquitous Comput- ing and Communications; Dependable, Autonomic and Secure Computing; Per- vasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), *2015 IEEE International Conference* on, *2015*, pp. 2359–2364.

Öztemel, E. (2003). Yapay Sinir Ağları. İstanbul: Papatya, s.15-18.

Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". *Philosophical Magazine. 2 (11)*: 559–572. doi:10.1080/14786440109462720.

Pennington, J., Socher, R. ve Manning, C. (2014). Glove: Global vectors for Word representation. In Wu, D., Carpuat, M., Carreras, X., & Vecchi, E. M. (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Vol. 14 (pp. 1532-1543). Doha, Qatar: Association for Computational Linguistics.

Pennington, J., Socher, R. ve Manning, C. (2014). R. JeffreyPennington and C. Manning. Glove: Global vectors for word representation.

Ravı, D., vd. (2017). Deep learning for health informatics. *IEEE journal of biomedical and health informatics*, 21(1), 4-21.

Ravı, D., vd. (2017). Deep learning for health informatics. *IEEE journal of biomedical and health informatics*, 21(1), 4-21.

Rehurek, R. ve Sojka, (2010). P. Software framework for topic modelling with large corpora. In Witte, R., Cunningham, H., Patrick, J., Beisswanger, E., Buyko, E., Hahn, U., Verspoor, K., & Coden, A. R. (Eds.), *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45-50). Valletta, Malta: ELRA.

Rehurek, R. ve Sojka, P. (2010). Software framework for topic modelling with large corpora. LREC.

Safalı, Y., vd. (2019). Türkçe Akademik Çalışmaların Doc2Vec Modeli Kullanılarak Derin Öğrenme Tabanlı Sınıflandırılması. *International Artificial Intelligence and Data Processing Symposium (IDAP).*

Sankar, M., Batri, K., Parvathi, R. (2016). Earliest diabetic retinopathy classification using deep convolution neural networks. pdf. Int. J. Adv. Eng. Technol.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.

Sermanet, P., vd. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint* arXiv:1312.6229.

Severyn, A., ve Moschitti, A. (2015). Twitter Sentiment Analysis with Deep Convolutional Neural Networks, *SIGIR*.

Skymind.ai. A Beginner's Guide to Eigenvectors, Eigenvalues, PCA, Covariance and Entropy [Web log post]. Retrieved from https://skymind.ai/wiki/eigenvector

Smith, L.I. A tutorial on principal components analysis. Cornell Univ. USA 2002, 51, 52. [Google Scholar]

Steven, M. H. (2008). Principal Components Analysis. Holland SM. Principal Components Analysis (PCA). University of Georgia.

Stigler, S. M. (1982). Thomas Bayes's bayesian inference. *Journal of the Royal Statistical Society. Series A (General)*, 250-258.

Su, Z., Xu, H., Zhang, D. ve Xu, Y. (2014). Chinese sentiment classification using a neural network tool—Word2vec. Paper presented at the *2014 International*

Multisensor Fusion and Information Integration for Intelligent Systems Conference. Beijing, China. doi: 10.1109/MFI.2014.6997687.

Szegedy, C., vd. (2015). Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Şeker, S. E. (2015). Metin Madenciliği (Text Mining), *YBS Ansiklopedi v. 2, is. 3* (pp. 30-32).

Şen, M. U., ve Erdoğan, H. (2014). Learning word representations for Turkish. *22nd Signal Processing and Communications Applications Conference (SIU)*, Trabzon, 1742-1745.

Turing, A. (1950). Computing Machinery and Intelligence,

Turing, A. M. (1950). Computing Machinery and Intelligence, Mind, *Vol. 59, No. 236* (pp. 433–460).

van der Maaten, L. ve Hinton, G. E. (2008). Visualizing data using t-SNE. J. Mach. Learn. Research 9, 2579–2605.

van der Maaten, L.J.P.; Hinton, G.E. (Nov 2008). "Visualizing Data Using t-SNE" (PDF). *Journal of Machine Learning Research*. 9: 2579–2605.

Vo, Q., vd. (2017). Multi-channel LSTM-CNN model for Vietnamese sentiment analysis, *9th International Conference on Knowledge and Systems Engineering (KSE)*, Hue, 24-29.

Wallach, I. ve Liliean, R. (2009). The Protein-Small-Molecule Database, A Non-Redundant Structural Resource for the Analysis of Protein-Ligand Binding. *Bioinformatics. 25 (5)*: 615–620. doi:10.1093/bioinformatics/btp035. PMID 19153135.

Wieting, J., vd. (2016). Towards universal paraphrastic sentence embeddings. In *Proceedings of the International Conference on Learning Representations*, San Juan, Puerto Rico.

Yang, S., and Xia, Z. (2016). A convolutional neural network method for Chinese document sentiment analyzing. *2nd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 308-312.

Yang, Y., Slattery, S. ve Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18(2-3):219–241.

YouTube. (2013, November 6). Visualizing Data Using t-SNE [Video File]. Retrieved from https://www.youtube.com/watch?v=RJVL80Gg3lA.