T.C.
KASTAMONU ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖGRENMESİ TEKNİKLERİ İLE SOSYAL MEDYA KULLANIMI ÜZERİNE BİR DUYGU ANALİZİ ÇALIŞMASI

Mohamed Guma Ibrahim BODEA

Danışman	**Dr. Öğr. Üyesi İsmail YILDIZ**
Jüri Üyesi	**Doç. Dr. Göksal BİLGİCİ**
Jüri Üyesi	**Dr. Öğr. Üyesi Erman UZUN**

YÜKSEK LİSANS TEZİ
MALZEME BİLİMİ VE MÜHENDİSLİĞİ ANA BİLİM DALI

KASTAMONU – 2020

TEZ ONAYI

Mohamed Guma Ibrahim BODEA tarafından hazırlanan "**Makine Ögrenmesi Teknikleri ile Sosyal Medya Kullanımı Üzerine Bir Duygu Analizi Çalışması**" adlı tez çalışması **23/1/2020 tarihinde** aşağıdaki jüri üyeleri önünde savunulmuş ve **oy birliği** ile Kastamonu Üniversitesi Fen Bilimleri Enstitüsü **Malzeme Bilimi ve Mühendisliği Ana Bilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman	Dr. Öğr. Üyesi İsmail YILDIZ Kastamonu Üniversitesi	.....
Jüri Üyesi	Doç. Dr. Göksal BİLGİCİ Kastamonu Üniversitesi	.....
Jüri Üyesi	Dr. Öğr. Üyesi Erman UZUN Mersin Üniversitesi	.....

Enstitü Müdürü Doç. Dr. Nur BELKAYALI .....

TAAHHÜTNAME

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildirir ve taahhüt ederim.

Mohamed Guma Ibrahim BODEA

ÖZET

Yüksek Lisans Tezi

MAKİNE ÖGRENMESİ TEKNİKLERİ İLE SOSYAL MEDYA KULLANIMI ÜZERİNE BİR DUYGU ANALİZİ ÇALIŞMASI

Mohammed Guma Ibrahim BODEA
Kastamonu Üniversitesi
Fen Bilimleri Enstitüsü
Malzeme Bilimi ve Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi İsmail YILDIZ

Son yıllarda farklı platformlarda insanlar tarafından yazılan metinlerin yaygınlaşması ve özellikle erişimin de artması nedeniyle, söz konusu metinleri analiz etmek için makine öğrenmesi (İng. machine learning) tekniklerinin kullanılması belirgin bir ilgiye mazhar olmaktadır. Bu metinler insanlar tarafından yazıldığı için, doğru bilginin elde edilmesi, Doğal Dil İşleme (NLP) olarak bilinen yoğun bir işlem süreci gerektirir. Burada kullanılan tekniklerin karşılaşacağı başlıca zorluk, bu metinlerde bulunan çok fazla miktardaki bilgi ve kullanılan kelimeler gibi öznitelikler ve çıkarımı yapılmak istenen bilgi arasındaki karmaşık ilişkilerdir. Bu bağlamda, bilgi çıkarımı üzerinde hiç etkisi olmayan veya olumsuz etkisi olan kelimelerin ihmal edilmesi, çok boyutluluğu azaltarak ve bilgi sunumunun verimliliğini artırarak NLP tekniklerinin performansını önemli ölçüde artırabilir.Bu çalışmada, kelimelerin sınıflandırıcıların performansı üzerindeki etkisi hakkında elde edilen bilgileri temsil eden vektörleri ve aynı kelimelerin duygusal anlamını kullanan yeni bir öznitelik belirleme yöntemi önerilmektedir. Önerilen yöntemde, takviyeli öğrenim yoluyla ve veri kümesindeki her bir kelimeyi kaldırmanın etkisini izlemeye dayalı olarak eğitilen yapay bir sinir ağı kullanılmaktadır. Bu kelimeleri temsil eden vektörleri elde etmek için kelime kalıplama (İng. word embedding) kullanılır, bu sayede; bir kelime eğitim veri kümesinde yer almasa dahi, kendisi için üretilen vektörün değerlerine ve eğitim sırasında kullanılan, anlamca bu kelimeye en benzer kelimelere bağlı olarak sıralaması (İng. rank) tahmin edilebilir. Dolayısıyla, ne bütüncedeki herhangi bir kelime için, ne de bütünceye daha sonra eklenebilecek herhangi bir yeni kelime için karmaşık istatistiksel hesaplamalara gerek kalmaz.

Yapılan değerlendirme sonucunda, önerilen yöntemin eğitim kümesinde yer almayan her kelimenin sıra veya derecesini % 94.61 doğrulukla hesap etme yeteneği olduğu görülmüştür. Ayrıca, bahsedilen sıra ve derecelere dayalı özellik seçiminin; Destek Vektör Makinesi (SVM), Naïve Bayes (NB) ve Rastgele Orman (RF) gibi metini temsil etmek için sayı vektörlerini kullanan ve Evrişimli Sinir Ağı (CNN), Uzun-Kısa Süreli Bellek (LSTM) ve Geçitli Tekrarlanan Birim (GRU) gibi kelime kalıplamaya dayanan farklı sınıflama türlerinin performansını arttırdığı görülmüştür. Ayrıca, GRU sınıflandırıcı, %95.54 doğrulukla, literatürde yer alan diğer sınıflandırıcılara ve en gelişmiş yöntemlere kıyasla en yüksek performansı vermiştir.

Anahtar Kelimeler: Makine Öğrenmesi; Sosyal Medya; Metin Kategorizasyonu; Yapay Sinir Ağları; Takviye Öğrenme

2020, 75 sayfa
Bilim Kodu: 91

ABSTRACT

MSc. Thesis

A STUDY ON SENTIMENT ANALYSIS ON SOCIAL MEDIA USING MACHINE LEARNING TECHIQUES

Mohammed Guma Ibrahim BODEA
Kastamonu University
Graduate School of Natural and Applied Sciences
Department of Material Science and Engineering

Supervisor: Assist. Prof. Dr. İsmail YILDIZ

Abstract: In recent years, the use of machine learning techniques to analyze texts written by humans is attracting significant attention, according to the wide availability of these texts and their ease of access. As these texts are written by humans, the extraction of accurate knowledge requires intensive processing, known as Natural Language Processing (NLP). The main challenge that these techniques face is the enormous amount of information available in these texts and the complex relations among the features, i.e. words, and the knowledge required to be extracted. Accordingly, eliminating the words that has negative or no influence on the knowledge extraction can significantly improve the performance of NLP techniques, by reducing dimensionality and improving the efficiency of knowledge representation.

In this study, we propose a new feature selection technique that uses vectors that represent the sentimental meaning of words and knowledge extracted about the influence of words on the performance of the classifiers. The proposed method uses an artificial neural network that is trained using reinforcement learning by monitoring the influence of removing each word in the training dataset. Word embedding is used to provide the vectors that represent these words, so that, even if a word is not included in the training, its rank can be predicted by the proposed method depending on the values of the vector generated for it and the knowledge about the most similar words that are considered during the training. Accordingly, no complex statistical computations are required for each word in the corpus, as well as any new words that can be added to the corpus in the future.

The evaluation of the proposed method has shown its ability to predict the rank of each word that is not included in the training with 94.61% accuracy. Moreover, feature selection based on these ranks has been able to improve the performance of different types of classifiers, such as the Support Vector Machine (SVM), Naïve Bayes (NB) and Random Forest (RF), which use count vectors to represent the text, as well as the Convolutional Neural Network (CNN), Long- Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) classifiers, which rely on the word embedding vectors for the classification. Moreover, the GRU classifier has been able to achieve the highest performance, with 95.54% accuracy, compared to the other classifiers and state-of-the-art methods in the literature.

Keywords: Machine Learning; Social Media; Text Categorization; Artificial Neural Networks; Reinforcement Learning.

2020, 75 pages
Science Code: 91

TEŞEKKÜR

Öncelikle, bizlere bildilerimizi öğreten, sağlık ve sabır veren Yüce Allah'a şükürler olsun.

Bu çalışmayı başarıyla tamamlamamda katkı sağlayan;

Başta danışmanım Sayın Dr. Öğr. Üyesi İsmail YILDIZ'a minnettarım, mükemmel yol göstericiliği, bilgisini paylaşma yeteneği ve bu çalışmanın her adımında yardım ettiği için de ayrıca teşekkür ederim. Bana bu eğitim fırsatını verdiği için ülkem Libya'ya minnettarım. Kastamonu Üniversitesi çalışanlarına da teşekkürlerimi iletmek istiyorum. Özellikle annem, babam ve sevgili eşim başta olmak üzere bütün aileme, beni destekleme konusundaki kararlılıkları için teşekkür ederim.

Son olarak ancak son derece önemli olarak, bana yardım etmiş olan bütün arkadaşlarıma da teşekkür ederim.

Mohammed Guma Ibrahim BODEA
Kastamonu, Ocak, 2020

İÇİNDEKİLER

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

NLP	Doğal Dil İşleme (Natural Langauge Processing)
BoW	Kelime Torbası (Bag of Words)
OHE	One-Hot-Encoded
GLoVe	önerilen Global Vektörlerdir
AI	Yapay Zeka (Artificial Intelligence)
ML	Machine Learning
DL	Derin Öğrenme (Deep Learning)
SVM	Destek Vektör Makinesi (Support Vectore Machine)
RF	Rastgele Orman (Random Forest)
NB	Naive Bayes
CNN	Evrişimli Sinir Ağları (Convolutional Neural Network)
RNN	Tekrarlayan Sinir Ağları (Recurrent Neural Network)
LSTM	Uzun-Kısa Süreli Bellek (Long- Short-Term Memory)
GRU	Geçitli Tekrarlanan Birim (Gated Recurrent Unit)
STSTd	Stanford Twitter Sentiment Test dataset
GPU	Graphical Processing Unit
ÖB	Öznitelik Belirleme

ŞEKİLLER DİZİNİ

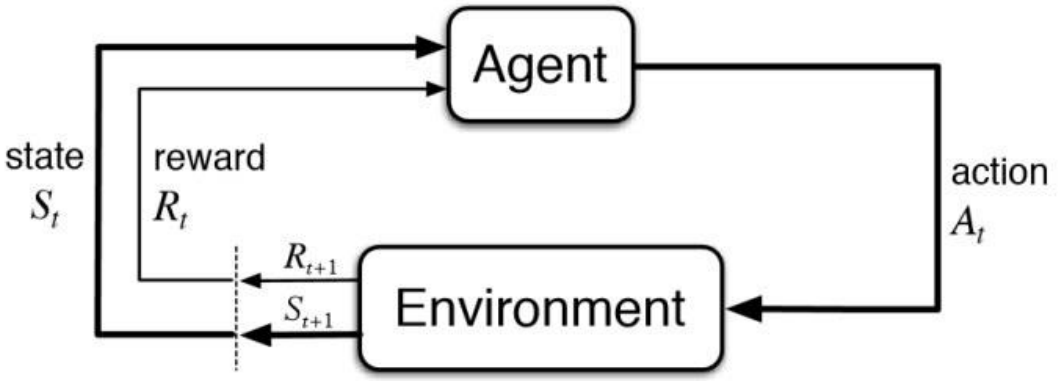
TABLOLAR DİZİNİ

1. GİRİŞ

İnternete erişim kolaylığı ve dijital cihaz kullanımının çok artması içinde bulunduğumuz dijital çağı ortaya çıkarmıştır ve bu çağda çoğu belge veya hizmet dijital formatta sunulmaktadır. Bu bağlamda, bilimsel makaleler, haberler, e-postalar ve hatta doğal dilde yazılmış olan konuşmalar gibi çeşitli belgeler internet ortamına yüklenmektedir [1-3]. Dolayısıyla, bu metinlerin otomatik olarak analiz edilmesi ve bu metinlerden kullanışlı bilgi çıkarımı büyük önem kazanmıştır. Bu tür bir bilgi çıkarımı, makine öğrenmesi teknikleri kullanılarak gerçekleştirilmektedir [4, 5].

Makine öğrenmesi (*İng.* machine learning-ML), bilgisayarların belirli bir ortamdan bilgi çıkarımı yapmasını sağlayan teknikleri araştıran disiplindir. Uygulandığı sahayla arasındaki etkileşime bağlı olarak makine öğrenmesi teknikleri üç ana kategoriye ayrılmaktadır; denetimsiz, denetimli ve takviyeli öğrenme [6, 7]. Denetimsiz ve denetimli kategorilerde, makine öğrenmesi tekniklerinin bilgi çıkarımını gerçekleştirme aşaması için, ortamdan toplanan örnekleri içeren bir veri kümesi gereklidir. Bu veri kümesindeki her örnek bir öznitelikler kümesi kullanılarak tanımlanır, bu öznitelikler için atanan değerler girdiyi karakterize eder. Özniteliklerin değerleri dışında ek bilgiye ihtiyaç duyan denetimsiz yöntemlerin aksine, denetimli makine öğrenmesi yöntemleri, o ortamdaki bir uzmanın bilgisini temsil eden ek bilgiye ihtiyaç duyar. Denetimli öğrenme yöntemlerinin rolü, verilerin içinde yer alan ve uzmanın sağladığı ek bilgilerle ilişkili olan örüntüleri tanımak ve çıkarımını yapmaktır. Daha sonra bu bilgi, çıkarımı yapılan bilgi ile öznitelik değerlerini karşılaştırarak girdi için uygun bilgileri otomatik olarak tahmin etmek için kullanılabilir [8, 9].

Takviyeli öğrenmede, makine öğrenmesi tekniği, çevresiyle etkileşim kurmak için bir aracı kullanır, burada o anki durumuna bağlı olarak ortama yönelik eylemler gerçekleştirir. Şekil 1.1'de gösterildiği gibi, çevre, belirli bir durumda aracıya, yürütülen eylemin kalitesini temsil eden ve ödül olarak bilinen geri bildirim gönderir. Eğitim sırasında ortamı aracıya tanımlayan, her bir eylem için beklenen ödül için bir fonksiyona yaklaşımda bulunulur. Böylece, aracı çevre ile en uygun etkileşimi sağlamak için ödülü en üst düzeye çıkaracak eylemi seçer [10, 11].



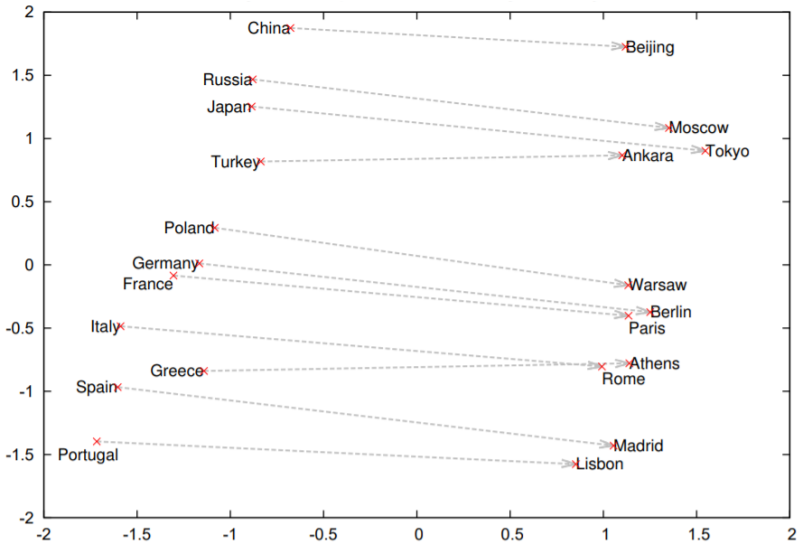
Şekil 1.1. Takviyeli öğrenmede aracıyla çevre arasında etkileşim.

Sınıflandırma yaygın olarak kullanılan denetimli makine öğrenmesi alanlarından biridir; burada örneklerin öznitelik değerleri ve her bir örneğin ait olduğu kategori ile ilişkileri araştırılır ve çıkarımları yapılır. Her örneğin kategorisi, uzman tarafından veri kümesine manuel olarak eklenir. Bu nedenle, sınıflandırma denetimli makine öğrenmesidir. Bilginin çıkarımından sonra, sınıflandırıcı gelecekteki girdilerin kategorisini öngörebilir, bu da o kategorinin karakteristik özelliklerine bağlı olarak o girdinin davranışı veya o girdi için gerek duyulan işlem süreçlerinin tahmininde kullanılabilir [12, 13].

Otomatik Metin Sınıflandırma (*İng.* Automatic Text Classification-ATC), her bir girdideki metinleri anlamına göre sınıflandırmak için yaygın şekilde kullanılan yöntemlerden biridir. Diğer sınıflandırma uygulamalarına benzer şekilde, ATC için kullanılan sınıflandırıcıların girdilerinin genellikle sayısal olması gereklidir [14]. Ayrıca, her girdiyi tanımlayan öznitelik sayısı sabit olmalıdır, böylece sınıflandırıcı tarafından çıkarımı yapılan bilgi istikrarlı ve düzenli olacak ve gelecekteki girdilerin sınıflarını tahmin etmek için kullanılabilecektir. Bu bağlamda, ATC'nin karşılaştığı temel iki zorluk, metin verilerinin sayısal hale dönüştürülmesi ve her bir dokümanı tanımlayacak öznitelik sayısının azaltılmasıdır [15].

Metin verilerini sayısal bir formata dönüştürmek için Sayma Vektörleri (*İng.* Count Vectors-CV) ve Terim Frekansı-Ters Belge Frekansı (*İng.* Term Frequency-Inverse Document Frequency, TF-IDF) gibi çeşitli teknikler önerilmiştir [16, 17]. Ancak, bu yöntemler sonucu ortaya çıkan vektörün boyutu, bütüncedeki farklı kelime sayısına

eşit olmaktadır, bunun doğal neticesi de yoğun işlem gerektiren çok büyük vektörlerin elde edilmesidir. Son zamanlarda, sinir ağları kullanımının hızla yaygınlaşması ve bu ağların farklı alanlarda belirgin bir şekilde iyi performans ortaya koymasıyla birlikte, bu ağlar bütüncedeki her bir kelimeyi sabit boyutlu bir vektöre dönüştürmede, yani kelime kalıplama işleminde kullanılmaya başlamıştır [18]. Bu yaklaşım, Şekil 1.2'de gösterildiği gibi, vektörler arası mesafeyi ölçerek kelimeler arası benzerliğin ölçülmesine olanak sağlamaktadır. Nihayetinde, daha doğru temsil sayesinde ATC yöntemlerinin performansı önemli ölçüde gelişmiştir.



Şekil 1.2. Kelime kalıplama yaklaşımıyla, kelimeler arası görecelilik ve kelimelere atanan değerlerin gösterimi [19].

ATC'de sınıflandırma işlemine tabi tutulacak olan verinin boyutu bütüncedeki farklı kelime sayısıyla doğru orantılı olduğu için, bütünce içerisinde yer alan ancak kullanışlı bir bilgi içermeyen kelimelerin elenmesi ve o şekilde işlenmesi sınıflandırıcının performansını önemli ölçüde etkilemektedir [20, 21]. Dolayısıyla, sınıflandırma aşamasında kelimelerin önemlerini ölçümlemek için farklı öznitelik seçim teknikleri önerilmiştir. Ancak, bu kelimelerin anlamını hesaba katmayan Bilgi Kazanımı (Information Gain, yani iki olasılık dağılımı arasındaki doğal uzaklık metriği-IG) gibi mevcut öznitelik belirleme teknikleri her bir kelimenin önemini ölçmek için istatistiksel yaklaşımları kullanmaktadır [22, 23].

1.1. Problemin Ortaya Konması

Dijital doküman sayısının hızla artmasıyla birlikte ATC, farklı uygulama alanlarında büyük ilgi görmektedir. Sınıflandırmada kullanılan çoğu tekniğin gereksinimleri doğrultusunda, her bir doküman sabit boyutlu bir sayısal vektöre dönüştürülmelidir. Literatürde bu vektörlerin nasıl elde edileceğine dair farklı yöntemler verilmektedir, bu yöntemlerin uygulanması sonucu elde edilen vektörün büyüklüğüyle bütüncedeki farklı kelime sayısı doğru orantılı olmaktadır. Sınıflandırıcıların performansını artırmak amacıyla sınırlı sayıda kelime seçimini netice verecek öznitelik seçme teknikleri uygulanır, bu şekilde tahminlerin kalitesi üzerinde hiçbir etkisi olmayan veya olumsuz etkisi olan kelimeler elenir. Bununla birlikte, mevcut teknikler, bütüncedeki her bir kelimenin önemini ölçümlemek için istatistiksel yaklaşımlara dayanmaktadır; bu da, bütüncedeki her bir sözcük için tüm veri kümesinin yoğun bir şekilde işlenmesini gerektirmektedir.

1.2. Çalışmanın Amacı

Bu çalışmada, takviyeli öğrenime ve kelime kalıplamaya dayanan yeni bir öznitelik belirleme yöntemi önerilmektedir. Burada eğitim aşamasında takviyeli öğrenme kullanılarak her bir kelimenin önemini tahmin etmek için sinir ağı uygulanır. Herhangi bir kelimeyi temsil eden vektör, aracının durumu ve bu özniteliği tutmak veya bırakmak gibi seçeneklerle beraber önerilen sinir ağına iletilir. Sınıflandırıcının performansına bağlı olarak, aracı, tutulması veya bırakılması gereken önemli kelimeleri öğrenir. Bununla birlikte, mevcut tekniklerden farklı olarak, önerilen yöntemde eğitim kümesinde yer almayan yeni bir kelimenin önemi tahmin edilebilmektedir; bu da o kelimenin vektörü ve eğitimden çıkarımı yapılan bilgiye, yani bitişik kelimelerin önemli olup olmadığına dayalı olarak yapılmaktadır. Dolayısıyla, önerilen yöntem daha hızlı karar alabilir çünkü bunun için gerekli olan tek girdi o kelimeyi temsil eden vektördür, bu vektör ise sinir ağı tarafından kelime kalıplamayla elde edilmektedir.

1.3. Çalışmanın Önemi

Mevcut metinlerde yer alan çok fazla bilginin var olması, makine öğrenmesi tekniklerinin bu metinleri analiz etmede ve bilgi çıkarımında hızla artan kullanımı ile,

mevcut istatistiksel tabanlı öznitelik belirleme yöntemlerinin kullanımı yoğun işlem gerektirir hale gelmiştir. Bununla birlikte, kelime kalıplama için son yıllarda yapay sinir ağlarının kullanılması, her bir kelimenin duygusal anlamını yansıtan vektörlerin elde edilmesini mümkün kılmıştır. Bu çalışmada, söz konusu vektörler kullanılarak, her bir kelimenin anlamı ve diğer kelimelerden çıkarımı yapılan bilgileri doğrudan temel alan, yine aynı kelimenin sıralamasını ölçmek için yeni bir yöntem önerilmiştir. Bu nedenle, bu kelimeleri sıralamak için, önemli ölçüde daha az hesaplama gerekli olmaktadır; bu sayede de bilgi çıkarımı yöntemlerinin performansını iyileştirecek olan hızlı öznitelik belirleme mümkün olmaktadır. Ayrıca, önerilen yöntem tüm metni tekrar analiz etmeye gerek kalmaksızın yeni bir kelimeyi sıralama yeteneğine sahiptir. Metini temsil etmek için kullanılan yönteme bakılmaksızın bu yöntemin söz konusu kelimeleri sıralama yeteneğini ortaya koymak amacıyla, önerilen öznitelik belirleme yöntemi kullanılarak farklı metin kategorizasyonu tekniklerindeki iyileştirmeler ölçümlenmektedir.

1.4. Tez Düzeni

Tezin geri kalan kısmının organizasyonu şu şekildedir:

- Çalışma konusu ve önerilen metotta kullanılan bazı tekniklerle alakalı literatür taraması İkinci Bölüm'de verilmektedir.
- Önerilen metot Üçüncü Bölüm'de detaylı olarak açıklanmaktadır.
- Önerilen metodun performansı ve bu bu performansın ölçümü için yapılan deneyler Dördüncü Bölüm'de anlatılmaktadır.
- Yapılan deneylerin sonuçları ve önerilen metodun avantajları Beşinci Bölümde yer almaktadır.
- Tezin sonuç kısmı ise Altıncı Bölüm'de yer almaktadır.

2. LİTERATÜR TARAMASI

İnsanlar tarafından yazılan metinlerin duygusal analizinin oldukça karmaşık olması dolayısıyla, Doğal Dil İşleme (NLP) için ancak karmaşık öznitelikleri tespit etme yeteneğine sahip teknikler kullanılabilir. Bu bağlamda, makine öğrenmesi teknikleri bu amaç için yaygın şekilde kullanılmaktadır. NLP duygusal analiz için, makine öğrenmesi teknikleri tarafından denetlenen sınıflandırma yöntemleri kullanmaktadır. Eğitim sırasında, sınıflandırıcı her sınıfın veya kategorinin girdilerindeki örüntüleri tanır ve bu örüntüler gelecekteki girdilerde kategori tahmini için araştırılır. Ancak, metinlerin çok boyutluluğu ve sınıflandırma yöntemlerinin çoğu için sayısal temsil ihtiyacı dolayısıyla, metin ön işlemeye tabi tutulmalı ve sayısal vektörlere dönüştürülmelidir.

2.1. Metin Ön İşleme

İnsanlar tarafından yazılan metinlerin belirli karakteristik özellikleri vardır, bu metinlerde yanlış yazılmış kelimeler, etkisiz kelimeler (*İng.* stop words) veya kullanıcı adları gibi anlamsız dizeler de bulunabilir. Ayrıca, belirli bir kelime için birden çok formatın varlığı metinin çok boyutluluğunu artırabilir; örneğin fiillerin geçmiş zamana, şimdiki zamana veya geniş zamana göre farklı çekimleri aslında aynı duygusal anlama sahiptir. Bütün bu nedenlerden ötürü, NLP uygulamalarında metinler yaygın olarak aşağıdaki yöntemlerle ön işleme tabi tutulmaktadırlar:

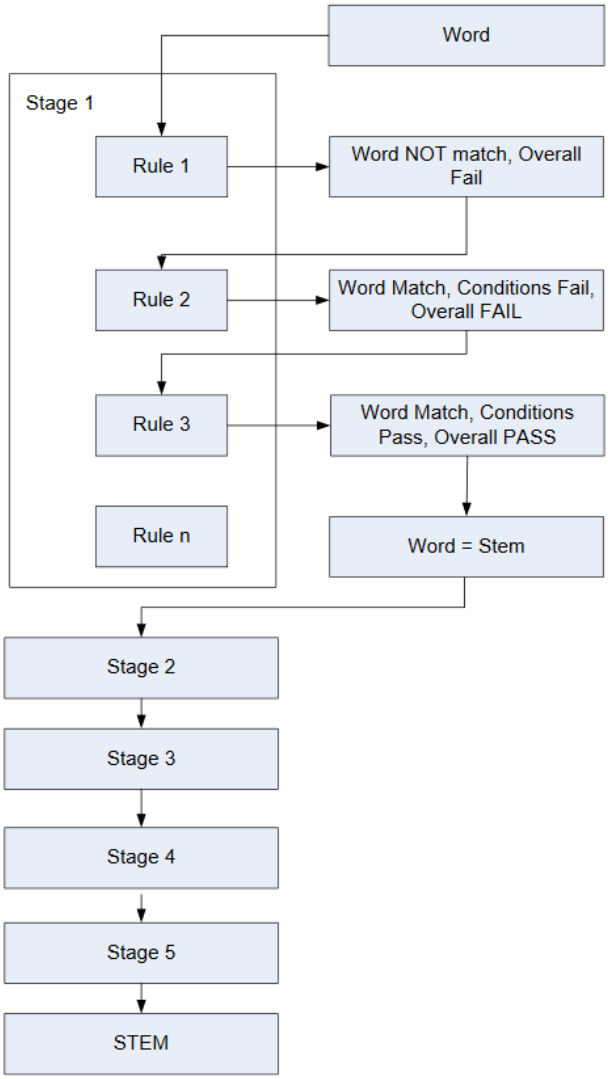
2.1.1. Çok Boyutluluğun Azaltılması

Bir bütüncenin çok boyutluluğu, içerdiği farklı kelimelerin toplam sayısı ile ölçülür. Bu nedenle, farklı kelime sayısının azaltılması, bütünce içindeki toplam kelime sayısına bakılmaksızın çok boyutluluğu azaltabilir. ‘Is’, ‘was’, ‘the’, ve ‘a’ gibi etkisiz kelimeler, insanlar tarafından yazılan metinlerde yaygın olarak kullanılır, ancak bu kelimelerin cümlenin duygusal anlamı üzerinde gerçekte bir etkisi yoktur. Böylece, NLP yönteminin performansını etkilemeden çok boyutluluğu azaltmak için bu kelimeler bütünceden çıkarılır. Ayrıca, metinler farklı miktarları tanımlamak için kullanılan sayılar da içerir. Sayının konumu NLP tekniği üzerinde bir etkiye sahip olsa da, o sayının kesin değeri daha az önemlidir, yani değerinden bağımsız olarak bir

sayının varlığını tanımak önemlidir. Böylece, sayısal biçimde yazılmış sayı ve rakamlar, farklı bir kelimeyle, örneğin "sayı" kelimesiyle değiştirilir.

Ayrıca kelime kökenine inme, türetilmiş kelimeleri cümlede kullanmaktansa, her kelimenin kökü alınarak farklı kelime sayısını azaltmak için kullanılır. Böylece, çok boyutluluk yani farklı kelimelerin sayısı azalırken, kelimenin duygusal anlamı korunmuş olur. Porter Stemmer [24] İngilizce bütünceleri işlemek için en yaygın kullanılan kelime kökenine inme yöntemlerinden biridir [25]. Kelime kökenine inme işlemi, her bir kelimeyi sırasıyla beş aşamadan geçirerek gerçekleştirilir. Şekil 2.1'de gösterildiği gibi, her bir aşama, kurallardan herhangi birine uyulması durumunda aşamanın sonlandırıldığı bir dizi kuraldan oluşur. Beş aşamanın her birinin amacı:

1. İsimlerdeki çoğul eklerinin ve fiilerdeki zaman eklerinin kaldırılması, örneğin 'cars' yerine 'car' veya 'drawing' yerine 'draw'.
2. Yaygın olarak kullanılan eklerin kaldırılması, örneğin 'functional' yerine 'function' veya 'directly' yerine 'direct' gibi.
3. Sonu özel olan bazı kelimelerin dönüştürülmesi, örneğin 'hopeful' yerine 'hope' ve 'duplicate' yerine 'duplic' gibi.
4. Peş peşe gelen eklerin tespiti ve kaldırılması, örneğin 'interference' yerine 'interfer' gibi.
5. Sonu sesli harfle biten yalın kelimelerin sonundaki sesli harfin kaldırılması, örneğin 'cease' yerine 'ceas' gibi.



Şekil 2.1. Porter kelimenin kökenine inme işlemi [26].

2.1.2. Sayım Vektörleri

Bütüncedeki her bir metni sabit boyutlu bir vektöre dönüştürmek için, her bir kelimenin bütüncede ortaya çıkma sayısı, ya da frekansı, hesaplanır ve o kelimenin vektör içindeki konumuna kaydedilir. Bu yaklaşım aynı zamanda BoW **(**Bag of Words-Kelime Torbası) olarak da bilinir. Vektörün boyutu, bütüncedeki farklı kelimelerin sayısına eşittir, yani her kelimeye has vektör içinde belirli bir konum vardır. Sonuçta, vektörün büyüklüğü bütüncedeki farklı kelimelerin sayısına eşittir, bu da daha küçük vektörler üretmek ve böylece bu vektörleri işlemek için gereken sınıflandırıcıların karmaşıklığını azaltmak için kelimelerin sayısının azaltılmasının önemini gösterir [27, 28].

2.1.3. Kelime Kalıplama

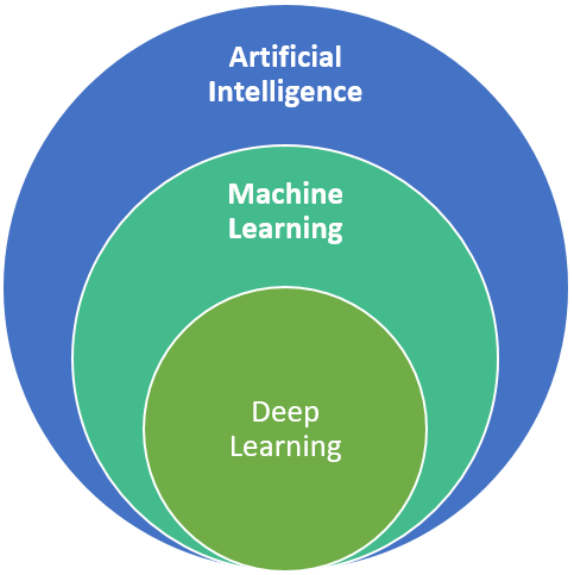
Metni sayısal değerlere dönüştürmenin bir başka yöntemi de kelime kalıplama (*İng.* word embedding) yöntemini kullanmaktır. Bu, bütüncedeki ortaya çıkış sayılarına bağlı olarak, her kelimenin duygusal anlamını değerlendirmek için bir bütünceyi analiz eden denetimsiz bir makine öğrenmesi tekniğidir. Benzer kontekst, yani bağlama sahip görünen kelimeler birbirine benzer kabul edilir, böylece bu kelimeler için benzer değerlere sahip bir vektör oluşturulur. Bununla birlikte, farklı bağlamlarda farklı kelimeler ortaya çıkabileceğinden, çok benzer olsalar da, kelime kalıplama yöntemiyle elde edilen değerler bu kelimelerin duygusal anlamını yansıtır. Word to Vector, ya da kısaca Word2Vec (kelimeden vektöre), Google haberlerinden toplanan devasa bir bütünce kullanılarak eğitilen popüler kelime kalıplama yöntemlerinden biridir. Bu yöntemin çıktısı 300-bileşenli (ya da değerli) bir vektördür, yani her bir kelime 300 boyutlu bir uzayda bir vektörle eşleştirilir [29, 30].

Bu sinir ağı, atlama diyagramı (*İng.* skip diagram) yaklaşımı kullanılarak eğitilir. Bu yaklaşımda, bütüncedeki her bir kelime için One-Hot-Encoded (OHE) denilen bir vektör üretilir. Vektörün boyutu, bütüncedeki farklı kelimelerin sayısına eşittir; burada, söz konusu kelimeye karşılık gelen konum (koordinat) hariç tüm değerler sıfıra eşitlenir. Daha sonra, girdi ve çıktı katmanları arasında ara katman olarak 300 sinirli gizli bir katman çalıştırılır. Sinir ağının çıktısı, bu kelimeyi çevreleyen kelimelerin, yani bir önceki ve bir sonraki kelimelerin olasılığıdır. Bu yaklaşıma göre, normalde benzer kelimelerle çevrelenen kelimelerin gizli katmanda benzer değerlere sahip olması gerekir, aksi takdirde çıktı katmanında farklı kelimeler tahmin edilir [30, 31].

Bir diğer kelime kalıplama yöntemi, Pennington ve diğ. tarafından önerilen Global Vektörlerdir (GLoVe) [32]. Bu yöntem, belirli bir bağlamda birbiri ardınca ortaya çıkan kelimeler arasındaki ilişkiyi hesaplamaya dayanır. Bu ilişkiler iki boyutlu bir dizin üzerine dağıtılır, böylece her bir kelimenin diğer kelimelerle olan ilişkileri bir vektörle temsil edilebilir. Bu yöntem çeşitli metin analizi yöntemlerinde kullanılmakla beraber, bütüncedeki kelimeler için oluşturulan vektörler bu kelimeler arasındaki ilişkileri temsil ederken Word2Vec yönteminde olduğu gibi kelimenin anlamını dikkate almazlar [18, 31].

2.2. Yapay Zeka

Yapay Zeka (Artificial Intelligence-AI) ya da kısaca YZ bilgisayarların, kurallarını tanımlamaya gerek kalmaksızın, bir ortamla etkileşim kurma yeteneği sağlamayı amaçlamaktadır. Yapay zeka, ortam ile etkileşerek bu kuralları tanımlar [33, 34]. Makine öğrenmesi (*İng.* Machine Learning—ML), ortamla etkileşimin, o ortamdan toplanan olaylar veya örneklerin oluşturduğu bir değer kümesi ile tanımlandığı YZ alanlarından biridir. Özellikler veya bu çalışmada tercih edeceğimiz kullanımıyla öznitelikler olarak da bilinen bu değerler, her örneği ya da olayı karakterize eder ve ortamı temsil eden bilgilerin çıkarımı için kullanılır. Bu çıkarım, öznitelik değerlerindeki örüntülerin ve ilişkilerin ve istenen bilginin tanınmasına dayanmaktadır [34, 35]. Son zamanlarda, makine öğrenmesinde Derin Öğrenme'nin (*İng*. Deep Learning—DL) kullanımı, bilgi çıkarımında belirgin şekilde daha iyi sonuçlar vermektedir, burada örüntü ve ilişkileri tespit etmede derin yapay sinir ağları kullanılır. Yapay sinir ağlarında insan beyninden esinlenilerek, üçten fazla katmanın kullanılması, öznitelikler arası karmaşık veya derin ilişki ve bilgi çıkarımı yeteneğine sahiptir. Böylece, bu tip makine öğrenmesi teknikleri ile daha iyi performans elde edilmiştir. Şekil 2.2 yapay zeka, yapay zeka ile öğrenme ve derin öğrenme hiyerarşisini özetlemektedir.



Şekil 2.2. Yapay Zeka (Artificial Intelligence), Makine öğrenmesi (Machine Learning) ve Derin Öğrenme (Deep Learning).

2.3. Makine öğrenmesi

Bilgisayarlara, insanlarla herhangi bir etkileşimde bulunmaksızın dış dünyadan bilgi edinme ve karar verme yeteneği kazandırmak yapay zeka ile öğrenme olarak bilinir. Yapay zeka ile öğrenmede, aynı algoritmalar, sistemin girdilerine bağlı olarak farklı çıktılar verebilirler, burada söz konusu girdiler sistemden daha önce hiç geçmediği halde sistem bu girdileri işleyebilme yeteneğine sahip olabilir. Veri madenciliği, yapay zeka ile öğrenmenin çalışma alanlarından biridir. Yapay zeka ile öğrenme teknikleri, denetimli, denetimsiz ve takviyeli öğrenme olmak üzere üç kategoriye ayrılır [36].

Denetimli öğrenmede, girdilerden bilgi çıkarımı yapabilmek için bu girdilerin etiketlenmesi gerekir. Denetimli öğrenme tekniklerinde girdiler ve bu girdilere verilen etiketler arasındaki ilişkiler araştırılmaktadır. Denetimli veri madenciliğinde en çok kullanılan tekniklerden biri sınıflandırmadır, burada her bir girdiye verilen etiket bu girdinin ait olduğu sınıfı temsil eder. Daha sonra sınıflandırıcılar, bu girdiyi karakterize eden öznitelikler ile girdinin üyesi olarak etiketlendiği sınıf arasındaki ilişkileri çıkarır. Bu bilgi daha sonra yeni girdiler için bir sınıf öngörmek amacıyla sınıflandırılmamış olan yeni girdilere uygulanır. Bu öngörü ya da tahmin, o sınıftaki girdilerin genel özelliklerine bağlı olarak, o yeni girdinin gelecekteki davranışını tahmin etmeye yardımcı olabilir [37].

Sınıflandırıcılar bilgi çıkarımı için etiketlenmiş veri kümesine ihtiyaç duyarlar, böylece bu veri kümesi sınıflandırıcıyı eğitmek için kullanılır. Bu veri kümesi, eğitim veri kümesi olarak bilinir. Ancak, sınıflandırıcılar tahmin için kullanıldığından, sınıflandırıcının performansını etiketlenmemiş veri kümesi kullanarak değerlendirmek mümkün değildir; aynı eğitim veri kümesini kullanmak da performans değerlendirmesi için iyi bir yöntem değildir, çünkü eğitim sırasında sınıflandırıcı bu girdileri kullanmıştır ve bu değerlendirme tahmin performansını ölçmez. Dolayısıyla, daha doğru ölçümler elde etmek için, etiketlenmiş veri kümesi iki gruba ayrılır. İlk grup eğitim aşaması için kullanılırken diğeri sınıflandırıcıyı test etmek için kullanılır. Bu yaklaşımda, değerlendirme için kullanılan veriler eğitime dahil edilmez, ancak bu veri kümesindeki girdilerin gerçek sınıfları bilinir; böylece doğru değerlendirme ve ölçümler için test veri kümesi sınıflandırıcı tarafından işlenir ve sınıflandırıcının tahminleri ile gerçek etiketler karşılaştırılır [38]. Veri kümesinden bilgi çıkarımı

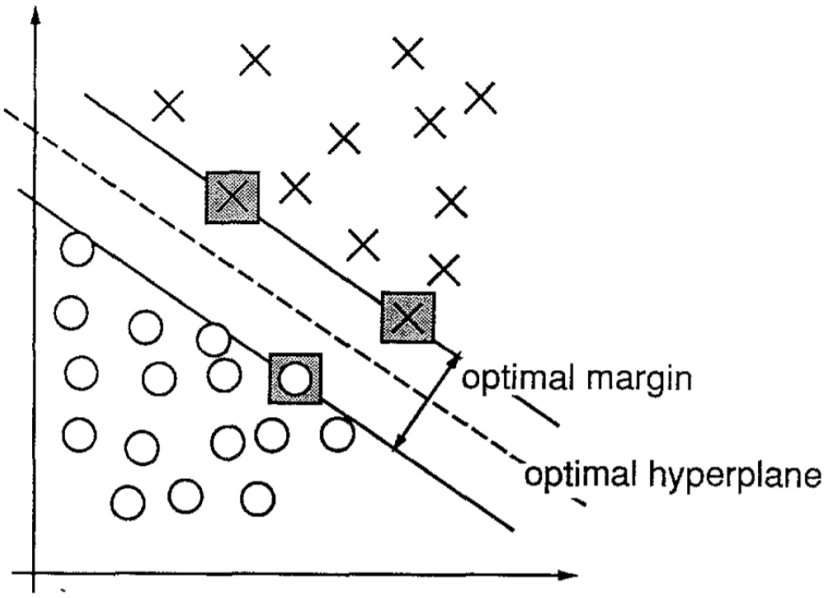
yapmak için kullanılan farklı sınıflandırıcılar vardır. Bu sınıflandırıcılar bilgi çıkarımı amaçlı farklı yaklaşımlara sahiptir. Bununla birlikte, sınıflandırıcıların performansı bir veri kümesinden diğerine değişebilir, ayrıca belirli bir veri kümesinde bir sınıflandırıcı diğerinden daha iyi performans gösterebilirken, diğer sınıflandırıcı başka bir veri kümesinde daha iyi performans gösterebilir. Bu nedenle, bir veri kümesinde en iyi performans verecek sınıflandırıcıyı seçmek için birden fazla sınıflandırıcının performansını test etmek önemlidir. Ayrıca, Destek Vektör Makinesi (Support Vector Machine—SVM), Rastgele Orman (Random Forests—RF) ve Derin Öğrenme (Deep Learning—DL) sınıflandırıcıları gibi diğerlerinden genelde daha iyi bir performans gösteren sınıflandırıcılar vardır.

2.3.1. SVM (Support Vector Machine- Destek Vektör Makinesi) Sınıflandırıcısı

SVM sınıflandırıcısı, eğitildikten sonra sağladığı tahminlerin doğruluğu açısından belirgin bir performans gösteren en eski sınıflandırıcılardan biridir. Bu sınıflandırıcı başlangıçta veri kümesindeki olası sınıf sayısının iki olduğu ikili (binary) sınıflandırma için önerilmiş olsa da, veri kümesinde herhangi bir sayıda sınıfın bulunabileceği çok sınıflı sınıflandırma problemlerini ele almak üzere geliştirilmiştir. Bu sınıflandırıcı çok boyutlu bir sanal uzay oluşturur ve veri örnekleri özniteliklerinin değerlerine bağlı olarak o uzaya dağıtılır. Dolayısıyla D-öznitelikli bir veri kümesi için, veri örneklerini dağıtmak üzere SVM sınıflandırıcısı tarafından D-boyutlu bir sanal uzay oluşturulur [39, 40].

SVM sınıflandırıcısının eğitimi sırasında, etiketlenmiş olan veri örnekleri, her bir öznitelik değeri uzayın karşılık gelen boyutuyla eşlenerek o alana dağıtılır. Bu veri örneğinin etiketine bağlı olarak, SVM sınıflandırıcısı tüm uzayı daha küçük bölgelere bölen hiperdüzlemler oluşturmaya çalışır, öyle ki her bölge tek bir sınıfın veri örneklerine sahip olmalıdır. Bununla birlikte, bu tür bir dağılımı başarmak, gerçek hayattaki veri kümelerinin çoğunda neredeyse imkansızdır; çünkü gürültü olarak da bilinen bazı veri örnekleri, bir sınıfın öznitelik değerlerine sahip olup başka bir sınıfa ait olabilir, veya birbirine çok yakın değerlere sahip olup farklı sınıflara ait veri örnekleri de olabilir. Dolayısıyla sınıflandırıcı, belirli bir sınıfın veri örneklerinin mümkün olduğunca o bölgede çoğunluğu oluşturacağı bölgeler üretmeye çalışır [41].

Uzaydaki dağılımına bağlı olarak, veri örneğini sınıflarına göre farklı alanlara bölmek için birden fazla olası hiperdüzlem vardır. Bununla birlikte, uzayı bölebilecek tüm olası hiperdüzlemlerden birisi optimal hiperdüzlem olarak seçilmelidir. Optimal hiperdüzlemin seçim sürecini anlamak için, SVM sınıflandırıcısının izlediği tahmin yaklaşımını görselleştirmek önemlidir. SVM sınıflandırıcısını kullanarak bir veri örneğine ait etiketi tahmin etmek için, yeni veri örnekleri sınıflandırıcı tarafından oluşturulan ve eğitim aşaması sırasında alt bölgelere bölünen uzayda konumlandırılır. Yeni veri örneğinin eşlendiği bölgeye atanan etikete bağlı olarak, bu etiket tahmin edilen etiket olarak seçilir. SVM sınıflandırıcısının yaptığı tahminin güvenilirliği, uzaydaki eşlenen nokta ve bölgeyi tanımlayan hiperdüzlem arasındaki mesafeye bağlıdır. Daha uzak mesafeler daha güvenli tahminler üretirken, hiperdüzlemin yakınına eşlenene noktaların daha az güveni vardır. Bu nedenle, Şekil 2.3'te gösterildiği gibi, eğitim sırasında her sınıftan en yakın noktalara en uzak olan hiperdüzlem, yani marjinleri en uzak olan hiperdüzlem seçilir [42].



Şekil 2.3. SVM sınıflandırıcıda uzayı bölecek optimal hiperdüzlemin gösterimi [42].

SVM sınıflandırıcısı Jianqiang ve diğ. [43] tarafından Stanford Twitter Duygu Testi veri kümesi kullanılarak duygusal analiz için değerlendirilmiştir. Aynı çalışmada değerlendirilen diğer sınıflandırıcılara göre bu sınıflandırıcının daha düşük performansı olduğu görülmüştür; ancak sayım vektörlerine kıyasla GLoVE kelime kalıplama kullanımının, bu sınıflandırıcının performansını geliştirebildiği de

gösterilmiştir. Bu kıyaslama, girdi metninin daha verimli ve doğru bir şekilde temsil edilmesinin, sınıflandırıcının performansı üzerinde önemli bir etkiye sahip olduğunu göstermektedir. Ancak, sınıflandırma için yapay bir sinir ağının kullanılmasına rağmen, Word2Vec kelime kalıplama yönteminin kullanımı çalışmaya dahil edilmemiştir.

2.3.2. Naïve Bayes (NB) Sınıflandırıcı

Naïve Bayes sınıflandırıcısı Bayes teoremine dayanarak veri kümesinde var olan sınıfların her birinde, her bir öznitelik değerinin olasılığını hesaplar. Daha sonra bu olasılıklar, gelecekteki girdilerin sınıfların her birinde olma olasılığını hesaplayarak kategorisini tahmin etmek için saklanır [44, 45]. *n* öznitelik tarafından karakterize edilen girdinin, *y* kategorisinde olma olasılığının nasıl hesaplanacağı Denklem 2.1'de verilmektedir.

$$P(y|x_1, x_2, \dots\dots, x_n) = \frac{P(y)P(x_1, x_2, \dots\dots, x_n|y)}{P(x_1, x_2, \dots\dots, x_n)} \tag{2.1}$$

Bu durumda Denklem 2.2, *i.* özniteliğin koşullu bağımsızlık varsayımıyla sonuçlandırılabilir, bu da devamında her girdi özniteliği *i* için Denklem 2.3'ü verecektir:

$$P(x_i|y, x_1, \dots., x_{i-1}, x_{i+1}, \dots., x_n) = P(x_i|y) \tag{2.2}$$

$$P(y|x_1, \dots., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, x_2, \dots\dots, x_n)} \tag{2.3}$$

Her kategori için girdinin muhtemel bir kategoride olma olasılığını hesapladığımızda, Denklem 2.4'te verildiği gibi, bu girdinin en yüksek olasılığı veren kategoride olacağı tahmini ortaya çıkacaktır.

$$\hat{y} = argmax\, P(y)\prod_{i=1}^{n} P(x_i|y) \tag{2.4}$$

2.3.3. Karar Ağacı ve RF (Random Forest-Rastgele Orman) Sınıflandırıcı

Bu sınıflandırıcı, bir karar ağacı tarafından oluşturulan, çok sayıda IF/THEN karşılaştırma bloğundan oluşan bir modele dayanır; burada her bloktaki koşullar,

eğitim veri kümesinde tespit edilen örüntülere dayanır. Bu bloklar çoklu seviyelere dağıtılır, belirli bir seviyedeki belirli bir bloğun sonucu bir sonraki seviyede yürütülen bloğu belirler. Karşılaştırmalara ek olarak, bloklar yaprak olarak bilinen kararları da içerebilir. Karar ağacı bir yaprağa ulaştığında, o yapraktaki etiket girdi verisi örneğinin tahmini olarak seçilir ve karşılaştırmalar sonlandırılır. Oluşturulan modeldeki en yüksek seviye, ağacın kökünü temsil eden tek bir karşılaştırma bloğundan oluşur [46].

Veri kümesindeki her öznitelik ile her veri örneğine atanan sınıflar arasındaki bilgi kazancı, karar ağacının kökü için en yüksek bilgi kazancı olan özniteliği seçmek amacıyla hesaplanır. Bu blok oluşturulduğunda, karşılaştırmanın iki olası sonucu vardır. Bu bloktaki karşılaştırmayı kullanıp verileri filtreleyerek, veri kümesinin tamamı iki parçaya ayrılır. Her bölünme/ayrışma (dal) başına her bir özniteliğin bilgi kazancı, o parçadaki veri örneklerine atanan etiketlere göre hesaplanır. Bölünmede kullanılan en yüksek bilgi kazanımına sahip öznitelik, bölünmede kullanılan karşılaştırmanın sonucuna bağlı olarak bir üst seviyedeki bloğa bağlanan bloktaki karşılaştırma için seçilir. Bölmedeki veri örneklerinin etiketlerinin homojenliği, önceden tanımlanmış bir eşik değerinden daha yüksekse, karşılaştırma yerine bir yaprak yani bir karar oluşturulur. Bu işlem, tüm dallar yapraklarda bitinceye kadar herhangi bir seviyede oluşturulan her karşılaştırma bloğu başına tekrarlanır [47]. Denklem 2.5 bir özniteliğin sağladığı bilgi kazanımını hesaplamak için kullanılır.

$$Gain(\vec{y}, j) = Entropy(\vec{y}) - Entropy(j|\vec{y}) \tag{2.5}$$

burada *n* değeri olan *y* özniteliğinin Shannon Entropisi, $Entropy(\vec{y})$ Denklem 2.6 kullanılarak hesaplanırken, sınıflara göre koşullu entropisi $Entropy(\overrightarrow{j|y})$, Denklem 2.7 ile hesaplanır.

$$Entropy(\vec{y}) = -\sum_{j=1}^{n} \frac{|y_j|}{|\vec{y}|} \log_2 \frac{|y_j|}{|\vec{y}|} \tag{2.6}$$

$$Entropy(\overrightarrow{j|y}) = \frac{|y_j|}{|\vec{y}|} \log_2 \frac{|y_j|}{|\vec{y}|} \tag{2.7}$$

Diğer sınıflandırıcılara kıyasla hızlı tahminler veren sadeliğine rağmen, karar ağacı sınıflandırıcısı aşırı uyum probleminden muzdariptir. Sınıflandırıcı tarafından çıkarımı

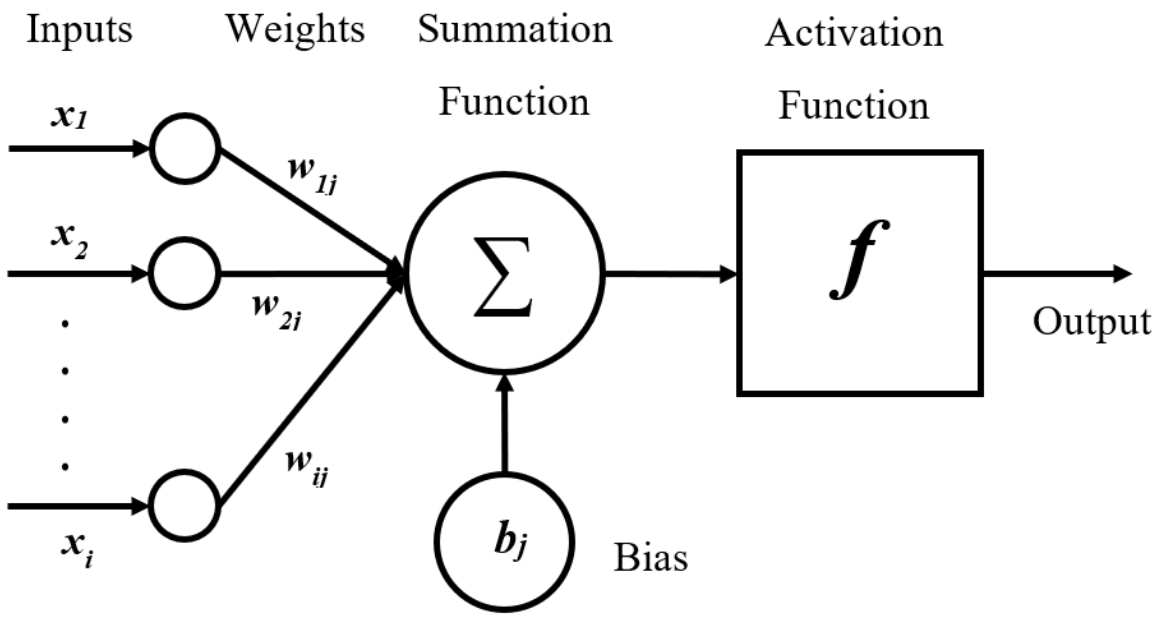
yapılan bilgi, eğitim veri kümesine katı bir şekilde bağlı olduğunda veya sınıflandırıcı tarafından yapılan tahminler çoklu örüntüleri algılamak yerine belirli özniteliklere bağlı olduğunda aşırı uyum ortaya çıkar; bu durum gürültü denilen örneklerin etkisini artırır. Karar ağacının hiyerarşisi, diğer özniteliklerin değerlerine bakılmaksızın, belirli kararların belirli bir öznitellikteki değerden büyük ölçüde etkilendiği aşırı uyumlu bir modelle sonuçlanabilir. Böyle bir sorunun üstesinden gelmek için, ormanın tahmininin tek bir karar ağacı yerine çok sayıda karar ağacına dayandığı rastgele orman sınıflandırıcısı kullanılır [48, 49].

Rasgele orman—RF sınıflandırıcısı, önceden tanımlanmış sayıda karar ağacı içeren bir orman oluşturur. Eğitim veri kümesi daha sonra ormandaki ağaç sayısına eşit sayıda kümeye bölünür. Her küme, eğitim veri kümesindeki tüm sınıfların veri örneklerini içerir ve mümkün olduğunca her kümede her sınıfa ait örnek sayısı, orijinal eğitim veri kümesindeki oranına yakın olur. Ormandaki her karar ağacı, oluşturulan kümelerden biri kullanılarak eğitilir, burada her bir küme tek bir ağaçla eğitilmelidir. Bu oluşturulan kümelerde, yani eğitim kümesinin alt kümelerinde, örnekler birbirinden farklı olduğundan, karar ağaçları altkümelerinde var olan örüntüleri kullanarak gerekli tahminlere ulaşmayı öğrenirler. Rasgele ormandaki ağaçların her biri, bir girdi verisi örneği için tahminde bulunmak için kullanılır. Karar ağaçları tahminlerine ulaşmak için farklı yolları izleyebileceğinden, aynı girdi verileri için ağaçlar farklı tahminlerde bulunabilirler. Bu sebeple, rastgele orman sınıflandırıcısı ormandan bir çıktı tahmini vermek için bu karar ağaçları arasındaki baskın tahmini araştırır. Bu yaklaşım sayesinde, belirli bir ağaçtaki belirli bir özniteliğe bağlılık daha az etkili olur, çünkü diğer öznitelikler ormandaki diğer ağaçlar tarafından araştırılmaktadır ve bu da aşırı uyumun etkisini azaltarak daha doğru tahminler üretir. Bu nedenle, tek bir karar ağacı yerine rastgele orman sınıflandırıcısı farklı uygulamalarda yaygın olarak kullanılmaktadır [50].

2.3.4. Yapay Sinir Ağları - Ysa (Artıfıcıal Neural Networks - Ann)

İnsanların beyninden esinlenilen YSA'larda hesaplamalar, ağ üzerinde katmanlar halinde dağılmış yapay sinirler olarak bilinen birimlerde yapılır. Belirli bir sinirin girdileri dışardan gelebilir veya bir önceki katmanın sinirlerinin çıktıları olabilir. Bir sinirin çıktısını hesaplamak için, sağlanan tüm girdilerin, her bir girdi için atanmış olan

belirli bir değerle çarpılarak ağırlıklı toplamı alınır ve daha sonra Şekil 2.4'te gösterildiği gibi aktivasyon fonksiyonu olarak bilinen doğrusal olmayan bir fonksiyonun parametresi olarak işleme tabi tutulur. Söz konusu fonksiyonun doğrusal olmama durumu, daha karmaşık özellikleri algılama özelliğine sahip daha esnek bir çıktı sağlar. Bununla birlikte, gerektiğinde sinirin girdilerine bias olarak bilinen ek değerler eklenebilir, böylece hesaplamalarda belirli bir sapma söz konusu olacaktır [51, 52].



Şekil 2.4. Yapay bir sinirin içindeki hesaplamaların gösterimi [51].

YSA'nın türünden bağımsız olarak, bu ağlarda iki türlü hesaplama bulunmaktadır, girdiden çıktı yönüne doğru yürütülen hesaplama (ileri pas) ve çıktıdan girdi yönüne doğru yürütülen hesaplama (geri pas) [53]. İleri pas, ağın girdilerini esas alarak çıktılarını hesaplamak için kullanılır, her bir katmanın çıktısı bir sonrakinde yürütülen hesaplamalarda kullanılmak üzere hesaplanır. Geri pasta ise, ağırlıkların değerleri gradyan azalması (gradient descent) ile güncellenir. İleri pasla YSA'nın çıktıları ve veri kümesindeki amaçlanan çıktı değerleri arasındaki sapma ölçülerek, çıktıların ağırlıklarının türevleri hesaplanır. Gradyan azalması, konum ağırlıklarının hatayı azaltmak için ne şekilde güncellenmesi gerektiğini tanımlamak için kullanılır, ağırlık değişimi o konumdaki gradyan azalmasının eksilisi şeklinde olmalıdır. Böyle bir güncelleme, sinir ağının girdi değerlerinden istenen çıktıyı üretmesini ve böylece

gerekli görevi gerçekleştirmesini sağlar. Bu işlemi birkaç iterasyonla tekrarlayarak, ileri pastan gelen çıktı ile amaçlanan çıktı arasındaki kayıp, minimum kayba ulaşılana kadar sinir geri yayılım kullanılarak azaltılır, bu da sinir ağının performansını artıracaktır [54, 55].

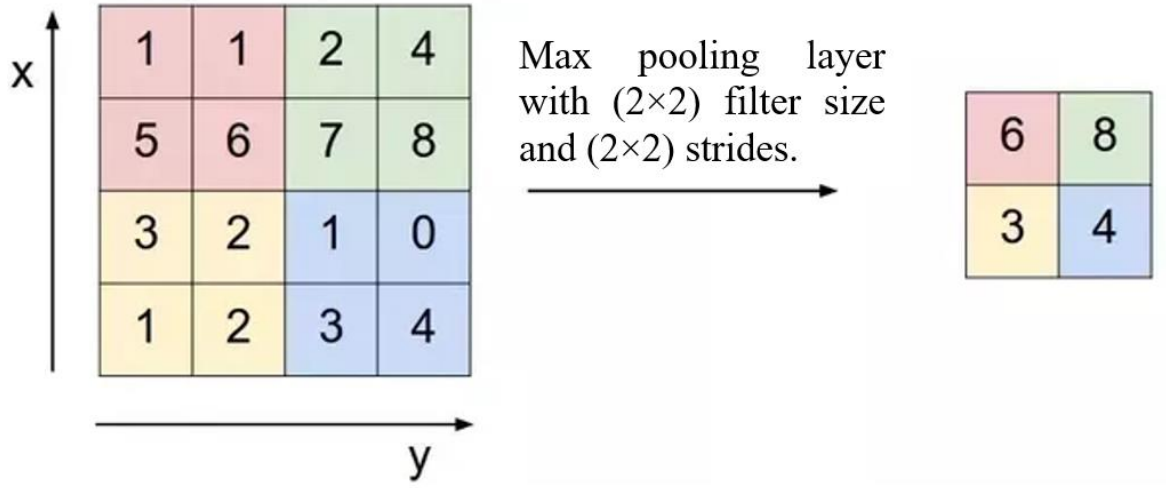
2.3.4.1. CNN (Convolutional Neural Networks-Evrişimli Sinir Ağları)

CNN'ler evrişimli katmanlar içerirler; CNN'ler her bir sinirin girdisi boyunca evrişim sağlayan iki boyutlu filtrelerden oluşan evrişimli katmanlar içerir. Matematiksel olarak, filtre aslında sinirin ağırlık katsayılarıdır, sinirin girdideki yerel iki boyutlu örüntüleri tespit etmesini sağlarlar. Bir evrişimli katmandaki filtrelerin boyutları sabittir ve girdideki örüntüler filtrenin boyutları dahilinde tespit edilebilir. Bununla birlikte, sinir ağının derinliklerine, yani girdi katmanından daha uzak katmanlara gidildiğinde, her filtre bir önceki katmanın filtrelerince algılanan örüntüler tarafından tanımlanan örüntüleri algılar. Bu, CNN'nin tanınan örüntüleri birleştirmesini ve daha karmaşık öznitelikleri algılamasını sağlar. Bir evrişimsel katmanda bir sinirin çıktısı girdisinden farklı boyutlara sahip olabilse de, boyutların sayısı girdideki ile benzerdir, yani iki boyutlu bir girdiyi işleyen bir sinir iki boyutlu bir dizin üretecektir [56, 57].

Evrişim sırasında filtrenin her adımda hareket ettiği değerlerin sayısı adım uzunluğu olarak isimlendirilir, adım uzunluğu yatay ve dikey hareketler için farklı değerlere sahip olabilir. Filtre içindeki tüm değerler karşılık gelen ağırlıklarıyla çarpılır ve sinir içinde işlenirler, sinir vereceği çıktıları filtrelerinin evrişimleri sırasında alınan düzenlemeye uygun olarak ayarlar. Her evrişim sırasında birden fazla değerin atlanması, her ne kadar sinirin çıktılarının boyutunu azaltsa ve bu da sonraki katmanlarda yer alacak hesaplamaları kolaylaştırsa da, önemli örüntüleri algılayamamayı netice verebilir ve bu da CNN'in performansını olumsuz etkileyebilir. Önemli bilgileri kaybetmeden bir sinirden elde edilen çıktının boyutunu azaltmak için, evrişimli katmanlardan sonra birleştirme katmanları (pooling layers) yerleştirilebilir [58].

Sinirin çıktısı birleştirme katmanının girdisidir, bu katmanda da girdisine etki eden evrişimli filtreler bulunur. Ancak bu filtreler girdileri işlemek için farklı bir yaklaşım izlerler, çünkü bu çıktıların birer ağırlığı yoktur ve sinirlere iletilmezler. Her ne kadar farklı türlerde birleştirme katmanları bulunsa da, işlenen verinin boyutlarını önemli

bilgileri kaybetmeksizin küçültmek amacıyla en yaygın şekilde kullanılan birleştirme katmanı Max-Pooling'dir. Şekil 2.5'te gösterildiği gibi, max-pooling'deki bir filtre verili boyutlar içindeki maksimum değeri arar ve o bölgeyi temsil etmek için aynı değeri çıktı olarak verir. En yüksek değeri seçerek, o bölgedeki en önemli öznitelik seçilmiş olur, böylece evrişimli katmanlarda adım uzunluğunun büyümesi sonucu oluşak bilgi kaybı daha az olası olmuş olur [58].



Şekil 2.5. Max-Pooling filtresinin çıktısı.

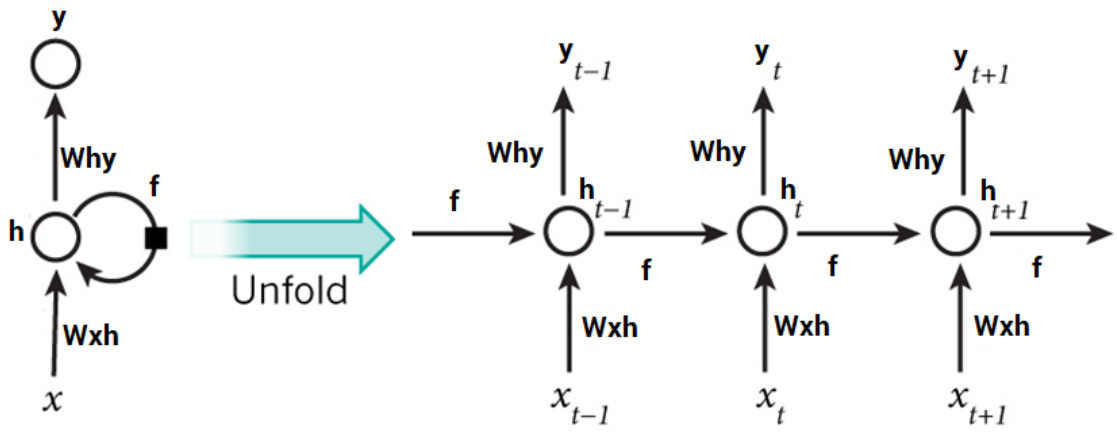
Bir girdinin değerinin yanısıra konumunu dikkate alma yeteneği sebebiyle, CNN'ler NLP'de yaygın olarak kullanılmaktadır. Böyle bir ağ, örneğin cümle içinde "*burada değil*" ifadesinin "*yok*" kelimesine eşdeğer olduğunu anlayabilir, böylece sinir ağının çıktısı açısından bu iki sinirin etkisi benzer olabilir. Ayrıca çoğu uygulamada olduğu gibi, sinir ağından istenen çıktı iki boyutlu değilse, son evrişimsel katmanın çıktısı düzleştirip tek boyutlu hale getirilebilir ve başka bir tek boyutlu katmana tam olarak bağlanabilir. Girdideki özniteliklerin karmaşıklığına bağlı olarak, sinir ağına çıktı katmanından önce daha fazla katman eklenebilir [59, 60].

Jianqiang ve diğ. [43], Cicero ve Maria [61] tarafından yapılan değerlendirmeler, duygusal metin kategorizasyonunda CNN'in üstünlüğünü göstermektedir. Bu daha iyi performans, CNN'nin her bir kelimenin temsiline ek olarak konumunu da dikkate alma yeteneğinin bir sonucudur. Aynı şekilde, bu sinir ağının kullanımı Saif ve diğ. tarafından uygulanan Duygusal Sözlük yönteminden daha iyi bir performans sergilemiştir [62]. Bu karşılaştırmalar, yapay sinir ağlarının, özellikle de daha büyük

eğitim verileri olduğunda, diğer yapay zeka ile öğrenme yöntemlerine olan üstünlüğünü ortaya koyarken, kelimelerin konumlandırılmasını dikkate alan diğer sinir ağlarının kullanımının umut verici sonuçlar verebileceğini de göstermektedir.

2.3.4.2. RNN (Recurrent Neural Networks-Tekrarlayan Sinir Ağları)

Tekrarlayan sinir ağları ya da kısaca RNN'ler, CNN'lere benzer şekilde iki boyutlu girdileri işleyebilir ve her bir girdi kümesi için tek bir değer çıktısı verebilir. Bununla birlikte, RNN'lerin bu girdileri işlemek için kullandığı yaklaşım farklıdır; burada önceki girdi demetinden gelen çıktı ağırlıklandırılır ve önceki katmandan gelen girdilere eklenir veya dışarı aktarılır. Şekil 2.6'da gösterildiği gibi, *t*'de konumlandırılmış olan şu anki demetten önceki demetin çıktı değerini ayarlamak için bir ağırlık değeri *f*'nin kullanıldığını varsayalım. *t*'deki sinirin çıktısının hesaplamaları sırasında, *t-1*'in *h* çıktısı, *f* kullanılarak ağırlıklandırıldıktan sonra dahil edilir. Bu *t* demetinin çıktısı da *f* ile ağırlıklandırılır ve bir sonraki *t+1* demetindeki *x* girdilerine dahil edilir. Bu işlem, girdi kümesindeki tüm demetler işlenene kadar tekrarlanır [63, 64].

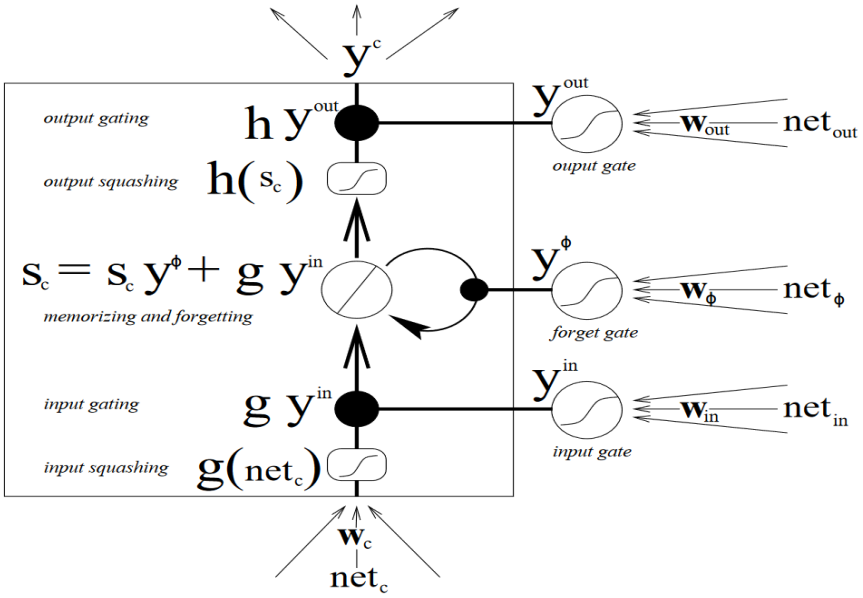


Şekil 2.6. Bir RNN sinirindeki hesaplamalar.

RNN'lerin mevcut demet hesaplarında önceki demetlerde hesaplanan çıktıları dahil edebilme yeteneği dolayısıyla bu tip sinir ağları zaman serilerinde ve NLP uygulamalarında yaygın olarak kullanılmaktadır. Bir cümle, o cümledeki her kelimenin etkisine ve konumuna göre analiz edilebilir. Örneğin, '*değil*' gibi negatif bir kelime işlendiğinde elde edilen çıktı, bir önceki kelimenin girdileri ile birleştirilebilir,

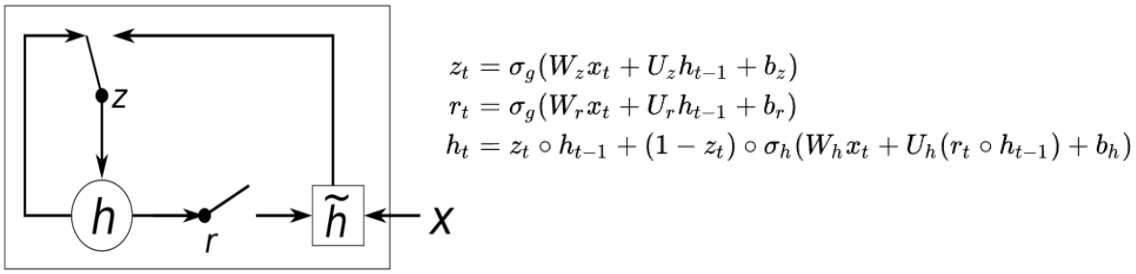
böylece önce yer alan kelimenin anlamı tersine çevrilebilir [65, 66]. Bununla birlikte, sinirden alınan belirli bir çıktının etkisi, ağa girilen demetin pozisyonuna göredir, yani bu durumda, o anda işlenmekte olan demete göredir. *t* anında oluşacak çıktı üzerinde, *t-1*'den gelen çıktının *t-2*'den gelen çıktıya göre daha fazla etkisi vardır. Bununla birlikte, NLP dahil olmak üzere birçok uygulamada, bu tür davranışlar belirli koşullarda önemli etki gösterebilir, ve diğerlerinde ise olumsuz etki gösterebilir. Bu nedenle bu uygulamalarda daha karmaşık bir RNN türü kullanılmaktadır, bu türde belirli bir çıktının etkisi, serideki konumundan ziyade mevcut hesaplamalardaki önemine göre ayarlanır [67].

Bu amaçla, Long-Short Term Memory ya da kısaca LSTM (Uzun-Kısa Süreli Bellek) ağları, girdi ve çıktı arasında değerlerin akışını kontrol edebilmek için geçitler (gate) kullanır. Her bir geçit belirli bir konumdan gelen girdileri kabul eden ayrı bir ağ kullanarak kontrol edilir. Şekil 2.7'de gösterildiği gibi, net_c dışardan gelen değerleri alan ve ağırlıklarına bağlı olarak çıktılarını hesaplayan girdi ağıdır. Bir başka ağ net_{in}, girdi geçidi değeri y^{in} aracılığıyla net_c'den çıktı akışını tanımlayan geçidi kontrol etmek için bu girdilerin bir kopyasını alır. net_ϕ tarafından kontrol edilen unutma geçidi değeri y^ϕ kullanılarak önceki çıktının etkisi ayarlanır. Bu S_c çıktısı bir aktivasyon fonksiyonu tarafından baskılanır, daha sonra çıktı geçidinden alınan y^{out} değerleri ile ayarlanır, çıktı geçidi ise bir önceki zaman noktasında elde edilen çıktıları kullanarak geçidin değerlerini hesaplayan net_{out} ile kontrol edilir. Her bir geçit farklı bir sinir ağı tarafından kontrol edildiği için, her bir sinir ağının ağırlıkları eğitim sırasında güncellenir, böylece o anki zaman noktasında var olan girdi değerlerine ve önceki zaman noktalarından toplanan çıktı değerlerine göre uygun olan karar alınır [68].



Şekil 2.7. Bir LSTM sinir ağında veri akışının gösterimi [68].

Bir başka tekrarlayan sinir ağı çeşidi Gated Recurrent Unit (Geçitli Tekrarlanan Birim) ya da kısaca GRU olarak bilinir, Cho ve diğ. [69] tarafından önerilen bu sinir ağında bir dizi önceki değer konumlarına bakmaksızın seçilebilir. Dolayısıyla bu sinir ağının, mevcut girdinin konumlarına bağlı olarak değil, değerlerine bağlı olarak geçmiş değerlerin etkisini ayarlama yeteneği bulunmaktadır. Şekil 2.8'de gösterildiği gibi, söz konusu sinir ağı bu amaçla iki kontrol geçidi kullanır, bunlar güncelleme ve sıfırlama (update-reset) geçitleridir. Geçmiş verilerdeki herhangi bir alakasız bilgi, sinir ağındaki sıfırlama geçidi kapatılarak elenir, yani yürütülen hesaplamalarda göz ardı edilmiş olur. Güncelleme geçidi, gerektiğinde, sinir ağında yürütülen hesaplamalarda yer alan verilerin boyutunu kontrol eder. Bu tür sinir ağları, zamana duyarlı bilgileri analiz etmede daha iyi performans sergilemiştir; zamana duyarlı olmaktan kasıt, her bir değerin göründüğü zamanın tahminler üzerinde etkili olduğu verilerdir; LSTM'ye kıyasla daha basit hesaplamalar yapar.



Şekil 2.8. Bir GRU'nun yapısı.

2.4. Derin Q-Öğrenme (Deep Q-Learnıng)

Bir ortamı tanımlayan fonksiyona yaklaşımda bulunmak ve belirli bir durum için her bir eyleme ait Q değerlerini tahmin etmek ve böylece aracının en uygun eylemi seçebilmesi amacıyla yapay sinir ağının kullanılması Q-Öğrenme (Q-Learning) olarak bilinir. Bu öğrenme yaklaşımının amacı, sinir ağına ortamdan toplanan gerçek ödülleri sağlamaktır, böylece sonraki hareketlerde veya operasyonlarda sinir ağı bu ödülleri tahmin edebilir [11]. Ancak, sinir ağı aracının etkileşime girdiği ortam hakkında öncesinde bilgi sahibi olmadığından, eğitim, sürecin başında rastgele eylemler yürütmeye dayanır [70]. Sinir ağı çevre hakkında daha fazla bilgi edinmeye başladığında, aracının kararları daha az rastgele olmaya ve sinir ağının tahminlerine daha bağımlı olmaya başlayabilir. Böylesi bir davranışı kontrol altında tutmak için, aracı tarafından alınan kararlardaki rasgeleliğin kontrol edilmesi amacıyla bir değer tanımlanır. Bu değer epsilon ile gösterilir ve normalde yüksek bir değerle başlar, yani başlangıçta daha rastgele eylemler vardır, sinir ağı ortam hakkında daha fazla bilgi edindikçe bu değer de azalır [71].

Rastgele veya sinir ağının çıktılarına dayalı bir eylemin yürütülmesi arasında seçim yapmak için, epsilon değeri rastgele oluşturulmuş bir değerle karşılaştırılır. Rastgele değer epsilondan küçükse, aracı tarafından seçilen eylem en yüksek ödülü veren eylemdir, yani sinir ağının tahminlerine dayanarak eylem yürütülür. Aksi takdirde, eylem rastgele seçilir ve ortama karşı yürütülür [72]. Her iki durumda da, seçilen eylemin mevcut durumda yürütülmesi üzerine ortamdan toplanan ödül, aracının girdiği yeni durum için sinir ağı tarafından öngörülen maksimum Q değeri ile birlikte sinir ağını eğitmek için kullanılır [73, 74].

Aracı bir bölümü veya bir aşamayı bitirdiğinde, sinir ağı, o bölüm veya aşama sırasında aracı tarafından toplanan verilerle, yani farklı durumlar, eylemler ve ödüller kullanılarak eğitilir ve epsilon değeri, gama değeri olarak bilinen önceden tanımlanmış bir oranla azaltılır. Bu süreç, sinir ağının, aracının karşı karşıya olduğu her bir durum için optimal eylemi seçmesine yardımcı olabilecek doğru Q değeri üretmek için yeterli bilgi kazanması beklenen tanımlanmış eğitim bölümü/aşaması sayısına ulaşılana kadar tekrarlanır [10, 75]. Sinir ağlarının eğitim sırasında hiç karşı karşıya kalmadığı durumlar için yaklaşımlar sağlama yeteneği, bu ağların Derin Q-Öğrenme (DQN) yaklaşımında kullanılmasına imkan verir, böylece uygun kararları verebilmek için aracının hala yaklaşık Q değerleri vardır. Bu yaklaşımı, durumlar ve karşılık gelen Q değerlerini içeren tabloların kullanımı ile karşılaştırmak, yaklaşım hesaplamalarının faydalarını gösterir, çünkü Q tablosuna dahil edilen durumlara karşılık gelen Q değerleri, aracı tarafından tanınabilir [76, 77]. Bu nedenle, karmaşık ortamların fonksiyonlarına yaklaşımda bulunmak için DQN yaygın olarak kullanılmaktadır.

2.5. Performans Değerlendirmesi

Bir sınıflandırıcının performansı, genellikle iki kriter açısından değerlendirilir; bu kriterler tahminlerin doğruluğu ve F1-Puanı'dır. Tahminlerin doğruluğu, değerlendirmede baz alınan veri kümesindeki, yani test veri kümesindeki doğru şekilde sınıflandırılan örnek sayısının tüm örnek sayısına oranıdır [78]. Ayrıca, bir sınıflandırıcının F1-Puanı'nı hesaplamak için, o sınıflandırıcının yaptığı tahminlerdeki kesinlik (*İng*. precision) ve hassasiyet (*İng*. recall) değerlerine bakılması önemlidir. Kesinlik ve hassasiyet her bir sınıf için ayrı ayrı hesaplanır; her ikisi de test veri kümesi bağlamında olmak üzere, kesinlik doğru şekilde o sınıfa ait olduğu tahmin edilen veri sayısının o sınıfa ait olduğu tahmin edilen toplam veri sayısına oranına eşittir; hassasiyet ise, o sınıfa ait olduğu doğru şekilde tahmin edilen eleman sayısının o sınıfa ait gerçekteki eleman sayısına oranıdır. Denklem 2.8 her bir sınıf için kesinlik ve hassasiyet değerlerini kullanarak F1-Puanı formülünü, Denklem 2.9 ise sınıflandırıcı için genel F1-Puanı formülünü vermektedir, burada F1-Puanlarının ağırlıklı ortalaması alınır, c veri kümesindeki toplam sınıf sayısını, T ise verideki eleman sayısını temsil etmektedir.

$$F1\_Puanı_c = 2 \times \frac{kesinlik \times hassasiyet}{kesinlik + hassasiyet} \quad (2.8)$$

$$F1\_Puanı = \sum_{c=1}^{C} \frac{F1\_Puanı_c * T_c}{T} \quad (2.9)$$

3. METODOLOJİ

Öznitelik belirleme için önerilen yöntem bu bölümde detaylı şekilde tanımlanacaktır. Bu tanımlamada, önerilen yöntemin metodolojisi ve uygulamasının yanı sıra, temel özellikleri ve hedeflenen amaçlara yönelik olarak her tekniğin nasıl kullanıldığı yer alacaktır. Ayrıca, sinir ağını önerilen yöntemle eğitmek için takip edilen eğitim süreci ve performans değerlendirmede izlenen yaklaşım da bu bölümde izah edilecektir.

3.1. Araştırma Soruları

S1: Tahminlerin kalitesi açısından metinleri kategorize etmek için hangi teknik daha uygundur?

- SVM sınıflandırıcısı tarafından sağlanan tahminlerin doğruluğu ve F1-puanı nedir?
- NB sınıflandırıcısı tarafından sağlanan tahminlerin doğruluğu ve F1-puanı nedir?
- RF sınıflandırıcısı tarafından sağlanan tahminlerin doğruluğu ve F1-puanı nedir?
- CNN sınıflandırıcısı tarafından sağlanan tahminlerin doğruluğu ve F1-puanı nedir?
- LSTM sınıflandırıcısı tarafından sağlanan tahminlerin doğruluğu ve F1-puanı nedir?
- GRU sınıflandırıcısı tarafından sağlanan tahminlerin doğruluğu ve F1-puanı nedir?

S2: Hangi teknik bu tahminleri daha hızlı bir şekilde verebilmektedir?

- Her bir tahmin için SVM sınıflandırıcısının ihtiyaç duyduğu ortalama işlem süresi nedir?
- Her bir tahmin için NB sınıflandırıcısının ihtiyaç duyduğu ortalama işlem süresi nedir?
- Her bir tahmin için RF sınıflandırıcısının ihtiyaç duyduğu ortalama işlem süresi nedir?

- Her bir tahmin için CNN sınıflandırıcısının ihtiyaç duyduğu ortalama işlem süresi nedir?
- Her bir tahmin için LSTM sınıflandırıcısının ihtiyaç duyduğu ortalama işlem süresi nedir?
- Her bir tahmin için GRU sınıflandırıcısının ihtiyaç duyduğu ortalama işlem süresi nedir?

S3: Önerilen yöntem, sinir ağının eğitimine dahil olmayan kelimeler için doğru sıralamalar vermekte midir?

- Önerilen öznitelik belirleme yönteminin verdiği tahminler, bu özniteliklerin sınıflandırıcıların performansı üzerindeki gerçek etkisine benzemekte midir?

S4: Önerilen yöntemin verdiği tahminler her tür sınıflandırıcı için uygun mudur?

- Önerilen öznitelik belirleme yöntemi, bütüncede yer alan metinleri temsil etmek için sayım vektörleri kullanan SVM, NB ve RF sınıflandırıcıları için ne gibi bir iyileştirme sağlamaktadır?
- Önerilen öznitelik belirleme yöntemi, bütüncede yer alan metinleri temsil etmek için kelime kalıplama kullanan CNN, LSTM ve GRU yöntemleri için ne gibi bir iyileştirme sağlamaktadır?

S5: Önerilen öznitelik belirleme yöntemi, sınıflandırıcıların mevcut olan en son duygu analizi teknikleri yöntemlerinden daha iyi bir performans sergilemesine imkan vermekte midir?

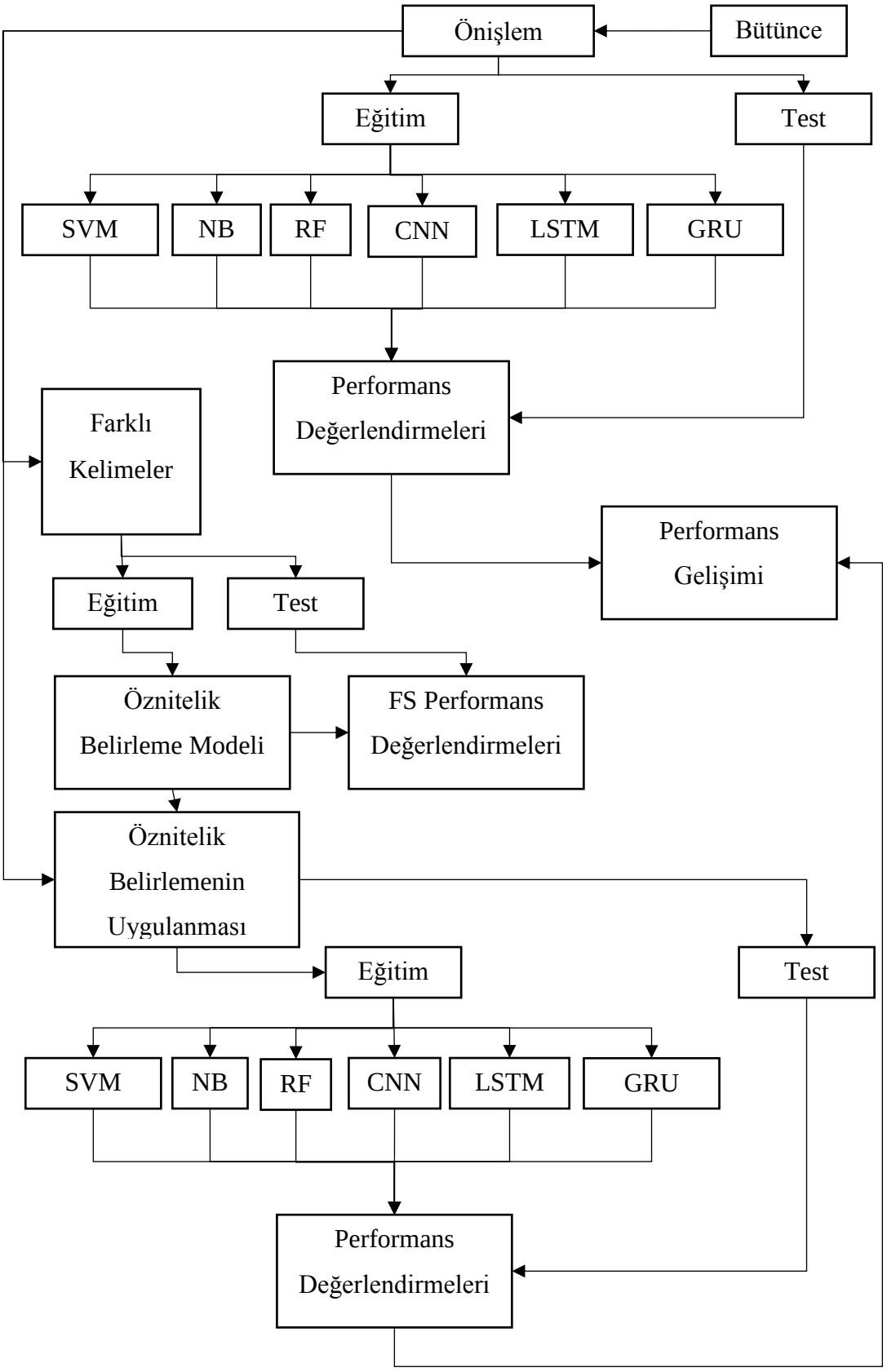
3.2. Metodolojiye Genel Bakış

Önerilen öznitelik belirleme yöntemi, karmaşık hesaplamalar yaparak bütüncenin tamamını birden analiz etmek yerine, bir kelimenin bütünce içindeki önemini değerlendirmek için kelime kalıplama yöntemi tarafından elde edilen vektörlere dayanmaktadır. Bilinen kelimenin önemine bağlı olarak, bu kelimeye benzeyen bilinmeyen bir kelime doğrudan değerlendirilebilir. Bu bağlamda, önerilen yöntem bilinen kelimeler kullanılarak eğitilmelidir, bu eğitimde söz konusu kelimelerin sınıflandırıcının performansı üzerindeki etkisi değerlendirmeye tabi tutulmalıdır. Kelimeler üç kategoriye ayrılır:

1. Pozitif: sınıflandırıcının performansı üzerinde pozitif etkisi olan kelimeleri içerir, yani bu kategorideki kelimelerin yokluğu sınıflandırıcı tarafından yapılan tahminlerin kalitesini düşürecektir.
2. Nötr: sınıflandırıcının performansı üzerinde hiçbir etkisi olmayan kelimeleri içerir, yani bu kategorideki kelimelerin yokluğu sınıflandırıcı tarafından yapılan tahminlerin kalitesini değiştirmeyecektir. Bununla birlikte, bu kategorideki kelimeler sınıflandırıcının performansını yine de etkileyebilir, çünkü bu kelimelerin dahil edilmesi daha karmaşık modeller gerektirecek, bu da sonuç almak için gereken süreyi uzatacaktır.
3. Negatif: sınıflandırıcının performansı üzerinde negatif etkisi olan kelimeleri içerir, yani bu kategorideki kelimelerin yokluğu sınıflandırıcı tarafından yapılan tahminlerin kalitesini arttıracaktır. Sınıflandırıcının performansını artırmak için bu kelimelerin elenmesi, hem tahminlerin kalitesi hem de gerekli modelin karmaşıklığı açısından zorunludur.

3.3. Çalışmanın Adımları

Şekil 3.1'de gösterildiği gibi, çalışmanın ilk adımı bütünce içindeki metinleri ön işleme tabi tutmaktır, böylece sınıflandırıcılara daha etkin ve kullanışlı veri sağlanacaktır. Daha sonra her sınıflandırıcının performansı, ön işleme tabi tutulmuş veriler kullanılarak değerlendirilir. Önerilen öznitelik belirleme yöntemini eğitmek ve sonrasında verdiği tahminlerin kalitesini değerlendirmek için, bütünce içindeki kelimeler eğitim ve test kümelerine ayrılır. Daha sonra, sadece üst sıralardaki öznitelikleri, yani kelimeleri seçmek için önerilen yöntemden elde edilen sıralamalar kullanılır. Daha sonra, önerilen yöntemin sağladığı performans gelişmesini değerlendirmek için, aynı sınıflandırıcılar seçilen kelimeleri kullanarak aynı veri kümesini sınıflandırırlar.



Şekil 3.1. Çalışmanın Temel Adımları.

3.4. Metin Ön İşleme

Veri kümesinin çok boyutluluğunu azaltmak ve cümleler içinde duygusal etkisi olmayan kelimeleri elemek için, veri kümesi bir dizi önişlem adımına tabi tutulur. Bu adımlar:

- **Sayıların elenmesi:** Metinde yer alan her bir sayı, "number" (="sayı") kelimesiyle değiştirilir çünkü o sayının gerçek değeri aynı kelimeye herhangi bir duygusal anlam katkısında bulunmamaktadır. Bu sayede metinde temsil edilen bilgiyi etkilemeksizin vektörlerin boyutu önemli ölçüde azaltılacaktır.
- **Noktalama işaretlerinin elenmesi:** [!"#$%&'()*+,-./:;<=>?@[\]^_`{}~]:. gibi noktalama işaretleri, metne kayda değer bir bilgi katkısında bulunmadıkları gibi, girdinin çok boyutluluğunun artmasına sebebiyet verebilirler. Bu noktalama işaretleri, sayısal biçime dönüştürmeden önce metinden elenirler.
- **Etkisiz kelimelerin (stop word) elenmesi:** İnglizce'de 'the', 'a', 'on', 'are' ve 'all' gibi kelimeler "stop word=etkisiz kelimeler" olarak tanımlanırlar. Bu kelimeler bulundukları cümleye herhangi bir duygusal anlam kazandırmamaktadır. Dolayısıyla, bütünceden elde edilecek modelin karmaşıklığını azaltmak için bu tip kelimeler elenmelidir.
- **Kelime kökenine inme (stemming):** İnglizce'deki hemen hemen bütün kelimeler cümle içinde bulundukları konuma göre, kullanılan zamana göre ve daha başka sebeplerle farklı formlar kazabilirler. Aynı kelimenin farklı formlarının farklı kelimeler olarak kabul edilmesi sınıflandırıcıya yönlendirilen girdinin çok boyutluluğunu arttıracaktır. Dolayısıyla metinde yer alan kelimelerin anlamını bozmadan kökenine inilmesi, metinde yer alan farklı kelime sayısını azaltacaktır.

3.5. Öznitelik Belirleme

Önerilen öznitelik belirleme yöntemi Takviyeli Öğrenme'ye dayalıdır, takviyeli öğrenmede her bir kelimenin elenmesinden sonra sınıflandırıcının performansı ve böylece aynı kelimenin önemi değerlendirilir. Tablo 3.1'de açıklanan sinir ağı uygulanarak her bir girdinin tanımlanan üç kategoriden birinde olma olasılığı hesaplanır, bunun için de Word2Vec kelime kalıplama yöntemi kullanılır. Çıkış

katmanındaki sinirde kullanılan aktivasyon fonksiyonu, [-1, 1] aralığında sürekli değerler veren hiperbolik tanjanttır. Böylesi bir çıktı, hangi değerlerin dahil edileceğini ve hangilerinin hariç tutulacağını belirlemek için kontrol edilebilen daha esnek değerlere olanak vermektedir. Bu kontrol, dahil edilen öznitelikleri elenenlerden ayıran bir eşik değeri belirlemek suretiyle kolayca sağlanabilir. Örneğin, eşik değeri -0.5 olarak alınırsa, nötr kategorideki kelimeler de belirlenen özniteliklere dahil edilmiş olur; 0.5 olarak alındığında ise söz konusu kelimeler belirlenen özniteliklerin dışında bırakılmış olur. Dolayısıyla, eşik değerinin belirlenmesi, sınıflandırma aşamasında dahil edilecek kelimelerin minimum önemini tanımlamış olacaktır. Ayrıca, girdi katmanındaki nöron sayısı 300'dür, bu sayı Word2Vec sinir ağı tarafından elde edilen vektörlerin büyüklüğüne göre belirlenmiştir.

Tablo 3.1. Önerilen öznitelik belirleme yöntemi için kullanılan sinir ağının tanımı.

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	154112
dense_2 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8256
dropout_3 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 32)	2080
dropout_4 (Dropout)	(None, 32)	0
dense_6 (Dense)	(None, 16)	528
dropout_5 (Dropout)	(None, 16)	0
dense_7 (Dense)	(None, 8)	136
dense_8 (Dense)	(None, 1)	9

Total params: 329,345
Trainable params: 329,345
Non-trainable params: 0

3.6. Önerilen Sinir Ağlarının Eğitilmesi

Bölüm 2.2.1'de gösterildiği gibi, sınıflandırmada dahil edilen özniteliklere en hassas olan sınıflandırıcı SVM'dir, çünkü tahmin hesabı aşamasında bütün öznitelikler eşit olarak kabul edilmektedir. Dolayısıyla, önerilen sinir ağının eğitiminde RL

yaklaşımıyla bu sınıflandırıcı kullanılmaktadır. Birincisi, sınıflandırıcının doğruluğu bütüncedeki her kelimeyi dahil eden sayım vektörleri kullanılarak ölçümlenir. Bu doğruluk, her bir kelimenin sınıflandırıcının performansı üzerindeki etkisi ve kontrol değeri olarak loglanır. Daha sonra her bir kelime bütünceden çıkarılır, yeni sayım vektörleri oluşturulur ve sınfılandırıcının doğruluğu ölçülür. Eğer doğrulukta azalma varsa, çıkarılan kelime önemlidir ve önerilen sinir ağı bu tepkiye göre eğitilir. Word2Vec sinir ağı tarafından oluşturulan vektör, önerilen sinir ağı için girdi olarak kullanılırken, pozitif etkinin bir göstergesi olarak çıktıya ise 1 değeri verilir. Benzer şekilde, eğer doğruluk değişmezse, 0 olarak etiketlenir ve sinir ağı eğitilir. Son olarak, eğer doğrulukta artış varsa, -1 olarak etiketlenir, çünkü bu kelimenin sınıflandırıcı performansı üzerinde negatif etkisi vardır.

Sinir ağının bu yaklaşımla eğitilmesi nedeniyle, eğitim aşamasında yer almayan herhangi bir kelimenin basit bir şekilde sinir ağından geçirilerek değerlendirilmesi mümkündür, bu işlem sonucunda elde edilen değer o kelimenin önemini, yani sıralamasını temsil edecektir. Bu tahmin, Word2Vec tarafından elde edilen vektör değerleri arasındaki göreceliliğin ve kelimelerin anlamının ve bu çalışmada ortaya konan benzer kelimelerin benzer sıralamalara sahip olacağı hipotezinin birleşiminin bir sonucudur. Dahası, önerilen yöntemin bir diğer önemli özelliği, Word2Vec sinir ağının tanımlayamadığı kelimeleri eleyebilmesidir. Bu sinir ağının devasa bir veri kullanılarak eğitilmesinden dolayı, bu ağın tanımlayamadığı kelimelerin ya çok nadir kullanılan kelimeler olduğu veya bir yazım yanlışının söz konusu olduğu düşünülebilir, her iki durumda da sınıflandırıcı bu kelimeyi güvenli bir şekilde ihmal edebilir.

3.7. Performans Değerlendirmesi

Bu çalışmanın hipotezini ispatlamak ve önerilen yöntemi değerlendirmek için, bütüncedeki farklı kelimeler eğitim ve test kümeleri olarak bölünmüştür. Test aşamasında yer alan kelimeler eğitim aşamasında yer almamakta ve böylece yeni kelimeler gibi değerlendirmeye tabi tutulmaktadır. Bu prosedür, önerilen yöntemin herhangi bir kelimeyi eğitimde yer alsın veya almasın değerlendirme yeteneği bulunduğunu ispatlar, ve bunu da önerilen sinir ağını tekrar eğitime tabi tutma ihtiyacı olmaksızın yapar. İkinci olarak, önerilen yöntemin herhangi bir NLP uygulaması için

uygun olduğunu ispatlamak için farklı sınıflandırıcılarla aynı değerlendirmeler yapılır. Örneğin, kelime vektörlerinin kullanımı cümle içinde kelimelerin nerede yer aldığını dikkate almazken CNN ve RNN, Word2Vec iki boyutlu dizinlerle ürettiği vektörlerle bu pozisyonları dikkate alır.

3.8. Veri Toplama Süreci

Stanford Twitter Duygu Testi (The Stanford Twitter Sentiment Test—STSTd) [1] veri kümesi [79] popüler sosyal medya platformu olan Twitter'dan toplanan tweet'leri içermektedir. Tweet'te yer alan metnin duygusal anlamına göre her bir tweet pozitif veya negatif olarak kategorize edilmiştir. Ancak bu bu tweet'lerin etiketlenmesi manuel olarak değil, tweet'lerde yer alan emojilere göre yapılmaktadır. Gülen yüz gibi pozitiflik ima eden emojileri içeren tweet'ler pozitif olarak, üzgün yüz gibi negatiflik ima eden emojileri içeren tweet'ler ise negatif olarak değerlendirilir. Veri kümesi eğitim ve test aşamalarında sağlanır. Eğitim aşaması 800.000 pozitif ve 800.000 negatif tweet içerirken test aşaması 177 negatif ve 182 pozitif tweet içerir. Her bir veri kümesindeki bu belirgin sayı farkına rağmen, sınıflandırıcının test veri kümesinde kullanabileceği yeterince bilgi çıkarımının mümkün olması için bu ayrım daha önceki bütün çalışmalarda korunmuştur. Dolayısıyla, bu çalışmada yapılan deneylerde de aynı veri kümeleri kullanılmıştır.

[1] http://help.sentiment140.com/for-students

4. DENEY SONUÇLARI

Bütün modeller Python programlama diliyle kodlanmış ve yürütülmüştür [80], SVM, NB ve RF sınıflandırıcıları için Sci-Kit Learn makine öğrenmesi kütüphanesi [81], bütün yapay sinir ağları içinse Keras derin öğrenme kütüphanesi [82] kullanılmıştır. Deneyler için kullanılan bilgisayar Windows işletim sistemi kurulu, 2.4GHz Intel Core-i7 işlemcili, 16GB RAM'e sahiptir. Ayrıca aynı bilgisayarda 8GB RAM'e sahip Nvidia GTX1080Ti Graphical Processing Unit (GPU) ekran kartı bulunmaktadır.

4.1. Veri Ön İşleme

Bölüm 3.1'de anlatılan ön işleme adımlarından sonra elde edilen bütünce 412.604 farklı kelime içermekteydi. Yazım hatası olan kimi kelimeler, kısaltma veya vurgu amaçlı bazı kelimeler, örneğin today yerine 2day veya way yerine waaaaay gibi, tek bir tweet içinde yer almıştı. Bütüncede yer alan kelimelerin tamamından her bir metin için bir sayım vektörü oluşturduk, her kelimenin metin içinde yer alma sayısı veya frekansı, o kelimenin ilgili pozisyonuna yazıldı. Dolayısıyla her bir sayım vektörünün boyutu 412.604 oldu. Bu vektörleri sınıflandırıcıları eğitmek ve önerilen öznitelik belirleme yönteminden önceki performanslarını değerlendirmek için kullanıldı.

4.2. Öznitelik Belirlemeden Önce Sınıflandırıcının Performansı

SVM sınıflandırıcısının performansını metinlerden çıkarımı yapılan bütün öznitelikleri kullanarak değerlendirdik. Sınıflandırıcıyı doğrusal çekirdekle (*İng.* kernel) eğittikten sonra, sınıflandırıcının tahminlerini test veri kümesiyle değerlendirdik. Tablo 4.1'de gösterilen karışıklık matrisi SVM sınıflandırıcısının deneydeki tahminlerini özetlemektedir, Tablo 4.2'de ise aynı karışıklık matrisine dayalı olarak performans ölçümleri yer almaktadır. Bu deneyde, sınıflandırıcının her bir tahmin için, ortalama tahmin süresi 535*u*s olarak ölçülmüştür.

Tablo 4.1. Öznitelik belirleme olmaksızın SVM sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	148	29
	Pozitif	30	152

Tablo 4.2. Öznitelik belirleme olmaksızın SVM sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.8315	0.8362	0.8338	177
Pozitif	0.8398	0.8352	0.8375	182
Ort/Topl	0.8357	0.8357	**0.8357**	359
Doğruluk	**0.8357**			

Veri kümesi üzerinde önişlemden sonra elde edilen aynı sayım vektörleriyle, NB sınıflandırıcısının performansı ölçülmüştür. Bu sınıflandırıcının verdiği tahminler Tablo 4.3'te özetlenmektedir. Tablo 4.4'te ise aynı karışıklık matrisine dayalı olarak performans ölçümleri yer almaktadır. Bu deneyde, sınıflandırıcının her bir tahmin için, ortalama tahmin süresi 14*u*s olarak ölçülmüştür

Tablo 4.3. Öznitelik belirleme olmaksızın NB sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	152	25
	Pozitif	21	161

Tablo 4.4. Öznitelik belirleme olmaksızın NB sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.8786	0.8588	0.8686	177
Pozitif	0.8656	0.8846	0.8750	182
Ort/Topl	0.8720	0.8719	**0.8719**	359
Doğruluk	**0.8719**			

Bu sonuçlar NB sınıflandırıcısının daha kısa yürütme süresinde daha iyi tahminler sağladığını göstermektedir. SVM sınıflandırıcıya kıyasla ortaya çıkan bu daha kısa yürütme süresinin sebebi, NB sınıflandırıcısının daha basit hesaplamalara ihtiyaç duymasıdır, çünkü eğitim sırasında öznitelik değerleri ve etiketlerin olasılık hesapları zaten yapılmaktadır. SVM'de girdi, eğitim sırasında oluşturulan alana eşlenir ve girdinin eşleşeceği alan, yani uygun olan etiket veya ait olduğu kategori hesaplanır. Bu durum, girdideki her bir özniteliğe eşit yaklaşılmasının SVM sınıflandırıcısının performansını sınırlayabileceğinin bir göstergesidir ve SVM'nin öznitelik belirlemeye karşı yüksek duyarlılığının da bir ispatıdır.

Ayrıca, RF sınıflandırıcısının performansı da aynı prosedür takip edilerek değerlendirilmektedir, yani aynı eğitim ve test veri kümeleri burada da kullanılmaktadır. Bu sınıflandırıcının sağladığı tahminler Tablo 4.5'te özetlenmiştir, bu tablodaki değerler kullanılarak RF sınıflandırıcısının performans ölçümü de yapılmıştır ve bu sonuçlar da Tablo 4.6'da verilmiştir. Ortalama olarak, RF sınıflandırıcısının bu deneyde her bir tahmin için gereksinim duyduğu süre 28*u*s olarak ölçülmüştür.

Tablo 4.5. Öznitelik belirleme olmaksızın RF sınıflandırıcısının karışıklık matrisi.

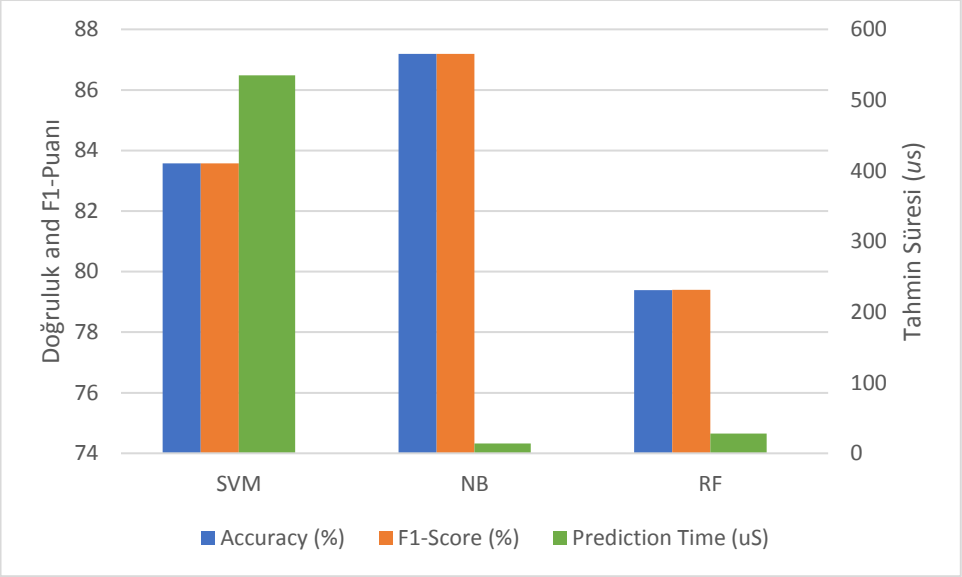
		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	142	35
	Pozitif	39	143

Tablo 4.6. Öznitelik belirleme olmaksızın RF sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.7845	0.8023	0.7933	177
Pozitif	0.8034	0.7857	0.7944	182
Ort/Topl	0.7941	0.7939	**0.7940**	359
Doğruluk	**0.7939**			

Sonuçlar tahminlerin kalitesi açısından RF sınıflandırıcısının SVM veya NB sınıflandırıcılarından daha iyi bir performans veremediğini göstermektedir, Ancak, yürütme süresi SVM sınıflandırıcıdan daha kısa olmakla beraber, NB

sınıflandırıcısından daha uzundur. Dolayısıyla, sınıflandırıcıların karmaşıklığı açısından, RF, SVM'ye göre daha az, NB'ye göre ise daha fazla hesaplama yapmaya ihtiyaç duymaktadır. Dahası, RF sınıflandırıcısı girdi vektörlerindeki özniteliklerin sadece bir alt kümesine ihtiyaç duyduğu için, sınıflandırıcıların performansı açısından bu sonuçlar öznitelik belirlemenin önemini göstermektedir. Şekil 4.1 bu deneyde değerlendirmeye tabi tutulan sınıflandırıcıların performanslarını özetlemektedir.



Şekil 4. 1. Öznitelik belirleme olmaksızın SVM, NB ve RF sınıflandırıcılarının genel özeti.

4.3. Önerilen Öznitelik Belirleme Yönteminin Performansı

Önerilen öznitelik belirleme yöntemi makine öğrenmesiye dayandığı için, modelin ortamla olan etkileşimi belirleyen kuralları tanıyabilmesi için eğitime ihtiyaç duyulmaktadır. Dahası, yansız ve mantıklı bir değerlendirme için, değerlendirmenin eğitime dahil olmayan veri veya senaryolarla yapılması önemlidir. Ayrıca, önerilen yöntem kelime bazlı çalıştığı için, verinin mevcut ayrışımı değerlendirmesi için uygun değildir. Bu yüzden bütüncede yer alan farklı kelimeler %80'i eğitim %20'si test amaçlı olarak iki gruba ayrılacaktır.

Word2Vec modeli bütüncede yer alan 412.604 farklı kelimenin her biriyle beslenir, ve sonucunda model 12.779 kelime için vektör oluşturur. Geri kalan kelimeler ise model tarafından tanınmamaktadır, yani bu kelimeler modelin eğitildiği devasa bütüncenin

içinde bulunmayan yabancı kelimelerdir. Bu adım Bölüm 3.2'de gösterildiği gibi, öznitelik belirlemenin ilk aşamasıdır. Daha sonra 10.223 kelime, SVM sınıflandırıcısının bu kelimelerin bütünceden çıkarılmasına verdiği tepkilere göre eğitim amaçlı kullanılırken, geri kalan kelimeler test için, değerlendirme amaçlı kullanılır. Değerlendirme SVM sınıflandırıcısının önerilen yöntemle tahminlerine göre yapılır, burada üç kategori bulunmaktadır; negatif, nötr ve pozitif. Modelin verdiği sürekli değerlerden kategorizasyon için belirlenen eşik değerleri -0.33 ve 0.33'tür. Değeri -0.33'ten az olan kelimeleri negatif etkiye sahip, değeri 0.33'ten çok olan kelimeleri ise pozitif etkiye sahip olarak kategorize ettik. Bu değerlerin arasında kalan kelimeler ise nötr kategoride olarak kabul edilmiştir. Bu sonuçlar Tablo 4.7'de özetlenmektedir.

Tablo 4.7. Önerilen öznitelik belirleme yönteminin performansı.

		Tahmin edilen		
		Negatif	**Neutral**	**Pozitif**
Gerçek	**Negatif**	8	0	0
	Neutral	0	238	30
	Pozitif	0	108	2172

Bu sonuçlara göre, önerilen yöntem SVM sınıflandırıcısının her bir kelimenin kategorisini tahmin etmede belirgin bir şekilde performansının artmasını sağlamıştır. Baz alınan eşik değerleriyle, tahmin edilen kategorilerde %94.6'lık bir doğruluk oranı yakalanmıştır; burada eşik değerlerinde ince bir ayarlama yaparak yöntemin doğruluk performansını arttırmak mümkün olabilir. Ancak bu eşik değerleri bütün deneylerde aynı şekilde seçilmektedir; çünkü hiperbolik tanjant aktivasyon fonksiyonunun çıktısı olan sürekli çıktı değerleri üç eşit aralığa ayrılmaktadır.

4.4. Öznitelik Belirlemeden Önce Yapay Sinir Ağının Performansı

Önerilen yapay sinir ağı Word2Vec tarafından oluşturulan vektörlere dayandığı için, söz konusu modelin oluşturduğu vektörler —önerilen yöntemin çıktısı olan sıralamalar dahil edilmeden önce— modeli hem eğitmek hem de performansını ölçmek için kullanılmaktadır, yani 12.779 vektörün içinde yer alan bütün kelimeler kullanılmaktadır. Bu düzenekle, bütüncede yer alan en uzun metin 32 kelimeden

oluşmaktadır. Dolayısıyla, her bir metin 32×300'lük dizinlere dönüştürülmektedir, 32 kelimenin her biri Word2Vec'ün çıktısı olan 300 değer cinsinden ifade edilmektedir. Benzer şekilde, uygulanan sinir ağına ait CNN, LSTM ve GRU'nun, 128, 64, 64, 32, 32 sinirden oluşan beş özel katmanından sonra gelen girdi katmanları bu boyutlara ayarlanır. Daha sonra çıktı katmanı için tamamen bağlantılı olan 64 ve 16 sinirlik katmanlar yerleştirilir. Çıktı katmanında olan ve Sigmoid aktivasyon fonksiyonunu kullanan sinirin dışında, bütün sinirler ReLU activasyon fonksiyonunu kullanırlar. Sigmoid fonksiyonunun çıktıları [0, 1] aralığındadır ve 0.5'ten büyük olan değerler pozitif kabul edilir. CNN modeli ekstra iki tane max-pooling katmanı vardır, bunlar üçüncü ve dördüncü evrişimsel katmanlardan sonra yerleştirilirler. CNN'in tahminleri Tablo 4.8'de yer alan karışıklık matrisinde özetlenmektedir, bu matrise göre aynı sınıflandırıcının performans ölçümleri ise Tablo 4.9 ile verilmiştir.

Tablo 4.8.Öznitelik belirleme olmaksızın CNN sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	136	41
	Pozitif	0	182

Tablo 4.9.Öznitelik belirleme olmaksızın CNN performans ölçümleri.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	1.0000	0.7684	0.8690	177
Pozitif	0.8161	1.0000	0.8988	182
Ort/Topl	0.9068	0.8858	**0.8962**	359
Doğruluk	**0.8858**			

CNN'in kelimelerin pozisyonunu dikkate alma yeteneği ve çok fazla sayıda yabancı kelimenin elenmiş olması SVM, NB ve RF sınıflandırıcılarından daha iyi performans vermesini sağlamıştır. Ancak, CNN için ortalama tahmin süresi 120*u*s'dir, bu da değerlendirilen sınıflandırıcılardan göreceli olarak fazladır, bu sürenin fazlalığının sebebi CNN'de yer alan hesaplamaların göreceli olarak daha karmaşık olmasıdır.

Daha sonra, LSTM sinir ağının performansı Word2Vec'ten elde edilen aynı vektörler kullanılarak değerlendirilmiştir. Tablo 4.10'da yer alan karışıklık matrisinde bu sinir

ağının tahminleri özetlenmektedir, bu matrise göre aynı sinir ağının performans ölçümleri ise Tablo 4.11 ile verilmiştir. Bu sinir ağının her bir tahmin için gereksinim duyduğu ortalama süre 749*u*s'dir.

Tablo 4.10.Öznitelik belirleme olmaksızın LSTM sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	137	40
	Pozitif	2	180

Tablo 4.11.Öznitelik belirleme olmaksızın LSTM sınıflandırıcısının performans ölçümleri.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.9856	0.7740	0.8671	177
Pozitif	0.8182	0.9890	0.8955	182
Ort/Topl	0.9007	0.8830	**0.8918**	359
Doğruluk	**0.8830**			

Sonuçlar tahminlerin kalitesi açısından bu sinir ağının CNN'in performansına çok benzediğini göstermektedir, CNN biraz daha iyi bir performans vermiştir. Ancak LSTM sinir ağının gereksinim duyduğu süre CNN'den belirgin şekilde fazladır, bunun sebebi de LSTM birimleri arasında veri akışının kontrolü için daha fazla hesaplamanın gerekliliğidir.

GRU sinir ağının performansı modelin tanıyabileceği kelimeler için Word2Vec'ten elde edilen aynı vektörler kullanılarak değerlendirilmiştir. Tablo 4.12'de yer alan karışıklık matrisinde bu sinir ağının tahminleri özetlenmektedir, bu matrise göre aynı sinir ağının performans ölçümleri ise Tablo 4.13 ile verilmiştir. Bu sinir ağının her bir tahmin için gereksinim duyduğu ortalama süre 15*u*s'dir.

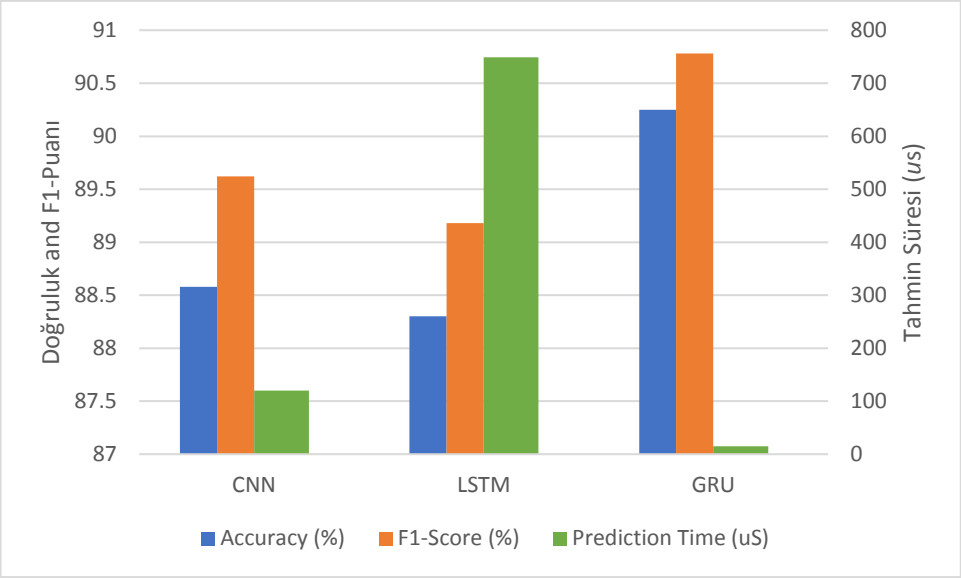
Tablo 4.12.Öznitelik belirleme olmaksızın GRU sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	145	32
	Pozitif	3	179

Tablo 4.13.Öznitelik belirleme olmaksızın GRU sınıflandırıcısının performans ölçümleri.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.9797	0.8192	0.8923	177
Pozitif	0.8483	0.9835	0.9109	182
Ort/Topl	0.9131	0.9025	**0.9078**	359
Doğruluk	**0.9025**			

Değerlendirilen bütün sınıflandırıcılar arasında, GRU hem tahminlerin kalitesi hem de her bir girdinin kategorisini tahmin etmek için gerekli olan hesaplamaların yapılması için ihtiyaç duyulan süre bağlamında en iyi performansı vermektedir. Bu kısa sürenin sebebi, bu sinir ağında CNN ve LSTM'ye göre yapılan hesaplamaların daha az karmaşık olması ve bu sinir ağının kelimelerin anlamının yanı sıra bulundukları konumu da dikkate alabilme yeteneğidir. Dolayısıyla, Word2Vec tarafından oluşturulan vektörleri kullanarak ve kelimelerin sadece sayım vektöründeki varlığını değil, anlamını ve cümle içindeki konumunu kullanarak, sınıflandırıcılar daha doğru bilgi çıkarımı yapabilmişlerdir. Öznitelik belirleme olmaksızın yapay sinir ağlarının performansı Şekil 4.2'de özetlenmektedir.



Şekil 4.2.Öznitelik belirleme olmaksızın yapay sinir ağlarının performans özeti.

4.5. Sınıflandırıcının Öznitelik Belirlemeyle Beraber Performansı

Önerilen öznitelik belirleme yöntemi kullanılarak sıralama değeri 0.33'ün üzerinde olan kelimeler seçilir, yani sınıflandırıcı üzerinde sadece pozitif bir etkiye sahip olduğu tahmin edilen kelimeler seçilmektedir. Bu seçime göre, sınıflandırma aşamasında sadece 8.264 kelime dahil edilir. Dolayısıyla, sayım vektörünün boyutu, orjinal bütüncenin boyutu olan 412.604 yerine 8.264'e inmiş olur. Ayrıca, en uzun metindeki kelime sayısı 27'dir, bu da sinir ağları için girdi boyutunun 27×300'e düşmesini sağlar.

Bu sayım vektörleri kullanılarak SVM sınıflandırıcısının performansı ölçümlenmiştir. Tablo 4.14'de yer alan karışıklık matrisinde bu sınıflandırıcının tahminleri özetlenmektedir, bu matrise göre aynı sınıflandırıcının performans ölçümleri ise Tablo 4.15'te verilmiştir. Bu sınıflandırıcının her bir tahmin için gereksinim duyduğu ortalama süre 442*us*'dir.

Tablo 4.14. Öznitelik belirlemeyle beraber SVM sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	137	40
	Pozitif	2	180

Tablo 4.15. Öznitelik belirlemeyle beraber SVM sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.8743	0.8644	0.8693	177
Pozitif	0.8696	0.8791	0.8743	182
Ort/Topl	0.8719	0.8719	**0.8719**	359
Doğruluk	**0.8719**			

Girdi vektörünün boyutlarındaki belirgin azalmaya rağmen, SVM tarafından bir girdinin kategorisini tahmin etmek için gerekli olan sürede belirgin bir azalmadan söz edilemez. Ancak, Doğruluk ve F1-Puanı değerlerinde görüldüğü gibi tahminlerin kalitesinde belirgin bir artış söz konusudur, bu da SVM sınıflandırıcısının girdi özniteliklerinin kalitesine fazlasıyla duyarlı olduğunu göstermektedir.

Önerilen öznitelik belirleme yöntemiyle seçilen kelimeler için olan sayım vektörleri kullanılarak NB sınıflandırıcının performansı ölçümlenmiştir. Tablo 4.16'da yer alan karışıklık matrisinde bu sınıflandırıcının tahminleri özetlenmektedir, bu matrise göre aynı sınıflandırıcının performans ölçümleri ise Tablo 4.17 ile verilmiştir. Bu sınıflandırıcının her bir tahmin için gereksinim duyduğu ortalama süre 11*u*s'dir.

Tablo 4.16. Öznitelik belirlemeyle beraber NB sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	150	27
	Pozitif	14	168

Tablo 4.17. Öznitelik belirlemeyle beraber NB sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.9146	0.8475	0.8798	177
Pozitif	0.8615	0.9231	0.8912	182
Ort/Topl	0.8877	0.8858	**0.8868**	359
Doğruluk	**0.8858**			

Sonuçlar, önerilen öznitelik belirleme yöntemi uygulandığında NB sınıflandırıcısının performansında düşük sayılacak bir artış olduğunu göstermektedir. Bu düşük artışın sebebi, düşük sıralı özniteliklerin düşük seviyedeki etkisinden kaynaklanmaktadır. Dolayısıyla yabancı kelimelerin ve nadiren kullanılan kelimelerin düşük olasılıkları vardır ve girdinin kategorisi açısından bu da sınıflandırıcının karar alması üzerinde düşük bir etkiye sahiptir.

Bu sayım vektörleri kullanılarak RF sınıflandırıcısının yaptığı tahminler kaydedilir ve her bir girdinin gerçek kategorisiyle kıyaslanarak RF'nin değerlendirmesi yapılır. Tablo 4.18'de yer alan karışıklık matrisinde bu sınıflandırıcının tahminleri özetlenmektedir, bu matrise göre aynı sınıflandırıcının performans ölçümleri ise Tablo 4.19'da verilmiştir. Bu sınıflandırıcının her bir tahmin için gereksinim duyduğu ortalama süre 22*u*s'dir.

Tablo 4.18. Öznitelik belirlemeyle beraber RF sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	152	25
	Pozitif	27	155

Tablo 4.19. Öznitelik belirlemeyle beraber RF sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.8492	0.8588	0.8539	177
Pozitif	0.8611	0.8516	0.8564	182
Ort/Topl	0.8552	0.8552	**0.8552**	359
Doğruluk	**0.8552**			

RF sınıflandırıcısının girdinin kategorisini tahmin etmesi girdi özniteliklerinin bir altkümesine dayalı olduğu için, ortalama her bir girdinin kategorisini tahmin süresinde esaslı bir değişiklik olmamıştır. Ancak, tahminlerin kalitesinde belirgin bir atış görülmektedir, bu da alt sıralarda yer alan özniteliklerin elenmesinden kaynaklanmaktadır. Eğitim sırasında oluşturulan karar alma ağacında bu tip

öznitelikler sınıflandırıcı üzerinde negatif bir etkiye sahip olabilir. Dolayısıyla bu özniteliklerin elemlenmesi sınıflandırıcının performansını arttırabilmiştir.

Word2Vec sınıflandırıcısı tarafından oluşturulan vektörler kullanılarak, ve öznitelik belirleme uygulandıktan sonra en uzun metin 27 kelime içerdiği için, yapay sinir ağları her bir metin için 27×300'lük dizinler kullanılarak değerlendirilmiştir. Tablo 4.20'de yer alan karışıklık matrisinde bu düzeneğe göre CNN sınıflandırıcısının tahminleri özetlenmektedir, bu matrise göre aynı sınıflandırıcının performans ölçümleri ise Tablo 4.21'de verilmiştir. Bu sınıflandırıcının her bir tahmin için gereksinim duyduğu ortalama süre 41*u*s'dir.

Tablo 4.20. Öznitelik belirlemeyle beraber CNN sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	152	25
	Pozitif	27	155

Tablo 4.21. Öznitelik belirlemeyle beraber CNN sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	1.0000	0.8588	0.9240	177
Pozitif	0.8792	1.0000	0.9357	182
Ort/Topl	0.9388	0.9304	**0.9345**	359
Doğruluk	**0.9304**			

CNN'de bir katmandan diğerine aktarılan verinin büyüklüğü girdinin büyüklüğüne bağlı olduğu için, bu sinir ağına yönlendirilen girdinin belirgin şekilde küçültülmesi, girdilerin işlenmesi süresini belirgin şekilde azaltmıştır. Ayrıca, nötr olan veya negatif etkiye sahip olan kelimelerin elenmesi de CNN sınıflandırıcısının performansını belirgin şekilde iyileştirmiştir.

Önerilen öznitelik belirleme yöntemiyle seçilen kelimeler için Word2Vec sınıflandırıcısı tarafından oluşturulan vektörlerle elde edilen aynı dizinler kullanılarak LSTM sınıflandırıcısı da değerlendirilmiştir. Tablo 4.22'de yer alan karışıklık matrisinde aynı düzeneğe göre bu sınıflandırıcının tahminleri özetlenmektedir, bu

matrise göre aynı sınıflandırıcının performans ölçümleri ise Tablo 4.23 ile verilmiştir. Bu sınıflandırıcının her bir tahmin için gereksinim duyduğu ortalama süre 94*u*s'dir.

Tablo 4.22. Öznitelik belirlemeyle beraber LSTM sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	148	29
	Pozitif	3	179

Tablo 4.23. Öznitelik belirlemeyle beraber LSTM sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.9801	0.8362	0.9024	177
Pozitif	0.8606	0.9835	0.9179	182
Ort/Topl	0.9195	0.9109	**0.9152**	359
Doğruluk	**0.9109**			

LSTM sinir ağının karmaşıklığı dolayısıyla girdinin boyutlarının küçültülmesi, girdinin kategorisinin tahmini için gerekli olan işlemler için gerekli olan süreyi belirgin şekilde azaltmıştır. Kategorilerin tahmininde hiç bir katkısı olmadığı düşünülen kelimelerin elenmesi de aynı şekilde tahminlerin kalitesini arttırmıştır.

Word2Vec kelime kalıplama modeli tarafından oluşturulan vektörlere dayalı aynı girdileri kullanarak GRU sinir ağı da değerlendirilmiştir. Tablo 4.24'te yer alan karışıklık matrisinde aynı düzeneğe göre bu sinir ağının tahminleri özetlenmektedir, bu matrise göre aynı sinir ağının performans ölçümleri ise Tablo 4.25'te verilmiştir. Bu sinir ağının her bir tahmin için gereksinim duyduğu ortalama süre 7*u*s'dir.

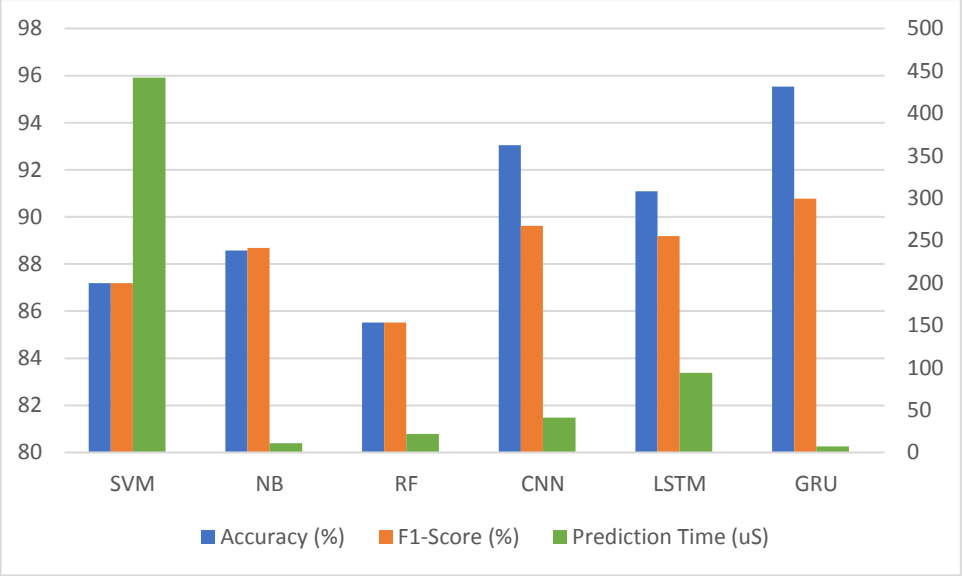
Tablo 4.24. Öznitelik belirlemeyle beraber GRU sınıflandırıcısının karışıklık matrisi.

		Tahmin edilen	
		Negatif	**Pozitif**
Gerçek	**Negatif**	164	13
	Pozitif	3	179

Tablo 4.25. Öznitelik belirlemeyle beraber GRU sınıflandırıcısının performans ölçümü.

	Kesinlik	**Hassasiyet**	**F1-Puanı**	**Destek**
Negatif	0.9820	0.9266	0.9535	177
Pozitif	0.9323	0.9835	0.9572	182
Ort/Topl	0.9568	0.9554	**0.9561**	359
Doğruluk	**0.9554**			

Önerilen öznitelik belirleme yöntemi olmaksızın yüksek performans gösteren GRU, önerilen yöntem kullanıldığında da üstünlüğünü korumaya devam etmiştir. Bunun yanısıra, her bir girdinin kategorisini tahmin etme süresi belirgin şekilde azalmıştır. Bu performans, GRU birimlerinin girdiler içindeki pozisyonlarını dikkate almasının, hesaplamaların daha az karmaşık olmasının ve kelimelerin Word2Vec ile temsil edilmesinin sonucudur. Önerilen öznitelik belirleme yöntemi kullanıldığında sınıflandırıcıların performansları Şekil 4.3'te özetlenmiştir.



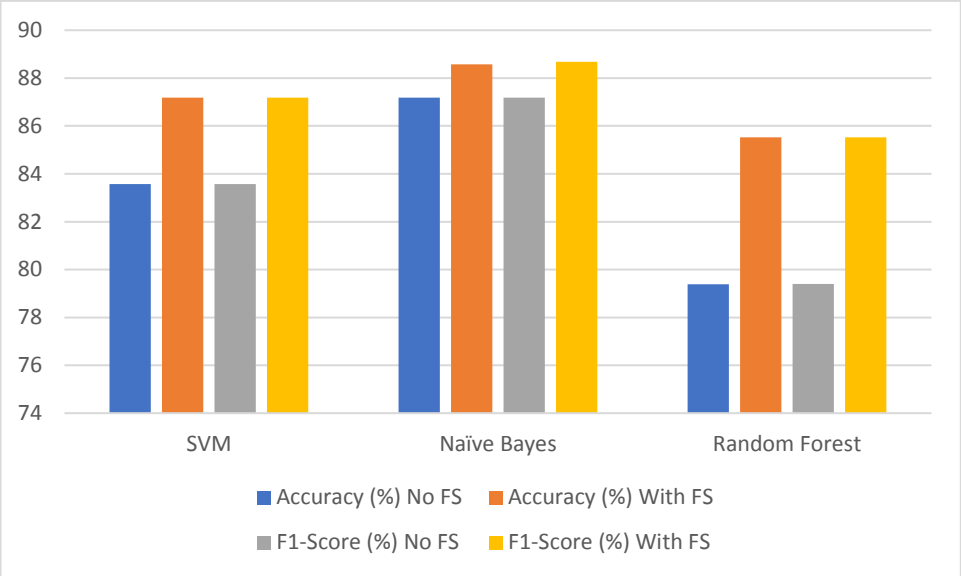
Şekil 4.3. Önerilen öznitelik belirleme yönteminin seçtiği öznitelikler kullanıldığında değerlendirilen sınıflandırıcıların performans özeti.

5. SONUÇLARIN ÖZETİ VE TARTIŞMA

Diğer öznitelik belirleme yöntemlerine benzer bir şekilde, önerilen yöntemin amacı da farklı sınıflandırıcı türlerinin performansını arttırmaktır. Böyle bir performans artışı, iki cihetten elde edilebilir; sınıflandırıcının yaptığı tahminlerin kalitesini artırarak ve sınıflandırıcı tarafından üretilen modelin karmaşıklığını azaltarak. Bu bağlamda, sınıflandırıcı daha kısa sürede daha doğru tahminlerde bulunmaktadır. Bu nedenle, önerilen öznitelik belirleme yönteminin performansını ortaya koyabilmek için, sayım vektörlerine dayalı olan sınıflandırıcıların yaptığı tahminlerin kalitesi cihetinden performanslarındaki gelişmeler Tablo 5.1'de verilmekte ve Şekil 5.1'de ise görsel eşliğinde sunulmaktadır.

Tablo 5.1. Önerilen öznitelik belirleme yöntemi kullanılarak SVM, NB ve RF sınıflandırıcıların yaptığı tahminlerin kalitesinde görülen artış. (ÖB=Öznitelik Belirleme)

	Doğruluk (%)			**F1-Puanı (%)**		
	ÖB Yok	**ÖB Var**	**Gelişme**	**ÖB Yok**	**ÖB Var**	**Gelişme**
SVM	83.57	87.19	3.62	83.57	87.19	3.62
Naïve Bayes	87.19	88.58	1.39	87.19	88.68	1.49
Rastgele Orman	79.39	85.52	6.13	79.4	85.52	6.12



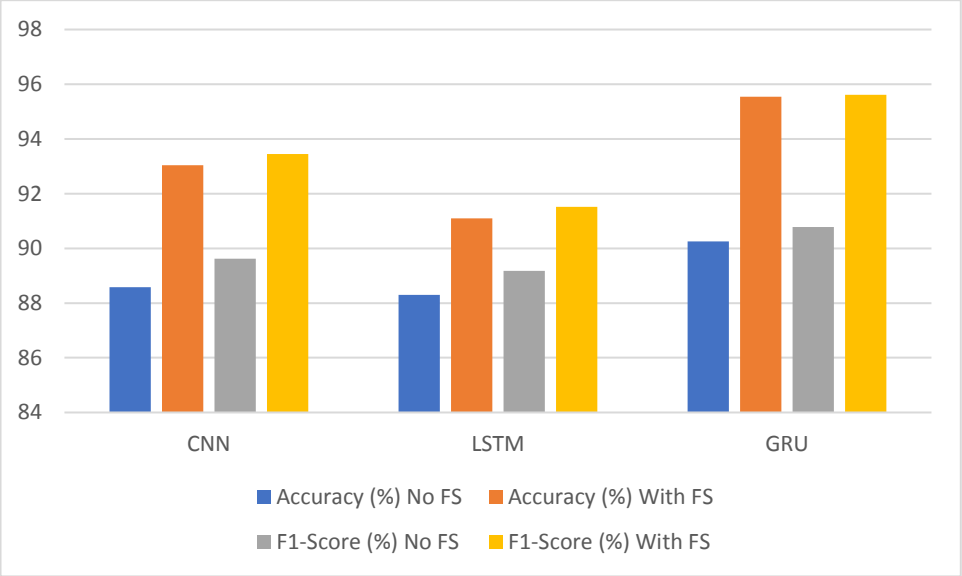
Şekil 5.1. Önerilen öznitelik belirleme yöntemi kullanılarak veya kullanılmadan SVM, NB ve RF sınıflandırıcıların performans ölçümünün grafiği.

Bu kıyaslamalar, önerilen öznitelik belirleme yönteminin en büyük etkiyi RF sınıflandırıcısı üzerinde gösterdiğini ortaya koymaktadır. Bunun sebebi, RF tarafından yürütülen karar alma sürecinde girdi özniteliklerinin bir alt kümesinin kullanılmasıdır. Dahası, ormanda birden fazla ağaç yapısının bulunması ve verinin bu ağaçlar arasında ayrıştırılmasına ihtiyaç duyulması, daha az etkin olan öznitelikler üzerinde hatalı bir vurguda bulunabilir, bu da nihayetinde ormanın genel kararlarını etkileyecektir. Dolayısıyla, sıralaması daha düşük olan özniteliklerin elenmesi RF sınıflandırıcısının performansını belirgin şekilde arttırmıştır. Ayrıca, tahminlerin kalitesi açısından önerilen yöntemin en az etkisi NB sınıflandırıcısı üzerinde görülmektedir. Bu düşük etkinin nedeni, çok düşük olasılığa sahip olan fazla sayıdaki yabancı kelimenin varlığıdır, bu kelimelerin NB sınıflandırıcısının karar alması üzerinde çok az etkisi bulunmaktadır.

Benzer şekilde YSA sınıflandırıcılarında gözlenen iyileşmeler de ölçümlenmiş ve bu gelişmeler Tablo 5.2'de ve Şekil 5.2'de gösterilmiştir. Kıyaslamalar önerilen yöntemin bu sınıflandırıcıların performanslarını da arttırdığını göstermektedir. Bununla birlikte, SVM, NB ve RF sınıflandırıcılarının performans gelişimleriyle kıyaslandığında, farklı YSA türlerinin performansındaki iyileşmeler birbirine benzeşmektedir. Bu durumun sebebi, yapay sinir ağlarının gerekli olan tahminlerde bulunma performansları üzerinde pozitif etkisi olduğu bilinen öznitelikleri algılayabilme yetenekleridir.

Tablo 5.2. Önerilen öznitelik belirleme yöntemi kullanılarak CNN, LSTM ve GRU sınıflandırıcıların yaptığı tahminlerin kalitesinde görülen artış.

	Doğruluk (%)			F1-Puanı (%)		
	ÖB Yok	**ÖB Var**	**Gelişme**	**ÖB Yok**	**ÖB Var**	**Gelişme**
CNN	88.58	93.04	4.46	89.62	93.45	3.83
LSTM	88.3	91.09	2.79	89.18	91.52	2.34
GRU	90.25	95.54	5.29	90.78	95.61	4.83

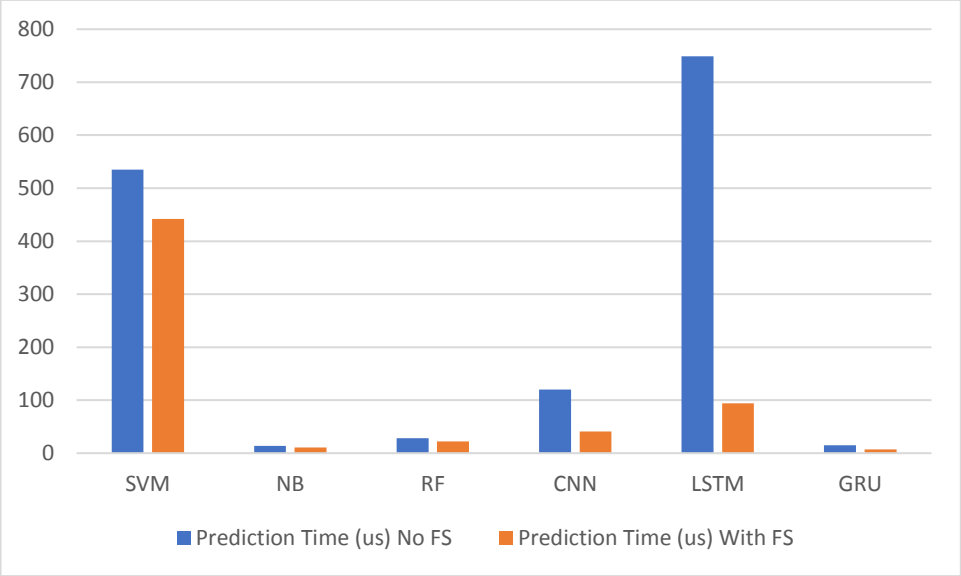


Şekil 5.2. Önerilen öznitelik belirleme yöntemi kullanılarak veya kullanılmadan CNN, LSTM ve GRU sınıflandırıcıların performans ölçümünün grafiği.

Önerilen öznitelik belirleme yönteminin tahminlerine dayalı olarak sadece üst sıralarda yer alan özniteliklerin seçilmesi sonucu ihtiyaç duyulan sürenin kısalması Tablo 5.3'te verilmektedir. Aynı kıyaslamanın grafiği de Şekil 5.3'te verilmektedir, görüldüğü gibi en büyük etki LSTM'de ortaya çıkmıştır, bunun sebebi LSTM birimlerinin karmaşıklığının çok yüksek olmasıdır. Dolayısıyla girdinin boyutundaki azalma, her katmandaki verilerin büyüklüğünün belirgin bir şekilde girdinin boyutlarına dayandığı CNN'in azalmasından daha fazladır.

Tablo 5.3. Değerlendirmeye tabi tutulan sınıflandırıcılarda, bir girdinin sınıfını tahmin edebilmek için gerekli olan ortalama sürede görülen azalma.

	Tahmin Süresi (*us*)		
	ÖB Yok	**ÖB Var**	**Azalma**
SVM	535	442	93
NB	14	11	3
RF	28	22	6
CNN	120	41	79
LSTM	749	94	655
GRU	15	7	8



Şekil 5.3. Farklı sınıflandırıcılarda her bir tahmin için harcanan ortalama sürelerin grafiği.

Önerilen yöntemle belirlenen öznitelikleri kullanan sınıflandırıcıların ve mevcut en son yöntemlerin performans karşılaştırması Tablo 5.4'te verilmektedir. Bu karşılaştırma, CNN'le elde edilen performansın önerdiğimiz yöntemden daha önce Jianqiang ve diğ. [43] ve Cicero ve Maria [61] tarafından önerilen yöntemlerin performansına benzer olduğunu göstermektedir. Bununla birlikte, önerdiğimiz öznitelik belirleme yönteminin kullanılması ile CNN'in bu yöntemlerden önemli ölçüde daha iyi performans verdiği görülmektedir; bu da önerdiğimiz yöntemin önemini göstermektedir. Dahası, aynı tasarımı kullandığımızda GRU sinir ağının CNN'den daha iyi bir performansının olduğu ortadadır. Dolayısıyla, kelimelerin konumlarını ve anlamlarını dikkate alma özelliği olan Word2Vec kelime kalıplama

modeline dayanan metin sınıflandırma uygulamaları için GRU sinir ağı daha uygundur. Ayrıca, önerdiğimiz öznitelik belirleme yöntemi, sayım vektörlerine dayanan diğer sınıflandırıcı türlerinin performansını da geliştirmeyi başarmış ve böylece daha önceki çalışmalarda kullanılan benzer yöntemlerden daha iyi bir performans sergilemiştir. Bu sonuçlar, kelimeleri istatistiksel yaklaşımlar yerine anlamlarına göre sıralayan öznitelik belirleme yöntemimizin, her tür sınıflandırıcının performansını geliştirme yeteneğini kanıtlamaktadır.

Tablo 5.4. Mevcut en son yöntemlerle performans karşılaştırması.

Çalışma	**Sınıflandırıcı**	**Doğruluk (%)**
Jianqiang ve diğ. [43]	BoW-SVM	68.81
	BoW-Linear Regression	71.04
	GloVe-SVM	81.61
	GloVe-LR	81.62
	CloVe-CNN	**87.62**
Cicero ve Maria [61]	CNN	**86.40**
Saif ve diğ. [62]	Sentiment Lexicon	84.70
Bu çalışma	SVM	87.19
	Naïve Bayes	88.58
	Random Forest	85.52
	CNN	93.04
	LSTM	91.09
	GRU	**95.54**

6. SONUÇ VE TAVSİYELER

Doğal dilde yazılmış cümlelerin karmaşık yapısına ek olarak, metinlerde yer alan karmaşık örüntüler ve çok sayıdaki öznitelik, bilgi çıkarımı amaçlı olarak bu metinlerin işlenmesini zorlaştırmaktadır. Bu amaçla makine öğrenmesi teknikleri kullanılmaktadır; bu öğrenmede bir ortamla etkileşimde bulunularak aynı ortam hakkında bilgi çıkarımı yapılmaya çalışılır. Etkileşim türüne bağlı olarak üç tür makine öğrenmesi çeşidi vardır, denetimsiz, denetimli ve takviyeli öğrenme. Her bir girdinin kategorisini tahmin etmek için, aynı girdinin öznitelik örüntülerini tanımaya çalışan sınıflandırma kullanılır. Takviyeli öğrenmede, ortam ile etkileşime girerek ve ortamın belli etkilere verdiği tepkiler ölçümlenerek öğrenme gerçekleşir.

Duygusal analiz amaçlı olarak sınıflandırma yaygın şekilde kullanılmaktadır, burada metin girdileri içeriklerine bağlı olarak pozitif veya negatif olarak sınıflandırılır. Bununla birlikte, metinlerin çok boyutluluğu ve tanınması gereken karmaşık örüntüler nedeniyle, nötr veya negatif etkiye sahip özniteliklerin, yani kelimelerin elenmesi sınıflandırıcının performansını önemli ölçüde artırma potansiyeline sahiptir. Bu performans artışının temelde iki ciheti vardır; sınıflandırıcının yaptığı tahminlerin kalitesi, yani doğruluğu ve bu tahminleri netice verecek hesaplamaları icra etmek için gereken süre.

Bu çalışmada, bütünce içerisinde yer alan kelimeleri sınıflandırma aşamasında önemine göre sıralayan bir öznitelik belirleme yöntemi önerilmektedir. Önerilen yönteme göre bir kelimenin sıralamasını tahmin etmede aynı kelimenin anlamı belirleyici olur, bu da diğer kelimelerden çıkarımı yapılan bilgilere dayanarak belirlenir. Sınıflandırıcının üzerindeki etkisi bilinen bir kelimeyle eş anlamlı olan veya benzer bir anlamı taşıyan kelimenin benzer bir etkiye sahip olacağı tahmin edilebilir. Bu yüzden, yeni bir kelimenin bütünce içindeki görünümüne bağlı olarak sıralamasını değerlendirmek için karmaşık istatistiksel hesaplamalara gerek kalmayacaktır. Önerilen yöntem SVM sınıflandırıcısının bir kelimenin bütünceden çıkarılmasına tepkisini ölçerek ve takviyeli öğrenme yöntemiyle eğitilir. Her bir kelime, kelime kalıplama kullanılarak çok boyutlu bir uzaydaki konumunu temsil eden sayısal bir

vektöre dönüştürülür. Bu konum kelimenin anlamını yansıtır; dolayısıyla konumlar arası mesafeler kelimeler arasındaki ilişkileri yansıtacaktır.

Önerilen öznitelik belirleme yönteminin performans değerlendirmesi, vektörleri veya iki boyutlu dizinleri girdi olarak kabul eden farklı tip sınıflandırıcılar kullanılarak yapılmaktadır. Sadece vektör girdilerini işleyebilen SVM, NB ve RF gibi sınıflandırıcılar için, her metin bir sayım vektörüne dönüştürülür, bu vektörde her kelimenin metin içindeki frekansı o kelimeye karşılık gelen konuma yazılır. Ayrıca, CNN, LSTM ve GRU gibi belirli yapay sinir ağları için gereksinimlerine uygun olarak bütüncedeki her bir kelime, kelime kalıplama kullanılarak 300 değerli olarak temsil edilir. Önerilen öznitelik belirleme yöntemi kullanılarak, bütüncenin orijinalindeki toplam 412.604 kelime yerine sadece 8.264 farklı kelimeyle temsil edilmesi mümkün olmuştur. Önerilen yöntemle belirlenen özniteliklerin kullanımı, değerlendirmeye tabi tutulan bütün sınıflandırıcıların performansını artırabilmiştir; söz konusu sınıflandırıcıların performansları literatürde mevcut olan en son yöntemlerden daha iyi performans göstermiştir.

Gelecekte yapılması planlanan çalışmalarda, önerilen yöntemin performansı tek bir sınıflandırıcı kullanmak yerine bir dizi sınıflandırıcı kullanılarak eğitilecektir. Böyle bir eğitim yöntemi daha uzun zaman alacak olsa da, birden fazla sınıflandıranın kullanılması, değerlendirilmekte olan kelimeler için daha doğru bir sıralama sağlayabilir ve neticede, önerilen modelden daha iyi tahminler elde edilebilir. Bununla birlikte, bazı sınıflandırıcılar belirli kelimeleri doğru şekilde elemede etkin olamayabileceği için ve söz konusu kelimeler tek bir sınıflandırıcı tarafından üretilen modelde elenmiş olduğu için, hesaplanan sıralamaların daha az doğru olması da mümkündür.

KAYNAKLAR

[1] Leeflang, P. S., Verhoef, P. C., Dahlström, P. & Freundt, T. (2014). Challenges and solutions for marketing in a digital era. *European management journal*, *32*(1), 1-12.

[2] Haff, M. W., Fahey, C. S., Curtis, D. B., Larson, M. J. & Clarke, C. D. (2017). *U.S. Patent No. 9,626,655*. Washington, DC: U.S. Patent and Trademark Office.

[3] Cifuentes, M., Davis, M., Fernald, D., Gunn, R., Dickinson, P. & Cohen, D. J. (2015). Electronic health record challenges, workarounds, and solutions observed in practices integrating behavioral health and primary care. *The Journal of the American Board of Family Medicine*, *28*(Supplement 1), S63-S72.

[4] Young, T., Hazarika, D., Poria, S. & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, *13*(3), 55-75.

[5] Schmidt, A. & Wiegand, M. (2017). A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media* (pp. 1-10).

[6] Zhou, K., Qiao, Y. & Xiang, T. (2018). Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[7] Lison, P. (2015). An introduction to machine learning, *Language Technology Group (LTG)*, 1, 35.

[8] Amershi, S. & Conati, C. (2007). Unsupervised and supervised machine learning in user modeling for intelligent learning environments. In *Proceedings of the 12th international conference on Intelligent user interfaces*, 72-81.

[9] Zhou, D., Huang, J. & Schölkopf, B. (2007). Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, 1601-1608.

[10] Van Hasselt, H., Guez, A. & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

[11] Mnih,V. K., Kavukcuoglu, D., Silver, A. A. & Rusu, J. (2015). Human-level control through deep reinforcement learning, *Nature*, 518, 529.

[12] Tripathy, A., Agrawal, A. & Rath, S. K. (2016). Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications*, *57*, 117-126.

[13] Suthaharan, S. (2016). Machine learning models and algorithms for big data classification. *Integr. Ser. Inf. Syst*, *36*, 1-12.

[14] El-Halees, A. M. (2015). Arabic text classification using maximum entropy. *IUG Journal of Natural Studies*, *15*(1).

[15] Rahimi, A., Benatti, S., Kanerva, P., Benini, L. & Rabaey, J. M. (2016, October). Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, 1-8.

[16] Alghamdi, R. & Alfalqi, K. (2015). A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, *6*(1).

[17] Wang, Y., Zhou, Z., Jin, S., Liu, D. & Lu, M. (2017). Comparisons and selections of features and classifiers for short text classification. In *IOP Conference Series: Materials Science and Engineering*, 261,1, 218.

[18] Wang, P., Xu, B., Xu, J., Tian, G., Liu, C. L. & Hao, H. (2016). Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, *174*, 806-814.

[19] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111-3119.

[20] Lilleberg, J., Zhu, Y. & Zhang, Y. (2015). Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pp. 136-140.

[21] Pranckevičius, T. & Marcinkevičius, V. (2017). Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, *5*(2), 221.

[22] Liu, Y., Bi, J. W. & Fan, Z. P. (2017). Multi-class sentiment classification: The experimental comparisons of feature selection and machine learning algorithms. *Expert Systems with Applications*, *80*, 323-339.

[23] Uysal, A. K. (2016). An improved global feature selection scheme for text classification. *Expert systems with Applications*, *43*, 82-92.

[24] Porter, M. F., (2001). Snowball: A language for stemming algorithms, ed.

[25] Schofield, A. & Mimno, D. (2016). Comparing apples to apple: The effects of stemmers on topic models. *Transactions of the Association for Computational Linguistics*, *4*, 287-300.

[26] Issac, B. & Jap, W. J. (2009). Implementing spam detection using Bayesian and Porter Stemmer keyword stripping approaches. In *TENCON 2009-2009 IEEE Region 10 Conference*, 1-5.

[27] Zhang, Y., Jin, R. & Zhou, Z. H. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1-4), 43-52.

[28] Fillmore, N., Goldberg, A. B. & Zhu, X. (2008). *Document recovery from bag-of-word indices*. University of Wisconsin-Madison Department of Computer Sciences.

[29] Ma, L. & Zhang, Y. (2015). Using Word2Vec to process big text data. In *2015 IEEE International Conference on Big Data (Big Data)*, 2895-2897.

[30] Church, K. W. (2017). Word2Vec. *Natural Language Engineering*, *23*(1), 155-162.

[31] Ji, S., Satish, N., Li, S. & Dubey, P. K. (2019). Parallelizing word2vec in shared and distributed memory. *IEEE Transactions on Parallel and Distributed Systems*, *30*(9), 2090-2100.

[32] Pennington, J., Socher, R. & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532-1543.

[33] Acemoglu, D. & Restrepo, P. (2018). *Artificial intelligence, automation and work* (No. w24196). National Bureau of Economic Research.

[34] Bini, S. A. (2018). Artificial intelligence, machine learning, deep learning, and cognitive computing: what do these terms mean and how will they impact health care?. *The Journal of arthroplasty*, *33*(8), 2358-2361.

[35] Deng, L. (2018). Artificial intelligence in the rising wave of deep learning: The historical path and future outlook [perspectives]. *IEEE Signal Processing Magazine*, *35*(1), 180-177.

[36] Micheli-Tzanakou, E. (1999). *Supervised and unsupervised pattern recognition: feature extraction and computational intelligence*. CRC press.

[37] Suthaharan, S. (2014). Big data classification: Problems and challenges in network intrusion prediction with machine learning. *ACM SIGMETRICS Performance Evaluation Review*, *41*(4), 70-73.

[38] Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B. & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, *106*(9-10), 1469-1495.

[39] Chwialkowski, K., Strathmann, H. & Gretton, A. (2016). A kernel test of goodness of fit. JMLR: Workshop and Conference Proceedings.

[40] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J. & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, *13*(4), 18-28.

[41] Meyer, D. & Wien, F. T. (2001). Support vector machines. *Porting R to Darwin/X11 and Mac OS X*, *1*, 23.

[42] Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

[43] Jianqiang, Z., Xiaolin, G. & Xuejun, Z. (2018). Deep convolution neural networks for twitter sentiment analysis. *IEEE Access*, *6*, 23253-23260.

[44] Rish, I. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 3(22), 41-46.

[45] Ting, S. L., Ip, W. H. & Tsang, A. H. (2011). Is Naive Bayes a good classifier for document classification. *International Journal of Software Engineering and Its Applications*, *5*(3), 37-46.

[46] Safavian, S. R. & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, *21*(3), 660-674.

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

[48] Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217-222.

[49] Shaikhina, T., Lowe, D., Daga, S., Briggs, D., Higgins, R. & Khovanova, N. (2019). Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation. *Biomedical Signal Processing and Control*, *52*, 456-462.

[50] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

[51] Esser, S. K., Appuswamy, R., Merolla, P., Arthur, J. V. & Modha, D. S. (2015). Backpropagation for energy-efficient neuromorphic computing. In *Advances in neural information processing systems*, 1117-1125.

[52] Montufar, G. F., Pascanu, R., Cho, K. & Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, 2924-2932.

[53] Maclaurin, D., Duvenaud, D. & Adams, R. (2015, June). Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, 2113-2122.

[54] Lee, J. H., Delbruck, T. & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, *10*, 508.

[55] Nahato, K. B., Harichandran, K. N. & Arputharaj, K. (2015). Knowledge mining from clinical datasets using rough sets and backpropagation neural network. *Computational and mathematical methods in medicine*.

[56] Kayalibay, B., Jensen, G. & van der Smagt, P. (2017). CNN-based segmentation of medical imaging data. *arXiv preprint arXiv:1701.03056*.

[57] Lu, Y., Zhu, S. C. & Wu, Y. N. (2016). Learning FRAME models using CNN filters. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[58] Tolias, G., Sicre, R. & Jégou, H. (2015). Particular object retrieval with integral max-pooling of CNN activations. *arXiv preprint arXiv:1511.05879*.

[59] Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F. & Hao, H. (2015, June). Short text clustering via convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing* (pp. 62-69).

[60] Chen, D., Bolton, J. & Manning, C. D. (2016). A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

[61] Dos Santos, C. & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (pp. 69-78).

[62] Saif, H., He, Y., Fernandez, M. & Alani, H. (2014). Adapting sentiment lexicons using contextual semantics for sentiment analysis of twitter. In *European Semantic Web Conference* (pp. 54-63). Springer, Cham.

[63] Zaremba, W., Sutskever, I. & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

[64] Sutskever, I., Vinyals, O. & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104-3112.

[65] Liu, P., Qiu, X. & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

[66] Kuncoro, A., Ballesteros, M., Kong, L., Dyer, C., Neubig, G. & Smith, N. A. (2016). What do recurrent neural network grammars learn about syntax?. *arXiv preprint arXiv:1611.05774*.

[67] Sak, H., Senior, A. W. & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

[68] Gers, F. A., Schmidhuber, J. & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

[69] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[70] Oh, J., Guo, X., Lee, H., Lewis, R. L. & Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*,2863-2871.

[71] Chen, Y. & Kulla, E. (2019). A Deep Q-Network with Experience Optimization (DQN-EO) for Atari's Space Invaders. In *Workshops of the International Conference on Advanced Information Networking and Applications*, 351-361.

[72] Yoon, S. & Kim, K. J. (2017, August). Deep Q networks for visual fighting game AI. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 306-308.

[73] Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, 153-160.

[74] Hussein, A., Uçan, O. & Bayat, O., (2019). Centralized Reinforcement Learning for the Internet of Things Devices, *AURUM Journal of Engineering Systems and Architecture*.

[75] Foerster, J., Assael, I. A., De Freitas, N. & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*, 2137-2145.

[76] Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D. & Blundell, C. (2017). Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2827-2836.

[77] Osband, I., Blundell, C., Pritzel, A. & Van Roy, B. (2016). Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems*, 4026-4034.

[78] Sokolova, M. & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, *45*(4), 427-437.

[79] Go, A., Bhayani, R. & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N project report*, *Stanford*, *1*(12), 2009.

[80] Go, A., Bhayani, R. & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N project report*, *Stanford*, *1*(12), 2009.

[81] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*(Oct), 2825-2830.

[82] Chollet, F. (2018). Keras: The python deep learning library. *Astrophysics Source Code Library*.

EKLER

EK 1: Ön İşlem Adımının Yürütülmesi

```
In [ ]: import gensim
        from numpy import *
        from pandas import read_csv
        import re, string
        from nltk.stem import ISRIStemmer
        from nltk.corpus import stopwords

In [ ]: ps=ISRIStemmer()

In [ ]: tr=array(read_csv('training.csv',delimiter=',',header=None,encoding='latin-1'))[:,[0,-1]]
        for x in range(tr.shape[0]):
            tr[x,1]=re.sub(r'^\s+|\s+$','',re.sub(r'\s+',' ',' '.join([ps.stem(re.sub(r'\b\S{1}\b','',re.sub(r'\b[0-9]+\b\s*',
                                                ' number',x).translate(str.maketrans('', '', string.punctuation)))) for x in re.sub(r'@\S+',''
                                                ,re.sub(r'\s+',' ', re.sub(r'#\S+','', re.sub(r'http\S+','url',re.sub(r'6\w{1,5};','',
                                                tr[x,1])))))).split(' ') if not x in stopwords.words('english')])))
        save('Training',tr)

In [ ]: ts=array(read_csv('testdata.csv',delimiter=',',header=None,encoding='latin-1'))[:,[0,-1]]
        for x in range(ts.shape[0]):
            ts[x,1]=re.sub(r'^\s+|\s+$','',re.sub(r'\s+',' ',' '.join([ps.stem(re.sub(r'\b\S{1}\b','',re.sub(r'\b[0-9]+\b\s*',
                                                ' number',x).translate(str.maketrans('', '', string.punctuation)))) for x in re.sub(r'@\S+',''),
                                                re.sub(r'\s+',' ', re.sub(r'#\S+','', re.sub(r'http\S+','url', re.sub(r'6\w{1,5};','',
                                                ts[x,1])))))).split(' ') if not x in stopwords.words('english')])))
        save('Testing',ts)
```

Kod	**Açıklaması**
re.sub(r'^\s+|\s+$',"	Art arda gelen boşlukların kaldırılması.
re.sub(r'\b[0-9]+\b\s*', ' number'	Herhangi bir sayının "number" (İng. "sayı") ifadesiyle değiştirilmesi.
translate(str.maketrans(", ", string.punctuation))	Noktalama işaretlerinin kaldırılması.
re.sub(r'http\S+','url'	http veya https ile başlayan bütün URL adreslerinin "url" ifadesiyle değiştirilmesi.
if not x in stopwords.words('english')	stop words (etksiz kelimeler) sözlüğünde yer alan kelimelerin dikkate alınmaması.

EK 2: Yapay Sinir Ağlarında Öznitelik Sıralamasının İşlenişi

```
In [ ]: model=Sequential()
        model.add(Dense(512,activation='relu', input_dim=300))
        model.add(Dense(256,activation='relu'))
        model.add(Dropout(.2))
        model.add(Dense(128,activation='relu'))
        model.add(Dropout(.2))
        model.add(Dense(64,activation='relu'))
        model.add(Dropout(.2))
        model.add(Dense(32,activation='relu'))
        model.add(Dropout(.2))
        model.add(Dense(16,activation='relu'))
        model.add(Dropout(.2))
        model.add(Dense(8,activation='relu'))
        model.add(Dense(1,activation='tanh'))
        model.compile(optimizer='adam',loss='mse',metrics=['mse'])
```

EK 3: Yapay Sinir Ağının Önerilen Öznitelik Belirleme Yöntemine Göre Eğitilmesi

Word2Vec modelinin ve ön işleme tabi tutulmuş eğitim ve test veri kümelerinin yüklenmesi.

```
In [ ]: model = gensim.models.KeyedVectors.load_word2vec_format('SimModel.bin', binary=True)

In [ ]: ts=load('Training.npy',allow_pickle=True)
        xtr=tr[:,1]
        ytr=tr[:,0].astype(int)
        ts=load('Testing.npy',allow_pickle=True)
        xts=ts[:,1]
        yts=ts[:,0].astype(int)
```

Ön işleme tabi tutulmuş metnin Sayım Vektörlerine Çevrilmesi.

```
In [ ]: cv=CountVectorizer()
        cv.fit(xtr)

In [ ]: trcv=cv.transform(xtr)
        tscv=cv.transform(xts)
```

Seçili öznitelikleri kullanarak SVM'nin tahminlerinin doğruluğunu ölçümleyen bir uyum fonksiyonunun tanımlanması.

```
In [ ]: def fitness(sFeatures):
            svm=SVC(kernel='linear',random_state=101)
            strx=trcv[:,argwhere(sFeatures==1).reshape(-1)]
            svm.fit(strx,ytr)
            p=svm.predict(tscv[:,argwhere(sFeatures==1).reshape(-1)])
            return metrics.accuracy_score(yts,p)
```

Tek tek her bir özniteliği ortadan kaldırırarak uyum fonksiyonunun ölçümlenmesi. Kontrol uyumu, yani bütün özniteliklerin kullanımıyla elde edilen değerlerle eldeki özniteliğin kullanımını kıyaslayarak o özniteliğin -1, 0, 1 değerlerinden biriyle sıralanması.

EK 3: Yapay Sinir Ağının Önerilen Öznitelik Belirleme Yöntemine Göre Eğitilmesi

```
In [ ]: features=ones(trcv.shape[1])
        words=cv.get_feature_names()
        desc=zeros((len(words),300))
        lbls=zeros((len(words)))
        c=0
        ctrl=fitness(features)
        for x in pb(range(features.shape[0])):
            try:
                desc[c]=model[words[x]]
                features[x]=-1
                nctrl=fitness(features)
                if nctrl==ctrl:
                    lbls[c]=0
                elif nctrl<ctrl:
                    lbls[c]=1
                    features[x]=0
                else:
                    ctrl=nctrl
                c+=1
            except:
                features[x]=-1
```

EK 4: SVM, NB ve RF'nin Uygulanması

İlk olarak ön işleme tabi tutulmuş eğitim ve test veri kümeleri yüklenir.

```
In [ ]: tr=np.load('Training.npy',allow_pickle=True)
        ts=np.load('Testing.npy',allow_pickle=True)
```

Daha sonra her bir veri kümesi için sayım vektörleri oluşturulur.

```
In [ ]: count_vect = CountVectorizer()
        etr = count_vect.fit_transform(tr[:,1])

In [ ]: ets=count_vect.transform(ts[:,1])
```

Eğitim veri kümesi kulanarak bir SVM sınıflandırıcı oluşturulur ve eğitilir, daha sonra performansını ölçümleyebilmek için test veri kümesi kullanılarak tahminler gerçek etiketleriyle kıyaslanır.

```
In [ ]: svm=SVC(kernel='linear',random_state=101)

In [ ]: svm.fit(xtr,ytr)

In [ ]: p=svm.predict(xts)

In [ ]: metrics.confusion_matrix(yts,p)

In [ ]: metrics.accuracy_score(yts,p)

In [ ]: print(metrics.classification_report(yts,p))
```

Benzer şekilde aynı prosedür uygulanarak RF ve NB sınıflandırıcıları yürütülür, ancal aşağıda gösterildiği gibi gerekli olan sınıflandırıcı kullanılır.

```
In [ ]: rf=RandomForestClassifier(random_state=101)

In [ ]: nb=MultinomialNB()
```

EK 5: Öznitelik Belirleme Yönteminden Önce Kullanılan Yapay Sinir Ağları

Word2Vec modelindeki her bir kelime 300 değerle temsil edildiği ve en uzun tweet'te 32 kelime bulunduğu için, verikümesindeki her bir tweet 32×300'lük bir dizine dönüştürülür. Daha kısa olan tweetlerin boşlukları aşağıda gösterildiği gibi -1 ile doldurulur.

```
In [ ]: vts=full((dts.shape[0],32,300),-1.0)
        for x in pb(range(dts.shape[0])):
            tx=dts[x,1].split(' ')
            c=0
            for y in range(32-len(tx),32):
                try:
                    vts[x,y,:]=desc[argwhere(array(words)==tx[c])].reshape(-1)
                except:
                    print(tx[c])
                    pass
                c+=1
```

Her bir tweet'in boyutuna göre, CNN, LSTM ve GRU modelleri, uygun girdi formatına göre uygulanır. Yansız bir değerlendirme ve kıyalama yapabilmek için Sapmasız Yapay Sinir Ağlarının farklı türleri için benzer topolojiler uygulanır.

```
In [ ]: model=Sequential()
        model.add(Conv2D(128,(1,300),input_shape=(32,300,1),activation='relu'))
        model.add(Conv2D(64,(3,1),activation='relu'))
        model.add(Conv2D(64,(3,1),activation='relu'))
        model.add(MaxPool2D((2,1)))
        model.add(Conv2D(32,(2,1),activation='relu'))
        model.add(MaxPool2D((2,1)))
        model.add(Conv2D(32,(2,1),activation='relu'))
        model.add(Flatten())
        model.add(Dense(64,activation='relu'))
        model.add(Dropout(0.2))
        model.add(Dense(16,activation='relu'))
        model.add(Dense(1,activation='sigmoid'))
        model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
```

EK 5: Öznitelik Belirleme Yönteminden Önce Kullanılan Yapay Sinir Ağları

```
In [ ]: model=Sequential()
        model.add(CuDNNLSTM(128,input_shape=(32,300),return_sequences=True))
        model.add(CuDNNLSTM(64,return_sequences=True))
        model.add(CuDNNLSTM(64,return_sequences=True))
        model.add(CuDNNLSTM(32,return_sequences=True))
        model.add(CuDNNLSTM(32))
        model.add(Dense(64,activation='relu'))
        model.add(Dropout(0.2))
        model.add(Dense(16,activation='relu'))
        model.add(Dense(1,activation='sigmoid'))
        model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
```

```
In [ ]: model=Sequential()
        model.add(CuDNNGRU(128,input_shape=(32,300),return_sequences=True))
        model.add(CuDNNGRU(64,return_sequences=True))
        model.add(CuDNNGRU(64,return_sequences=True))
        model.add(CuDNNGRU(32,return_sequences=True))
        model.add(CuDNNGRU(32))
        model.add(Dense(64,activation='relu'))
        model.add(Dropout(0.2))
        model.add(Dense(16,activation='relu'))
        model.add(Dense(1,activation='sigmoid'))
        model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
```

EK 6: Öznitelik Belirleme Yönteminden Sonra Kullanılan Yapay Sinir Ağları

Önerilen öznitelik belirleme yöntemi uygulandıktan sonra en uzun tweet'in uzunluğu 27'ye düşmüştür. Dolayısıyla her bir tweet için oluşturulan dizin aşağıda gösterildiği gibi 27×300 boyutlarına düşmüştür.

```
In [ ]: vts=full((dts.shape[0],27,300),-1.0)
        for x in pb(range(dts.shape[0])):
            tx=dts[x,1].split(' ')
            c=0
            for y in range(27-len(tx),27):
                try:
                    vts[x,y,:]=desc[argwhere(array(words)==tx[c])].reshape(-1)
                except:
                    print(tx[c])
                    pass
                c+=1
```

Her bir tweet'in boyutuna göre, CNN, LSTM ve GRU modelleri, uygun girdi formatına göre uygulanır. Yansız bir değerlendirme ve kıyalama yapabilmek için Sapmasız Yapay Sinir Ağlarının farklı türleri için benzer topolojiler uygulanır.

```
In [ ]: model=Sequential()
        model.add(Conv2D(128,(1,300),input_shape=(27,300,1),activation='relu'))
        model.add(Conv2D(64,(3,1),activation='relu'))
        model.add(Conv2D(64,(3,1),activation='relu'))
        model.add(MaxPool2D((2,1)))
        model.add(Conv2D(32,(2,1),activation='relu'))
        model.add(MaxPool2D((2,1)))
        model.add(Conv2D(32,(2,1),activation='relu'))
        model.add(Flatten())
        model.add(Dense(64,activation='relu'))
        model.add(Dropout(0.2))
        model.add(Dense(16,activation='relu'))
        model.add(Dense(1,activation='sigmoid'))
        model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
```

```
In [ ]: model=Sequential()
        model.add(CuDNNLSTM(128,input_shape=(27,300),return_sequences=True))
        model.add(CuDNNLSTM(64,return_sequences=True))
        model.add(CuDNNLSTM(64,return_sequences=True))
        model.add(CuDNNLSTM(32,return_sequences=True))
        model.add(CuDNNLSTM(32))
        model.add(Dense(64,activation='relu'))
        model.add(Dropout(0.2))
        model.add(Dense(16,activation='relu'))
        model.add(Dense(1,activation='sigmoid'))
        model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
```

ÖZGEÇMİŞ

Adı ve Soyadı : Mohamed Guma Ibrahım BODEA
Doğum Tarihi : 06.05.1980
Medeni Hali : Evli
Dil : Arapça, İngilizce
E-posta : mohamedbodea1980@gmail.com

Öğrenim Geçmişi

Lise : Shohadaa Al shuairef, (1998).
Lisans : Bilgisayar Mühendisliği, Azzaytuna Üniversitesi, (2010).

Mesleki Deneyim

İş Yeri : Sağlık Bakanlığı, Libya, 2001-Devam Ediyor