
GAZİ ÜNİVERSİTESİ
Gazi
1926

DERİN ÖĞRENME KULLANARAK BÜYÜK BOYUTLU DOKÜMANLARDA İÇERİK TABANLI BENZERLİK İLE KÜMELEME

Kevser ÖZDEM

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2021

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Kevser ÖZDEM

07/06/2021

DERİN ÖĞRENME KULLANARAK BÜYÜK BOYUTLU DOKÜMANLARDA İÇERİK TABANLI BENZERLİK İLE KÜMELEME

(Yüksek Lisans Tezi)

Kevser ÖZDEM

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2021

ÖZET

Günümüzde veri boyutu büyük bir hızla artmaktadır. Çok büyük boyuttaki veri üzerinde günümüz teknolojisiyle bile kısa sürede işlem yapmak mümkün olmamaktadır. Bu yüzden, çok sayıdaki büyük boyutlu dokümanı az sayıda birbiriyle ilişkili ve anlamlı küme halinde düzenleme gerektiren kümeleme önemli bir araştırma konusu haline gelmiştir. Son yıllarda birçok alanda başarılıyla uygulanan derin öğrenme yöntemleri denetimsiz öğrenme uygulamalarında da başarılı bir şekilde kullanılabilmektedir. Bu çalışmada, büyük boyutlu dokümanlarda içerik benzerliğine göre kümeleme için derin öğrenme tabanlı bir model geliştirilmiştir. Geliştirilen derin öğrenme modelinde CNN ve LSTM ağları birlikte kullanılmıştır. Geliştirilen modeli test etmek için 386 adet İngilizce ders kitabından oluşan toplam 7,61 GB boyutundaki bir veri kümesi kullanılmıştır. Deneysel çalışmalarda ortalama doğruluğu %66 olan 18 farklı küme elde edilmiştir. Deneysel sonuçlar, geliştirilen model ile elde edilen kümelerin, literatürde yaygın olarak kullanılmakta olan k-means ve CURE kümeleme algoritmalarına göre daha yüksek başarıya sahip olduklarını göstermiştir. Geliştirilen model ile oluşturulan kümeler, 0,65 NMI ve 0,59 AMI değerlerine sahiptir. Ayrıca, Silhouette ve Davies-Bouldin iç değerlendirme ölçütlerinde de sırasıyla 0,81 ve 0,95 değerleri elde edilmiştir.

Bilim Kodu	:	92432
Anahtar Kelimeler	:	Büyük boyutlu doküman kümeleme, derin öğrenme, CNN, LSTM, hash fonksiyonu
Sayfa Adedi	:	67
Danışman	:	Prof. Dr. M. Ali AKCAYOL

DEEP LEARNING BASED LARGE-SCALE DOCUMENT CLUSTERING WITH CONTENT-BASED SIMILARITY

(M. Sc. Thesis)

Kevser ÖZDEM

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

June 2021

ABSTRACT

Nowadays, the size of data is increasing rapidly. It is not possible to process very large data in a short time even with today's technology. Hence, clustering, which requires organizing large numbers of large-scale documents into small numbers of interrelated and meaningful clusters, has become an important research topic. Deep learning methods, which have been successfully applied in many fields in recent years, can also be used successfully in unsupervised learning applications. In this study, a deep learning-based model has been developed for clustering based on content similarity in large-scale documents. CNN and LSTM networks have been used together in the developed deep learning model. A data set consisting of 386 English textbooks with a total size of 7.61 GB has been used to test the developed model. In experimental studies, 18 different clusters with an average accuracy of 66% have been obtained. The experimental results have shown that the clusters obtained with the developed model had higher success than the k-means and CURE clustering algorithms, which are widely used in the literature. The clusters created with the developed model have values of 0.65 NMI and 0.59 AMI. In addition, values of 0.81 and 0.95 have been obtained for the internal evaluation metrics Silhouette and Davies-Bouldin, respectively.

Science Code	:	92432
Key Words	:	Large-scale document clustering, deep learning, CNN, LSTM, hash function
Page Number	:	67
Supervisor	:	Prof. Dr. M. Ali AKCAYOL

TEŞEKKÜR

Çalışmalarım süresince her daim bana yol gösteren, bilgi ve deneyimlerini aktaran kıymetli danışmanım Prof. Dr. M. Ali AKCAYOL'a ve beni yetiştirerek bugünlere gelmemi sağlayan, öğrenim hayatım boyunca beni daima destekleyen aileme teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
GB	Gigabayt
MB	Megabayt

Kısaltmalar	**Açıklamalar**
AMI	Adjusted Mutual Information
ANN	Artificial Neural Network
ARI	Adjusted Rand Index
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
BRNN	Bidirectional Recurrent Neural Network
CFS	Clustering algorithm by Fast finding and Searching of density peaks
CLARANS	Clustering Large Applications based on RANdomized Search
CNN	Convolutional Neural Network
CURE	Clustering Using REpresentatives
DBN	Deep Belief Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DNN	Deep Neural Network
ELMD	Exemplar-based Low-rank sparse Matrix Decomposition
FCM	Fuzzy C-Means
GRNN	Gated Recurrent Neural Network
HAL	Hyperspace Analogue to Language

Kısaltmalar	Açıklamalar
HCA	Hierarchical Clustering Algorithms
HS	Harmony Search
IMDB	Internet Movie Database
LCS	Longest Common Subsequence
LSA	Latent Semantic Analysis
LSTM	Long Short Term Memory
MEDLINE	Medical Literature Analysis and Retrieval System
MI	Mutual Information
MLP	Multi Layer Perceptron
NLTK	Natural Language ToolKit
NMI	Normalized Mutual Information
OPTICS	Ordering Points To Identify the Clustering Structure
ReLU	Rectifier Linear Unit
RI	Rand Index
ROCK	RObust Clustering using linKs
RNN	Recurrent Neural Network
SOM	Self Organizing Maps
TF-IDF	Term Frequency and Inversed Document Frequency

1. GİRİŞ

Kümeleme, çok sayıda dokümanı az sayıda anlamlı ve tutarlı kümeler halinde düzenleyerek, bilgilendirici tarama mekanizmaları için bir temel bilgi sağlamaktadır. Günümüzde veri boyutu her geçen gün daha hızlı bir şekilde artmaya devam etmektedir. Bu durum göz önünde bulundurulduğunda birçok alanda olduğu gibi doküman kümelemede de büyük boyutlu dokümanlarda işlem yapmak oldukça önemli bir konu haline gelmiştir.

Kümeleme işleminin gerçekleştirilebilmesi için öncelikle benzer dokümanların tespit edilmesi gerekmektedir. Benzer dokümanların belirlenebilmesi için benzerlik ölçütlerinden yararlanılmaktadır. Ancak her tür kümeleme problemi için evrensel olarak belirlenmiş bir en iyi benzerlik ölçütü bulunmamaktadır. Bu nedenle, üzerinde çalışılan kümeleme problemine uygun benzerlik ölçütünü seçmek önemlidir.

İnternet kullanımının yaygınlaşması ile veri çeşitliliği her geçen gün hızla artmaktadır. Bu büyük miktardaki veriden anlamlı ve önemli bilginin kısa sürede çıkarılması önemli bir konu haline gelmiştir. Büyük boyutlu ve çok sayıdaki veriden önemli bilginin çıkarılmasında başarıyla uygulanan yapay zeka (artificial intelligence), makine öğrenmesi (machine learning) ve derin öğrenme (deep learning) gibi kavramlar oldukça önemli hale gelmiştir. Günümüzde derin öğrenme yöntemlerinin farklı alanlarda başarıyla uygulanmasından ötürü doküman kümeleme problemlerinde de kullanılmaya başlanmıştır [1, 2, 3]. Model tabanlı kümeleme yöntemleri, matematiksel modellerle verilerin uygunluğunu bulmakta ve optimize etmektedir. Modele dayalı kümeleme yöntemleri, geleneksel kümelemeye benzer olarak her küme için özellik bilgilerini de ortaya çıkarmaktadır [4]. Ancak, literatürde derin öğrenme yöntemleri kullanılarak geliştirilen kümeleme algoritmalarının çok az bir kısmı büyük boyutlu dokümanlarda çalışmak için uygundur.

Luo ve arkadaşları tarafından 2007 yılında yapılan çalışmada, büyük boyutlu doküman kümelemek amacıyla dinamik bir Self Organizing Maps (SOM) algoritması önerilmiştir [5]. Bu algoritmada dinamik olarak uygun sayıda nöron eklenerek daha sonra etki düzeyi düşük nöronlar çıkartılmıştır. Böylece nöron sayısı kademeli olarak azaltılarak optimizasyon yapılmıştır. Geliştirilen modelde yoğun kullanılan ve etki düzeyi yüksek nöronlar sadece ilgili dokümanlar üzerinde eğitilmiştir. Böylece kümeleme kalitesi geliştirilmiştir.

Gerçekleştirilen deneysel çalışmalarda, önerilen algoritmanın diğer SOM türevlerine göre hem kümeleme başarısı hem de hız açısından daha iyi sonuçlar verdiği görülmüştür.

Feng ve arkadaşları tarafından 2010 yılında yapılan çalışmada, büyük boyutlu metin kümeleme için uygulanabilecek yeni dinamik ve uyarlanabilir bir SOM algoritması önerilmiştir [6]. Bu algoritmanın dinamik bir ağın özelliklerine ve uyarlanabilir öğrenme hızına sahip bir model olduğu belirtilmiştir. Düşük boyutlu metin kümeleme uygulamaları için önemli avantajlara sahip olan SOM yardımıyla yüksek boyutlu giriş verileri, düşük boyutlu bir özellik haritasına eşlenmiştir. Ağdaki ağırlık vektörleri, giriş alanından rastgele metin vektörleri seçilerek başlatılmaktadır. Öğrenme aşamasında, en iyi eşleşen (kazanan) düğümü bulmak amacıyla, giriş vektörünün ağırlık vektörleri ile kosinüs benzerlikleri hesaplanarak en büyük olan seçilmektedir. Benzerlik belirli bir eşik değerden daha küçük olduğunda, ağa yeni bir düğüm eklenerek giriş modeli yeni düğümün ağırlık vektörüne atanmaktadır. Ağ kararlı hale geldikten sonra benzer sınıfları temsil eden benzer düğümler optimizasyon için birleştirilmektedir. Deneysel sonuçlar, önerilen algoritmanın sinir ağının metin kümeleme ve öğrenme hızında başarılı olduğunu göstermiştir.

Wang ve arkadaşları tarafından 2011 yılında yayımlanan çalışmada, büyük ölçekli verileri kümelemek için Exemplar-based Low-rank sparse Matrix Decomposition (ELMD) isimli bir model önerilmiştir [7]. Matris ayrışmasına dayalı kümeleme ve düşük-rank matris yaklaşımını birleştiren bu yöntemin mevcut kümeleme yöntemlerine göre gürültüye karşı sağlamlık, matris yaklaşımıyla özellik seçimi, etkili hesaplama gibi avantajlara sahip olduğu belirtilmiştir. Deneysel çalışmalarda, ELMD'nin hem sentetik dokümanlar hem de gerçek dünya dokümanları üzerinde birçok kümeleme yönteminden daha iyi performans gösterdiği belirtilmiştir.

Xiufeng ve arkadaşları, 2011 yılında yaptıkları çalışmada literatürde metin kümeleme algoritmasının çok fazla olmasına rağmen çoğunlukla sınırlı veri kümesini hedeflediklerini belirtmişlerdir [8]. Çalışmalarında, iyileştirilmiş Clustering Using REpresentatives (CURE) algoritmasına dayanan büyük ölçekli doküman kümeleme algoritması önerilmiştir. CURE algoritmasının rastgele örnekleme ve segmentasyona dayanan bir yöntem olduğu belirtilmiştir. Önerilen yöntemde, büyük ölçekli veri kümesi bölünerek kümeleme işlemi tekrarlı bir şekilde gerçekleştirilmektedir. 30260 adet veri üzerinde yapılan testlerde kümeleme başarısı olarak 0,81 duyarlılık (recall) ve 0,83 kesinlik (precision) değerleri elde

edilmiştir. Çalışmada, iyileştirilmiş CURE algoritmasının kümeleme hassasiyetini sağlayabildiği ve gerçek dünya problemlerinde uygulanabilir olduğu belirtilmiştir.

Spasojevic ve arkadaşları tarafından 2011 yılında yapılan çalışmada, kitapları içerik benzerliğine dayalı olarak kümelemek için iki aşamalı bir yöntem önerilmiştir [9]. İlk olarak, hash teknikleri ve eşikler kullanılarak benzer kitaplar ve benzer sayfalar tespit edilmektedir. Sayfa ve kitaplardan özellikler çıkarmak için min-hashing yöntemine dayanan bir mekanizma tanımlanmıştır. Daha sonra, eşleşen kitaplar arasındaki tam ilişkiyi tanımlamak amacıyla benzer kitaplar arasında çeşitli sinyaller kullanılmıştır. Bu sinyallerin, izole edildiklerinde iyi bir hassasiyet elde ettiği belirtilmiş, ancak manuel olarak seçilecek mantıklı kombinasyonların kümeleme başarısını önemli ölçüde iyileştirebileceği ifade edilmiştir.

Young ve arkadaşları tarafından 2012 yılında yapılan bir çalışmada, tekrarlayan sinir ağından esinlenilerek geçici düzenlilikleri denetimsiz bir şekilde tanımlayabilen tekrarlayan bir kümeleme algoritması önerilmiştir [10]. Her durumun bir önceki duruma bağlı olduğu bir problem, artımlı bir kümeleme algoritmasına ihtiyaç duymaktadır. Bu nedenle, girişin yoğun bölgelerini temsil eden küme merkezlerini öğrenmenin yanı sıra, algoritma mevcut merkezlere göre her yeni gözlemi kodlayarak bir durum değeri sağlamalıdır. Bu durum değeri durumu kümeleme algoritmasının sonraki girişine eklenmektedir. Önerilen artımlı kümeleme algoritmasında, mevcut gözlem ve önceki durum değeri birleştirilerek merkezler oluşturulmaktadır. Önceki durum değeri, mevcut giriş vektörünün merkezlerin her birine ait olma olasılığının bir ölçüsü olarak tanımlanmıştır. Deneysel sonuçlara göre, önerilen yaklaşımın her uzunlukta zamansal nitelikleri karakterize edebildiği ve derin öğrenme mimarilerine uygulanarak özellik vektörleri üretebileceği belirtilmiştir.

Forsati ve arkadaşları tarafından 2013 yılında gerçekleştirilen çalışmada, Harmony Search (HS) optimizasyon yöntemine dayalı doküman kümeleme algoritmaları önerilmiştir [11]. Kümeleme işlemi bir optimizasyon problemi olarak modellenmiştir. Önerilen algoritmalarda, HS ve k-means algoritmaları birleştirilerek tüm çözüm uzayında küreselleştirilmiş bir arama gerçekleştirilmektedir. Ayrıca, orijinal k-means algoritmasında rastgele seçilen başlangıç parametrelerine olan bağımlılık azaltılarak daha kararlı bir yapı elde edilmiştir. Beş farklı veri kümesi üzerinde yapılan deneysel çalışmalarda, önerilen algoritmaların k-means ve genetik algoritmalara kıyasla daha iyi kümeler bulabildiği ve

küme kalitelerinin f-skor, entropi ve küme merkezlerinin ortalama uzaklığı gibi ölçütlerle karşılaştırıldığı belirtilmiştir.

Chen tarafından 2015 yılında yapılan bir çalışmada, kümelemenin performansının büyük ölçüde veri sunumuna bağlı olduğu belirtilerek parametrik olmayan kümeleme ve Derin İnanç Ağı (Deep Belief Network - DBN) önerilmiştir [1]. Model, özellik gösterimi ve boyut küçültme için gözetimsiz derin öğrenmeden yararlanarak parametrik olmayan kümeleme gerçekleştirmektedir. Önerilen modelin, veri boyutundaki değişimlere göre model boyutunu otomatik olarak uyarlayabilen ayrımcı bir kümeleme yöntemi olduğu belirtilmiştir.

Yang ve arkadaşları tarafından 2015 yılında yapılan bir çalışmaya göre, son zamanlarda popüler olan derin öğrenme gözetimsiz öğrenme üzerinde başarılı olmuştur [12]. Çalışmada, DBN ve Fuzzy C-Means (FCM) algoritmasını birleştiren yeni bir kümeleme modeli önerilmiştir. Çalışmaya göre derin mimari büyük veri kümesinden çok daha kolay öğrenebilmektedir. Bu nedenle, düşük boyutlu veri kümesi için, giriş verileri yalnızca düşük boyutlu uzaya değil, aynı zamanda yüksek boyutlu uzaya da eşlenmektedir. Böylece, sorunu çözmek için hem DBN hem de FCM'nin avantajlarından yararlanılmaktadır. Deneysel sonuçlar, önerilen modele ait performansın çoğu test veri kümesinde standart k-means ve FCM' den daha iyi olduğunu göstermektedir.

Tang ve arkadaşları tarafından 2015 yılında yapılan çalışmada, aşağıdan yukarıya biçiminde vektör tabanlı doküman öğrenimi için bir sinir ağı modeli önerilmiştir [13]. İlk olarak model, Evrişimli Sinir Ağı (Convolutional Neural Network - CNN) ve Uzun Kısa Süreli Bellek (Long Short Term Memory - LSTM) ile cümle gösterimini öğrenmektedir. Daha sonra, cümlenin anlamı ve ilişkileri Geçitli Tekrarlayan Sinir Ağı (Gated Recurrent Neural Network – GRNN) ile çıkarılmaktadır. Dört büyük ölçekli veri kümesinde yapılan deneysel sonuçlarda, modelin çeşitli modern algoritmalara göre üstün performans sergilediği ve GRNN'nin, duygu sınıflandırması için doküman modellemede standart RNN'e göre çok daha iyi sonuçlar verdiği ve LSTM'in CNN'e göre daha başarılı olduğu belirtilmiştir.

Karaa ve arkadaşları tarafından 2016 yılında yayımlanmış bir çalışmada, genetik algoritmaya dayalı yeni bir kümeleme tekniği önerilmiştir [14]. Önerilen algoritma, metinsel verilerle çalışmak için vektör uzayı modeli kullanılmıştır. Genetik algoritmanın uygunluk işlevini hesaplamak amacıyla yeni bir model tanımlanmıştır. Dokümanlar, vektör uzayı

modeliyle modelledikten sonra, her terim sayısal ağırlıklı veri olarak anlamsal yönleri dikkate alınmadan işlenmektedir. Deneysel çalışmalarda, Medical Literature Analysis and Retrieval System (MEDLINE)'da bulunan 500 özetten oluşan bir veri kümesi kullanılmıştır. Ancak, önerilen yöntemin herhangi bir metin veri kümesi veya elektronik doküman üzerinde uygulanabileceği belirtilmiştir.

Saiyad ve arkadaşları tarafından 2016 yılında yapılan çalışmada, geleneksel metin kümeleme yöntemlerinde görülen problemler ve anlamsal kümeleme yaklaşımı ele alınmıştır [15]. Semantik yaklaşımın, kümelemenin doğruluğu ve kalitesi açısından geleneksel yaklaşımdan daha iyi olduğu sonucuna varılmıştır. Çalışma kapsamında yapılan literatür taramasında en yaygın ve popüler kümeleme algoritmasının k-means algoritması ve türevleri olduğu belirtilmiştir. Doküman kümelemede en çok karşılaşılan sorunların ise eş anlamlılık, çok anlamlılık, yüksek boyutluluk ve küme etiketleme olduğu ifade edilmiştir. Ayrıca sözcük zinciri tabanlı ve ontoloji tabanlı tekniklerin kombinasyonunun kümelemede karşılaşılan problemlerin çözülmesine yardımcı olabileceği belirtilmiştir.

Kulkarni ve arkadaşları tarafından 2016 yılında yapılan bir çalışmada, gözetimsiz derin öğrenme tabanlı bir kelime kümeleme yaklaşımı önerilmiştir [16]. Yaklaşım, altı katmanlı bir CNN mimarisi içermektedir. Kenar duyarlı eğitilmemiş bir CNN özellik çıkarıcı olarak kullanılmıştır. Her kelimeden bir özellik vektörü çıkarılarak kümeleme için iki aşamalı bir yaklaşım izlenmiştir. İlk aşamada, oldukça benzer olan kelime görüntüleri grafiğe bağlı bileşen kullanılarak kümelenmiştir. İkinci aşamada ise, başlangıçta kümelenmemiş olan kelimeler oluşturulmuş kümelerle karşılaştırılmıştır. Yaklaşım benzer şekil kalıplarını kelime düzeyinde tespit etmektedir. Deneysel sonuçların, derin öğrenme özellik vektörlerinin ayırt edici gücünü gösterdiği belirtilmiştir. Ayrıca, modelin sözcük segmentasyon hatalarına karşı başarılı olduğu ve büyük veri kümelerinde ölçeklenebilir olduğu vurgulanmıştır.

Saxena ve arkadaşları tarafından 2017 yılında yapılmış olan çalışmaya göre, karar ağaçları ve sinir ağları kümelemede en sık kullanılan iki yöntemdir [4]. Karar ağaçlarında veri, her yaprağın bir kavramı ifade ederek bu kavramın olasılıklı bir tanımını ifade ettiği hiyerarşik bir ağaç olarak modellenmektedir. Kümelemede ağaçlardan yararlanan pek çok algoritma bulunmakla birlikte en iyi bilineni CobWeb (örümcek ağı)'dir. Bu algoritmada her kavram bir nesne kümesini tanımlamaktadır ve her nesne ikili sistemde özellik listesi olarak

tanımlanmıştır. Algoritma yüksek tahmin oranı amaçlamaktadır ancak büyük ölçekli veri kümelerinde başarılı sonuçlar vermemektedir.

Çalışmaya göre, kümelemede kullanılan bir diğer yaklaşım olan sinir ağı modelinde her küme prototip nöronlara bağlı bir nöron ile belirtilmektedir. Öğrenmeden önce her bağlantı rastgele bir ağırlık ile ilişkilendirilmektedir. SOM algoritması, kümeleme için kullanılan sinirsel algoritmalar arasında en popüler olanlardandır. Algoritma, tek katmanlı bir ağ oluşturmaktadır. Öğrenme süreci prototip nöronların yarışması ile gerçekleşmektedir. Ağırlık vektörü geçerli duruma en yakın olan nöron kazanmaktadır. Bu algoritma çoğunlukla özellik çıkarımı ve kümeleme analizinde veri görselleştirme çalışmalarında kullanılmaktadır.

Jahromi ve arkadaşları tarafından 2017 yılında yapılan bir çalışmada, daha iyi kümeleme performansı için derin öğrenme tabanlı bir model önerilmiştir [17]. Modelde, aynı aktivasyon fonksiyonuna sahip ancak farklı gizli nöron sayısına sahip iki katmanlı bir otomatik kodlayıcı kullanılmıştır. Veriler, kümeleme için vektör şeklinde birleştirilmiştir. Kümeleme işlemi k-means algoritması ile tamamlanmıştır. Deneysel sonuçlar, modelin orijinal k-means kümeleme sonuçlarından başarılı olduğunu göstermiştir.

Yang ve arkadaşları tarafından 2017 yılında yapılan bir çalışmaya göre, çoğu öğrenme yaklaşımı boyutsal küçültme ve kümelemeyi ayrı ayrı ele almaktadır [18]. Ancak ikisini birlikte optimize etmek her ikisinin de performansını önemli ölçüde artırmaktadır. Çalışmada, verileri daha iyi kümelemek için Derin Sinir Ağı (Deep Neural Network - DNN) ve k-means algoritmalarının kullanıldığı bir kümeleme yaklaşımı önerilmiştir. Çalışmadaki amaç, yüksek boyutlu verileri otomatik olarak k-means ile kümeleme için uygun olan gizli bir alana eşlemektir. Sentetik ve gerçek veriler ile yapılan deneylerde, önerilen modelin başarılı olduğu gösterilmiştir.

Sreedhar ve arkadaşları tarafından 2017 yılında yapılmış bir çalışmada, MapReduce kullanılarak büyük verilerin kümelenmesi için iki yaklaşım sunulmuştur [19]. Birinci yaklaşımda standart k-means algoritması kullanılarak MapReduce uygulamasına odaklanılmıştır. İkinci yaklaşımda ise, büyük veri kümelerinde maksimum küme içi ve minimum kümeler arası mesafelere sahip kümeler üreterek kümeleme kalitesinin artırılması amaçlanmıştır. Çalışmada deneysel sonuçların üretilmesi için üç farklı veri kümesi

kullanılmıştır. Bunlardan biri ses verilerinden, ikincisi ise hava tahminlerinden oluşmaktadır. Üçüncü veri kümesi olarak, Project Gutenberg'deki ücretsiz 3.000 İngilizce e-kitaptan oluşan bir altküme kullanılmıştır. Farklı sayıdaki doküman setleri üzerinde yapılan deneylerde, doküman sayısı 200'den 1000'e doğru arttıkça, standart k-means algoritmasının çalışma zamanının 60 dakikaya kadar çıktığı, önerilen algoritmanın ise kümeleme işlemini 5 dakikanın altında bir sürede tamamladığı gösterilmiştir.

Mei ve arkadaşları tarafından 2018 yılında yapılan çalışmada, dokümanları tutarlı kategoriler halinde gruplamak amacıyla bulanık küme teorisi kullanılarak doküman kümeleri arasındaki çakışmayı belirlemek için bir yöntem önerilmiştir [20]. Dokümanlar, veri kümesi büyük ve çok yüksek boyutlara sahip olduğunda, tamamen belleğe sığamayan yüksek boyutlu vektörlerden oluşan bir koleksiyon olarak temsil edilmektedir. Ancak, mevcut bulanık kümeleme yaklaşımlarının çoğunun küçük ve statik veri kümeleriyle ilgili olduğu belirtilmiştir. Çalışmada, büyük boyutlu verilerde bulanık kümeleme ele alınmıştır. Dokümana özel bulanık kümelemeyi büyük ölçekli problemlerde etkili hale getirecek bir yöntem önerilmiştir. Büyük doküman veri kümeleri ile yapılan deneysel çalışmalarda, önerilen yaklaşımın doküman kümelemede mevcut yaklaşımlardan daha yüksek başarı sağladığı gösterilmiştir.

Sardar ve arkadaşları tarafından 2018 yılında yapılan çalışmaya göre, büyük boyutlu verileri kümeleme işlemlerinde Hadoop dağıtık platform, MapReduce ve değiştirilmiş kümeleme algoritmaları, kümeleme işini birden çok düğüm arasında dağıtarak hesaplama süresini kısaltmak için kullanılmaktadır [21]. Çalışmada, MapReduce kullanılarak bölünebilir kümeleme algoritmaları ile ilgili son araştırmalar incelenmiştir. Birçok çalışmada, hesaplama süresini azaltmak için farklı veri kümelerinde k-means, k-prototypes, k-medoids, k-modes ve FCM gibi geleneksel kümeleme algoritmalarının MapReduce için değiştirilerek kullanıldığı belirtilmiştir.

Lv ve arkadaşları tarafından 2018 yılında yapılan bir çalışmaya göre, yüksek hacimli metinsel mahkeme dokümanları hakimin makul kararlar almasında büyük bir zorluk oluşturmaktadır [22]. Çalışmada, yardımcı vakaları geniş vakalardan ayırmak amacıyla mahkeme metnini kümeleyen derin bir yoğunluk zirvelerini arama ve hızlı bulma ile kümeleme (clustering algorithm by fast finding and searching of density peaks - CFS) modeli önerilmiştir. Önerilen modelde, metnin özelliklerini öğrenmek için bir dizi derin

öğrenme modeli kullanılarak özellik çıkarıcı tasarlanmıştır. Her bir nöron, her eğitim aşamasında farklı veriler üzerinde eğitilmiş farklı mimariler ile rastgele düşürülmüştür. Metin özellikleri elde edildikten sonra, CFS algoritması metni karşılık gelen kategorilere kümelemek için kullanılmıştır. Algoritmada, daha düşük lokal yoğunluğa sahip olan sınırla çevrelenmiş ve küme merkezi diğer küme merkezlerinden uzak kümeler elde edilmiştir. Algoritmanın, küresel olmayan kümeleri keşfederek metin nesnelerini benzerlik temelinde farklı sınıflara grupladığı belirtilmiştir. Çalışmada, önerilen modelin kümeleme doğruluğu açısından CFS algoritmasından daha iyi performans gösterdiği ve büyük metin verilerini kümeleme potansiyelinin olduğu belirtilmiştir.

Subramani ve arkadaşları tarafından 2018 yılında yapılan bir çalışmaya göre, konu modelleme bir doküman koleksiyonundaki ortak konuları keşfetmek için kullanılan bir metin madenciliği tekniğidir [2]. Sinir ağı modellerinin en büyük avantajı, büyük derlemler için ölçeklenebilir olmalarıdır. Bu modeller, büyük boyutlu derlemi daha küçük parçalar haline getirerek hafızaya sığması ve hesaplanması kolay olan toplu model eğitimine izin vermektedir. Modelde, ayrı ayrı gruplar üzerinde kademeli olarak güncelleme yapılmaktadır. Bu yaklaşım, tüm verinin hafızada tek bir nesne olarak depolanmasını ve hesaplanmasını, dolayısıyla büyük veri için uygun olmasını sağlamaktadır. Ayrıca, sinir ağı modelleri, kelime sayısı yerine kelime bağlamı yoluyla anlam çıkarabilmektedir. Çalışmada konular, dokümanlar ve kelimeler arasındaki olasılık ilişkilerini modelleyerek benzer dokümanların kümelenmesi için yapay sinir ağları kullanılmıştır. Önerilen modelde, derlem içindeki kelimeler arasındaki anlamsal ilişkileri çıkarmak için katmanlar tasarlanarak her doküman için bir konu olasılık dağılımı sunulmuştur. Daha sonra, benzer dağılımları olan kosinüs benzerliğinde eşik değeri karşılayan dokümanlar bir araya getirilmiştir. Keras kullanılarak gerçekleştirilen modelin, Internet Movie Database (IMDB) veri kümesinde geleneksel ve yaygın olarak kullanılan kümeleme modelleri ile karşılaştırıldığında veri işleme kapasitesi ve ölçeklenebilirlik açısından öne çıktığı belirtilmiştir.

Janani ve arkadaşları tarafından 2019 yılında gerçekleştirilen çalışmada, büyük boyutlu dokümanları kümelemek amacıyla spektral kümelemeyi sürü optimizasyonu ile birleştiren bir algoritma önerilmiştir [23]. Önerilen algoritma, k-means ve standart sürü optimizasyon algoritmaları ile üç farklı veri kümesi üzerinde karşılaştırılmıştır. Deneysel sonuçlara göre önerilen algoritma, karşılaştırılan diğer algoritmalardan %6-7 oranında daha yüksek kümeleme doğruluğu sağlamıştır.

Curiskis ve arkadaşları tarafından 2020 yılında yapılmış çalışmada, Twitter ve Reddit platformlarından elde edilmiş üç veri kümesi üzerinde doküman kümeleme ve konu modelleme işlemi gerçekleştirilmiştir [24]. Dört farklı özellik temsil yöntemi ve dört farklı kümeleme algoritması kullanılarak sonuçlar karşılaştırılmıştır. Deneysel sonuçlar, çevrimiçi sosyal ağ verilerine ait kelime gömme temsillerinin doküman kümeleme için etkili kullanılabileceğini göstermiştir. Çalışmada en yüksek performans 0,49 Normalised Mutual Information (NMI) ve 0,49 Adjusted Mutual Information (AMI) değerleri ile, doc2vec temsilleri ve k-means kümeleme algoritması kullanılarak elde edilmiştir.

Sardar ve arkadaşları tarafından 2020 yılında yayımlanan çalışmada, doküman veri kümeleri için k-means kümeleme algoritmasının MapReduce tabanlı bir versiyonu önerilmiştir [25]. Çalışmada önerilen MapReduce tabanlı k-means algoritması optimum performans ölçüsünü belirlemek amacıyla 4 farklı boyutlu Hadoop kümesi ve 5 farklı boyutta oluşturulmuş doküman veri kümesi üzerinde denenmiştir. Deneysel sonuçlar, önerilen algoritmanın verimliliğinin, tüm farklı veri kümesi boyutları için geleneksel k-means algoritmasından daha iyi performans göstermiştir.

Tezin Literatüre Katkısı

Literatürde doküman kümeleme üzerine yapılmış pek çok çalışma bulunmaktadır. Ancak, bu çalışmaların büyük bir bölümü kısa metinler üzerine gerçekleştirilmiştir. Büyük boyutlu dokümanlar ile yapılmış çalışmalar ise çoğunlukla MapReduce gibi dağıtık yapılar kullanılarak k-means gibi klasik algoritmalar ile gerçekleştirilmiştir. Son zamanlarda derin öğrenmenin farklı alanlarda başarılı uygulanmasıyla doküman kümeleme alanında da derin ağlar kullanılmaya başlanmıştır. Ancak, çalışmalar yine kısa boyutlu metinler üzerinde yoğunlaşmıştır. Bu tez çalışmasında, büyük boyutlu dokümanları kümelemek için CNN ve LSTM tabanlı hibrit bir derin öğrenme modeli geliştirilmiştir.

Çalışmada, büyük boyutlu doküman olarak İngilizce ders kitaplarından oluşan bir veri kümesi kullanılmıştır. 386 adet doküman gerekli ön işlemlerden geçirildikten sonra geliştirilen derin öğrenme modeli ile özellik çıkarımı gerçekleştirilmiştir. Daha sonra bu özellikler kullanılarak dokümanlara hash fonksiyonu uygulanarak normalizasyon gerçekleştirilmiştir.

2. BENZERLİK ÖLÇÜTLERİ

Benzerlik ölçütleri, metin sınıflandırma, bilgi çıkarımı, doküman kümeleme, metin özetleme, başlık tespiti gibi uygulamalarda önemli bir rol oynamaktadır. Sözcükler arasındaki benzerlikleri bulmak, cümle ve doküman benzerliklerini bulmanın ilk aşamasıdır [26]. Benzerlik ölçütü, hedef nesnelerin yakınlık veya uzaklık derecesini yansıtmaktadır. Veride bulunan ve kümeleri ayırt edecek özellikleri ortaya çıkarmaktadır. Genel olarak benzerlik ölçütleri iki nesnenin arasındaki benzerliği, iki nesnenin özellikleri ve ölçütün kendisine bağlı olarak tek bir sayısal değere eşlemektedir. Farklı ölçütler farklı eşlemelere neden olmakla beraber, aynı kümeleme algoritması için farklı gereksinimlere de ihtiyaç olabilmektedir. Tüm kümeleme problemleri için evrensel olarak en iyi bir ölçüt bulunmamaktadır. Yapılacak kümeleme işlemi için uygun bir benzerlik ölçütü seçmek oldukça önemlidir [27].

Kelimeler, sözcüksel ve anlamsal olarak iki şekilde benzer olabilmektedir. Kelimeler benzer bir karakter dizisine sahiplerse sözcüksel benzer, aynı temaya sahipse anlamsal benzer olarak nitelendirilmektedir. Farklı bir temaya sahip, ancak aynı bağlamda kullanılan sözcükler ise birbirleriyle ilişkili değildir. Karakter dizisi (string) tabanlı algoritmalar sözcüksel benzerliği ölçmek için kullanılırken, derlem tabanlı ve bilgi tabanlı algoritmalar anlamsal benzerliği ölçmek için kullanılmaktadır [26].

2.1. String Tabanlı Algoritmalar

String tabanlı benzerlik ölçütleri karakter dizileri ve karakter dizilimi üzerinde gerçekleştirilmektedir. Longest Common Subsequence (LCS) algoritması, her iki string'de bulunan bitişik karakter zincirine dayanmaktadır. En uzun ortak alt sıra farklı yöntemlerle bulunabilmekte ve farklı algoritmalar tarafından da kullanılmaktadır.

Edit uzaklığı, iki string arasındaki uzaklığı bulmak için kullanılan ölçütlerden birisidir. Karşılaştırılan iki string'den birisini diğerine dönüştürmek için gerekli minimum sayıda karakter ekleme ve silme işlem sayısını göstermektedir [26]. Ancak, bu ölçütte x ve y string'leri arasında hesaplanan uzaklık, LCS'den faydalanılarak Eş. 2.1 kullanılarak da bulunabilmektedir.

$$d(x,y) = length(x) + length(y) - 2 * length(LCS) \quad (2.1)$$

Burada, x ve y iki string'i, $length()$ bir string'in uzunluğunu, LCS iki string arasındaki en uzun alt sırayı göstermektedir.

Levenshtein ölçütü de edit uzaklığına benzer olarak, bir string'i diğer string'e dönüştürmek için gereken minimum sayıda karakter ekleme, silme veya transpozisyon işlem sayısını göstermektedir [26]. x ve y string'leri arasındaki Levenshtein uzaklığı, Eş. 2.2'deki gibi hesaplanmaktadır.

$$lev_{x,y}(i,j) = \begin{cases} \max(i,j) & eğer \min(i,j) = 0, \\ \min \begin{cases} lev_{x,y}(i-1,j) + 1 \\ lev_{x,y}(i,j-1) + 1 \\ lev_{x,y}(i-1,j-1) + 1_{x_i \neq y_j} \end{cases} & diğer\ durumlar \end{cases} \quad (2.2)$$

Burada, x ve y iki string'i, i x string'ine ait karakter işaretçisini ve j y string'ine ait karakter işaretçisini göstermektedir.

Öklid uzaklığı, sayısal veriler için kullanılan en yaygın kullanılan uzaklık ölçütüdür. En yaygın kullanımı L_2-norm şeklindedir. İki vektör elemanı arasındaki farkların karelerinin toplamının karekökü alınarak hesaplanmaktadır. Eş. 2.3, x ve y vektörleri için bu Öklid uzaklığını göstermektedir. Kümelemede çok yaygın olmasına rağmen eğer iki vektör ortak özellik değerlerine sahip değilse, aynı özellik değerlerini içeren diğer vektör çiftlerinden daha küçük bir mesafeye sahip olabilmektedir [28].

$$d([x_1, x_2, \dots, x_n], [y_1, y_2, \dots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (2.3)$$

Burada, $[x_1, x_2, \dots, x_n]$ ve $[y_1, y_2, \dots, y_n]$ x ve y noktalarına ait vektörleri, n bu vektörlerin boyutunu göstermektedir.

Jaccard benzerliği, Eş. 2.4 ile görüldüğü gibi kümelerin kesişimlerinin birleşimlerine oranıyla hesaplanmaktadır. Dokümanlarda Jaccard benzerliği, her iki dokümanda da

paylaşılan terimlerin toplam ağırlığının, iki dokümandan herhangi birinde bulunan ancak paylaşılmayan terimlerin toplam ağırlıklarına oranıyla bulunmaktadır. Jaccard, 0 ve 1 arasında bir değere sahiptir. Uzaklık ise Eş. 2.5'te olduğu gibi benzerliğin birden çıkarılması ile elde edilmektedir [27].

$$SIM(x,y) = \frac{x \cap y}{x \cup y} = \frac{x \,.\, y}{x^2 + \, y^2 - x \,.\, y} \tag{2.4}$$

Burada x ve y iki kümeyi, $SIM(x,y)$ ise iki küme arasındaki Jaccard benzerliğini göstermektedir.

$$d(x,y) = 1 - SIM(x,y) \tag{2.5}$$

Burada x ve y iki kümeyi, $d(x,y)$ ise iki küme arasındaki Jaccard uzaklığını göstermektedir.

Kosinüs benzerliği, kosinüs açısı kullanılarak hesaplanan uzaklık ölçütüdür. Dokümanlar, terim vektörleri olarak ifade edilip, benzerlikler vektörler arasındaki açının kosinüsü ile ölçülmektedir. Bu nedenle, Jaccard ölçütünde olduğu gibi yalnızca [0, 1] aralığında değer alabilmektedir. Verilen x ve y vektörleri için bu benzerliğin hesaplanmasında kullanılan denklem Eş. 2.6'da sunulmuştur. Kosinüs benzerliğinin önemli bir özelliği doküman uzunluğundan bağımsız olmasıdır [27, 28].

$$SIM([x_1, \dots, x_n], [y_1, \dots, y_n]) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2} \sqrt{\sum_{i=1}^{n} {y_i}^2}} \tag{2.6}$$

Burada, $[x_1, x_2, \dots, x_n]$ ve $[y_1, y_2, \dots, y_n]$ x ve y dokümanlarına ait vektörleri, n bu vektörlerin boyutunu göstermektedir.

Dokümanlarda en sık kullanılan terimler mutlaka en çok bilgi içeren terimler değildir. Aksine, az sayıda dokümanda sıkça, ancak diğer dokümanlarda nadiren geçen terimler söz konusu belirli doküman grubu için daha ilgili ve o dokümanlara özel olma eğilimindedir. Bu nedenle bu terimleri kullanmak benzer dokümanları bulmak için daha kullanışlı bir ölçüttür. Bu önemli terimleri tespit etmek amacıyla, Term Frequency and Inverse Document Frequency (TF-IDF) kullanılmaktadır. Bu yaklaşımda, d dokümanındaki t teriminin sıklığı,

tüm doküman koleksiyonundaki sıklığı göz önüne alınarak Eş. 2.7'deki gibi hesaplanmaktadır [27].

$$TF - IDF(d,t) = tf(d,t) \times \log\left(\frac{D}{n}\right) \tag{2.7}$$

Burada $tf(d,t)$, t teriminin d dokümanındaki frekansını, D toplam doküman sayısını, n ise t teriminin geçtiği doküman sayısını ifade etmektedir [27].

Okapi BM25, metin bilgilerinin çıkarılması için kullanılan bir benzerlik fonksiyonudur. Belirli bir sorgu dokümanı (q) ile koleksiyondan bir doküman (d) arasındaki benzerliği ölçmektedir. Benzerlik ölçüsü temel olarak her iki dokümanda da ortak olan terimlerin katkılarının toplamıdır. Doküman terim frekansı ve sorgu terim frekansını kontrol eden bir yaklaşımdır. Bu ölçütün bir versiyonu Eş. 2.8 ile verilmiştir.

$$OKA(q,d) = \sum_{t \in q, t \in d} log\left(\frac{N - f_t}{f_t}\right)\left(\frac{f_{d,t}}{f_{d,t} + \sqrt{f_d}/av(\sqrt{f_d})}\right) \tag{2.8}$$

Burada $f_{d,t}$ t teriminin d dokümanında geçme sıklığını, f_t t teriminin geçtiği doküman sayısını, N koleksiyondaki toplam doküman sayısını ifade etmektedir.

2.2. Derlem Tabanlı Algoritmalar

Derlem tabanlı benzerlik, kelimeler arasındaki anlamsal benzerliği belirleyen bir ölçüttür. Derlem, dil araştırması için kullanılan geniş bir metin koleksiyonudur. Hyperspace Analogue to Language (HAL) bu kapsamdaki algoritmalardan biridir. Bu algoritmada, kelimeler arasındaki ilişkinin gücü satır ile kelimeler ise sütun ile temsil edilmektedir. Komşu kelimenin odak kelimesinden önce ve sonra yer alıp almadığına bağlı olarak kelime sıralaması elde edilmektedir. Latent Semantic Analysis (LSA) ise bu kategorideki en önemli tekniklerdendir. Veri kümesinde her paragraf için kelime sayısını içeren bir matris oluşturulmaktadır. Yakın anlamlı kelimelerin çoğunlukla benzer grupta yer alacağı varsayılmaktadır [26].

2.3. Bilgi Tabanlı Algoritmalar

Bilgi tabanlı benzerlik, kelimeler arasındaki benzerlik derecesini belirlemeye çalışan anlamsal bir benzerlik ölçüsüdür. Anlamsal ağlardan türetilen çeşitli bilgileri kullanmaktadır. En popüler semantik ağ, İngilizce'de büyük bir sözcüksel veri tabanı olan WordNet olarak bilinmektedir. İsimler, fiiller, zarflar ve sıfatlar, synset'ler olarak bilinen bilişsel eş anlamlı kümeler halinde gruplandırılmaktadır. Kavramsal-anlamsal ve sözcüksel ilişkiler yoluyla synset'ler birbirine bağlanmaktadır. Kelimeler ve kavramlar arasında anlamsal benzerlik ölçütleri tanımlanmıştır.

3. ÖNİŞLEMLER

Metin işlemede yaygın olarak kullanılmakta olan önişlemler arasında etkisiz sözcüklerin (stop words) çıkarılması, büyük-küçük harf değişimi, tokenize, gövdeleme (stemming) gibi işlemler bulunmaktadır.

3.1. Tokenize

Metin işlemede gerçekleştirilen önişlemlerin belki de en başında tokenize işlemi gelmektedir. Tokenize bir metni sözcüklere, sözcük öbeklerine veya diğer anlamlı parçalara ayırma işlemidir. Parçalara ayrılan metinde diğer işlemlerin gerçekleştirilmesi daha kolay olacağından çoğunlukla ilk aşamada yapılan işlemdir. Ayırma işlemi genellikle boşluk ya da noktalama işaretleri gibi karakterlere göre gerçekleştirilmektedir [29, 30].

3.2. Standartlaştırma ve Temizleme

Tokenize işleminde metin parçalara ayrıldıktan sonra standartlaştırma ve temizleme işlemleri gerçekleştirilmektedir. Standartlaştırma kısmında yaygın olarak küçük harf dönüştürmesi yapılmaktadır. Sözcüklerin büyük veya küçük harf ile yazılmış olması anlamsal açıdan bir fark yaratmamaktadır. Ancak, işlenen metinde sadece büyük-küçük harf ayrımından doğacak bir farklılık olması istenmemektedir. Bu nedenle çoğu çalışmada tüm metin karakterleri küçük harfe dönüştürülmektedir. Küçük harf dönüşümüne ek olarak metinde bazı temizleme işlemleri yapılmaktadır. Bunlar, fazla boşlukların kaldırılması, sayılar, noktalama işaretleri ve özel karakterlerin temizlenmesi gibi işlemlerdir [29, 30].

3.3. Etkisiz Sözcükler

Etkisiz sözcükler, belirli bir anlam ifade etmeksizin metinlerde yaygın olarak kullanılan sözcüklerdir. Edat ve bağlaçlar bu sözcüklere örnek olarak verilebilmektedir. İngilizce'de en yaygın kullanılan sözcükler the, to, be, of, and gibi kelimelerdir. Bu sözcükler, diğer sözcüklerle birlikte anlamlı cümleler oluşturmak için kullanılmaktadır. Ancak metin işlemede, benzerlik bulma gibi işlemlerde bu sözcüklerin varlığı bir fayda sağlamamaktadır. Tam tersine benzer olmayan iki metinde aynı bağlaçların kullanılmış olması yanlış benzerlik

durumu ortaya çıkarmaktadır. Aynı şekilde benzer iki metinde farklı bağlaçların kullanılması benzerlik oranını düşürecektir. Bu nedenle, metin verileri üzerinde gerçekleştirilen çalışmalarda etkisiz sözcüklerin ilgisiz olduğu varsayılmakta ve önişleme aşamasında kaldırılmaktadır. Bunun için, kullanılacak metnin diline özgü olarak bu sözcüklerin listesini içeren bazı kütüphaneler kullanılmaktadır [29, 30].

3.4. Gövdeleme

Sözcükler genellikle cümle içindeki işlevlerine göre değiştirilerek farklı formlara dönüştürülmektedir. Bu dönüşümler, iki veya daha fazla kelimeyi birleştirme, başa veya sona ek ekleme, benzer anlamlı yeni sözcükler türetme gibi farklı dilbilimsel işlemler sonucunda oluşmaktadır. Bu sözcük formları genellikle aynı anlama sahiptir [31]. Gövdeleme, ek almış ya da türetilmiş sözcüklerin gövdelerini elde etme işlemidir [29]. Bu işlem, metin işleme uygulamalarında doğruluğu iyileştiren bir yöntem olarak görülmektedir. Bunun nedeni, dokümanlar arasında tam olarak eşleşmeyen sözcüklerin uyumsuzluğundan kaynaklanan sorunun gideriliyor olmasıdır.

Gövdeleme işleminde çeşitli değişken terimler tek bir terime indirgendiğinden, bu işlem aynı zamanda dizin dosyasının boyutunu küçülten bir mekanizma olarak da görülmektedir. Bu boyut küçültme kimi zaman orijinal boyutun yarısını bulabilmektedir. Gövdeleme, üzerinde çok çalışılmış bir konu olduğundan literatürde önerilen çok sayıda teknik bulunmaktadır. Bu teknikler genel olarak kural tabanlı ve istatistiksel olmak üzere iki kategoriye ayrılmaktadır [31].

3.4.1. Kural tabanlı gövdeleme

Kural tabanlı teknikler dile özgüdür ve sözcüğün morfolojik varyantlarını temel biçimiyle eşleştirmek için önceden tanımlanmış, dile ilişkin belirli kuralları kullanmaktadır. Bu kurallar dilbilimciler tarafından oluşturulmaktadır. Kurala dayalı yöntemlerin çıktısı nitelik olarak istatistiksel yöntemlerin çıktısına göre daha iyidir. Bunun sebebi, istatistiksel yöntemlerin ekleri silmekle yetinmesi, kural tabanlı yöntemlerin ise gerekirse tüm kelimeyi değiştirebilmesidir. Ancak, kural tabanlı yöntemler dil uzmanları, sözlükler, kök tabloları gibi birçok kaynak gerektirmekte ve oluşturulmaları uzun zaman almaktadır [31].

Kural tabanlı gövdeleme yöntemleri kaba kuvvet (brute force), ek çıkarma (affix removal) ve morfolojik yöntemler olmak üzere üç alt kategoriye ayrılmaktadır [31, 32].

Kaba Kuvvet Gövdeleme

Bu teknikte, kelimenin kökünü elde etmek için bir arama tablosundan yararlanılmaktadır. Tabloda farklı formlarda sözcükler ve bu sözcüklerin gövde haline ait eşleşmeler bulunmaktadır. Gövdeleme işlemi esnasında ilgili sözcüğün gövde şekli bu tablodan bulunarak elde edilmektedir. Bu tekniklere tablo arama veya sözlük tabanlı teknikler de denilmektedir. En büyük avantajları, dile özgü tanımlanmış genel kurallara uymayan özel formdaki sözcüklerde sorun yaşanmamasıdır. Dezavantajları ise sözcüklere ait tüm formların toplanarak tabloya kaydedilmek zorunda olmasıdır ve bunun her zaman mümkün olmamasıdır. Bu nedenle, tabloda bulunmayan kelimelerde işlem gerçekleştirilememektedir. Ayrıca, tabloların depolanması için çok fazla alana ihtiyaç duyulmaktadır [32].

Ek Çıkarma ile Gövdeleme

Ek, bir kelimenin son ekini veya ön ekini ifade etmektedir. Bu teknikte ekler farklı formdaki sözcüklerden çıkarılmaktadır. Bunun için belirli bağlama duyarlı kuralların yanı sıra bir ek listesi kullanılmaktadır. Bu tekniğin en büyük zayıflıklarından biri, bazı durumlarda son eklerin kaldırılmasından sonra elde edilen köklerin dilin gerçek kelimeleri olmamasıdır. Bu sorunun ortaya çıkartacağı sorun uygulandığı alana göre değişmektedir [32]. Benzerlik tespiti içeren uygulamalarda bu kelimelerin tüm varyantları aynı şekilde kırpılacağından kelime anlamsız hale gelmiş olsa da benzerlik açısından bir problem oluşturmamaktadır. Ancak sözcük anlamının önemli olduğu uygulamalarda bu şekilde yanlış ayıklanmış bir kelimeden anlam çıkarmak zor alacaktır.

Morfolojik Gövdeleme

Dilin çekim morfolojisini göz önünde bulunduran teknikler, isimlerin formları, fiiller, tekilden çoğul hale geçme gibi sözdiziminden kaynaklanan kelime değişikliklerini tespit edebilmektedir. Böylece cins, zaman, durum, sayı veya kişiye bağlı olarak kelime biçimlerindeki değişiklikler tespit edilmektedir. Türevsel morfolojiyi göz önüne alan teknikler ise nominalizasyon (başka bir sınıftan türetilmiş isim), sıfatlardan türetilmiş

sözcükler, fiilden türetilmiş isimler gibi farklı durumları tespit etmektedir. Bu algoritmalar dilin hem sözdizimini hem de anlambilimini dikkate aldığından verimliliği oldukça yüksektir. Ancak bu algoritmaların geliştirilmesi, dil ve morfolojisi hakkında tam bilgi gerektirmektedir [31, 32].

3.4.2. İstatistiksel gövdeleme

İstatistiksel teknikler, bir dile ait kuralları öğrenmek için denetimsiz öğrenmeden faydalanmaktadır. Bu teknikte, sözlük ve çevresel derlem kullanılarak morfolojik olarak ilişkili kelimeler gruplandırılmaktadır. Böylece dil uzmanlarına veya herhangi bir ek dilbilimsel kaynağa ihtiyaç kalmamaktadır. Bu sebeple dilden bağımsız gövdeleme veya derlem temelli gövdeleme olarak da adlandırılmaktadır. İstatistiksel teknikler, yeni bir dili çok az çabayla sisteme dahil edebilmektedir. Bu özellik sayesinde bilgi erişimiyle ilgili uygulamalarda oldukça avantajlıdır. Kaynakların mevcut olmadığı veya etkili sonuçlar sağlamak için eksik olduğu birçok dil için herhangi bir ön bilgi gerektirmediğinden dolayı oldukça kullanışlıdır. Ancak dilin morfolojisini öğrenme aşaması çokça hesaplama içerdiği için uzun zaman almaktadır [31, 32].

4. KÜMELEME ALGORİTMALARI

Kümeleme (clustering), bir grup benzer nesnenin, kümeler oluşturmak için bir araya getirildiği bir veri madenciliği tekniğidir. Bir kümedeki nesnelerin arasındaki benzerlik, diğer kümelerdeki nesnelerle aralarındaki benzerliğe göre çok yüksektir. Birçok veride, önceden bir etiketleme bilgisi bulunmadığı için bu yöntemde nesneler benzerlik temelinde gruplandırılmaktadır. Bu nedenle, kümeleme bir denetimsiz öğrenme tekniği olarak veri analizinde önemli bir rol oynamaktadır. Son yıllarda veri miktarının çok büyük bir hızla artmasının bir sonucu olarak kümeleme için birçok yöntem oluşturulmuştur. Ancak, her kümeleme algoritmasının kendine özgü güçlü ve zayıf yanları bulunmaktadır [33].

Dokümanlar arasındaki benzerliği ölçmek ve benzer dokümanları birlikte gruplandırmak amacıyla kümeleme yöntemleri dokümanlar üzerine de uygulanmaktadır. Dokümanlarda kümeleme, web madenciliği, arama motorları ve bilgi erişimi gibi çeşitli alanlardaki geniş uygulanabilirliği nedeniyle üzerinde yoğun çalışılan bir konudur [34].

4.1. Bölünmeli Kümeleme

Bölünmeli (partitional) yöntemler, n tane nesneyi, k <= n olacak şekilde k tane kümeye bölerek işlem yapan bir kümeleme tekniğidir. Her kümenin birbiri ile benzer elemanları içermesi ve küme dışındaki nesnelerden farklı nesnelere sahip olması gerekmektedir. Bu şekilde elde edilen k kümenin her birinde en az bir nesne olmalıdır. Ayrıca veri içerisindeki her nesne kesinlikle bir küme ile ilgili olmalıdır. Bölünmeli kümeleme için en yaygın kullanılan yöntemler k-means ve k-medoids yöntemleridir [35].

Doküman kümelemede, denetimsiz öğrenme algoritması olarak k-means basit yapısından ötürü yaygın olarak kullanılmaktadır. Bu algoritmada, rastgele k tane nokta, k tane küme için merkez olarak seçilmektedir. Her küme için merkezler atandıktan sonra veri kümesinde her bir nokta kendine en yakın küme ile ilişkilendirilmektedir. Her nokta bir kümeye atandıktan sonra k tane kümenin yeni merkezleri küme içerisindeki noktaların ortalaması alınarak hesaplanmaktadır. Algoritma her iterasyonda yeni merkezler belirleyerek noktaları en yakın yeni merkeze ait kümelere atamaktadır. Algoritma, merkezler değişmeyinceye kadar devam etmektedir [36][37].

Ancak, k-means algoritmasında başlangıçta küme merkezleri rastgele belirlendiği için aynı giriş verisi için tekrar çalıştırıldığında farklı kümeler oluşabilmektedir. Ayrıca sapan verilerden (outlier) oldukça fazla etkilenmektedir. Bu nedenle, k-medoids algoritması merkez belirlemede veri ortalaması yerine küme içerisinde en merkezi pozisyonda bulunan noktayı (medoid) kullanmaktadır. Bu algoritmada, başlangıç durumu da dahil olmak üzere, merkezler veri içerisinde bulunan noktalar arasından seçilmektedir. k-means ile karşılaştırıldığında, sapan verilerden daha az etkilendiğinden dolayı, daha güvenilir bir algoritma olmasına karşılık büyük veri kümeleri için uygun bir algoritma değildir.

Bölünmeli kümeleme algoritmaları arasında Clustering Large Applications based on RANdomized Search (CLARANS) [38] gibi algoritmalar da bulunmaktadır [33]. CLARANS, her düğümün bir k-medoids kümesi olduğu bir grafik oluşturmaktadır. Bir medoid değiştirildikten sonra elde edilen kümeleme mevcut kümelemenin komşusu olmaktadır. CLARANS rastgele bir düğüm ile başlayarak bu düğümü komşuları ile karşılaştırmaktadır. Daha düşük hatalı bir komşu bulunursa süreç bu yeni düğümle devam etmektedir. Aksi takdirde yerel bir minimum bulunmakta ve algoritma tüm yerel minimumlar bulunana kadar yeniden başlamaktadır [39, 40].

Bölünmeli kümeleme algoritmaları, büyük veri ile çalışırken avantajlı olsalar da başlangıçta en iyi küme sayısını belirleme kolay değildir. Başlangıçta küme sayısının yanlış belirlenmesi zayıf kümeleme sonuçlarına neden olmaktadır [36]. Bunun için öncelikle optimum küme sayısının dirsek (elbow) metodu gibi bir yöntemle belirlenmesi gereklidir.

4.2. Hiyerarşik Kümeleme

Hiyerarşik Kümeleme Algoritmaları (Hierarchical Clustering Algorithms-HCA), bir ağaç şeklinde bağlanan kümeler oluşturmaktadır. Bu gösterime dendrogram adı verilmektedir. Ağaç içerisinde ebeveyn düğüm ana konuyu ve çocuk düğümler alt konuları temsil etmektedir. Bu şekilde, algoritma dokümanlar için anlamlı hiyerarşik kümeler sağlamaktadır [41].

HCA, birleştirici (agglomerative) ve ayırıcı (divisive) olmak üzere iki türe ayrılmaktadır. Birleştirici algoritmalar, aşağıdan yukarıya bir yol izleyerek tek noktalı kümelerle başlayıp her adımda en benzer veya en yakın kümeleri birleştirmektedir. Bu, küme benzerliği veya

uzaklığının tanımını gerektirmektedir. Ayırıcı algoritmalar ise, yukarıdan aşağıya bir yol izleyerek, bütün noktaları içeren bir küme ile başlayıp her adımda kümeyi ayırarak devam etmektedir. Bu yöntemde, her adımda hangi kümenin bölüneceğine ve bölünmenin nasıl gerçekleştirileceğine karar verilmelidir. Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), Clustering Using REpresentatives (CURE), RObust Clustering using linKs (ROCK) en popüler HCA algoritmalarıdır [42].

BIRCH algoritması, tüm veri noktalarını taramaksızın karar alma üzerine geliştirilmiştir. Her veri noktasının eşit derecede önemli olmadığı varsayımını kullanmaktadır. Maliyetleri en aza indirirken mümkün olan en iyi alt kümeleri türetmeye çalışmaktadır. Ayrıca, veri kümesinin tamamını önceden gerektirmeyen artımlı bir yöntemdir. BIRCH algoritmasında kümeler düzgün boyutlara ve şekillere sahip olmadığında, verileri yeniden dağıtmak için yalnızca merkezleri kullanmak sorunlara yol açmaktadır.

CURE, rastgele örnekleme ve segmentasyona dayanan bir yöntem benimsemektedir. CURE algoritmasında, öncelikle veri kümesinden alınan rastgele bir örnek bölünmekte ve her bölüm kısmen kümelenmektedir. Her küme tek bir merkez ile temsil edilmek yerine birbirinden olabildiğince uzak bir grup nokta ile temsil edilmektedir. Daha sonra bu noktalar, belirli bir oranda merkeze doğru yer değiştirmektedir. Yer değiştirme işleminin sonunda, farklı kümelere ait eşik değerden daha yakın noktalar tespit edilerek bu kümeler birleştirilmektedir. Kısmi kümeler ikinci bir geçiş ile tekrar kümelenerek istenilen özellikte kümeler elde edilmektedir.

ROCK, veri kümeleri arasındaki bağlantılara dayanan bir birleştirici kümeleme tekniğidir. İki küme arasındaki bağlantıların sayısı, veri kümesinde sahip oldukları ortak komşuların sayısıdır. Veriler arasındaki bağlantıların hesaplamasından sonra algoritma her küme için tek bir nokta ile başlamaktadır. Aralarında bağlantı olan kümeler kalmayana veya gerekli sayıda küme elde edilene kadar kümeler birleştirilmeye devam etmektedir [43].

4.3. Yoğunluk Tabanlı Kümeleme

Bu tür kümeleme yöntemleri, nesnelerin yoğunluğuna göre kümeler oluşturmaktadır. Veri alanının yüksek yoğunluğuna sahip bölgedeki verilerin aynı kümeye ait olduğu düşünülmektedir. Farklı şekillerde kümeler oluşturabilmek için oldukça uygun bir tekniktir.

Yoğunluk esaslı kümeleme, kümelerin düşük yoğunluklu bölgelerini yüksek yoğunluklu bölgelerden ayırmaya yardımcı olmaktadır. Nesnelerin yüksek yoğunluktaki bölgeleri, kümeler oluşturmak için bir araya getirilmektedir. Algoritmanın son bulması için yoğunluk parametresinin verilmesi gerekmektedir. Gürültülü verileri göz önüne alabilmekte ve tüm verileri tek bir kez üzerinden geçerek işlem yapabilmektedir [35].

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) ve Ordering Points To Identify the Clustering Structure (OPTICS) en tipik yoğunluk tabanlı algoritmalardır. DBSCAN, doğrudan bu tür kümeleme algoritmalarının temel fikrinden yararlanılarak geliştirilmiş en iyi bilinen yoğunluk temelli kümeleme algoritmasıdır. Birbirine yakın bir şekilde bulunan noktaları bir araya getirip düşük yoğunluklu bölgelerde tek başına bulunan noktaları sapan veri olarak işaretleme yapan bir algoritmadır. OPTICS ise DBSCAN algoritmasının bir versiyonu olarak en büyük zayıflıklarından birini ele almaktadır. Değişen yoğunluktaki verilerde anlamlı kümeleri tespit etme problemini çözmek için veri kümesinde bulunan noktalar sıralanmaktadır. Böylece en yakın noktalar sıralamada komşu olmaktadır [33].

4.4. Akış Verisi Kümeleme Algoritmaları

Akış verisi kümelemede kullanılan algoritmaların yüksek hızda akan anlık verileri kümelemesi gerekmektedir. Bu nedenle, tek yönlü tarama yapan, çok seviyeli ve çok yönlü algoritmalar tercih edilmektedir. Akış verisinde veri boyutu genellikle sonsuz olduğundan veriyi bir yerde biriktirip daha sonra işlem yapılamamaktadır.

4.4.1. STREAM algoritması

STREAM, böl ve yönet yaklaşımına dayanan, k-means benzeri bir bölünmeli kümeleme algoritmasıdır. Algoritma iki aşamadan oluşmaktadır. İlk aşamasında veri, bucket olarak adlandırılan parçalara bölünmektedir. Her bucket için k-median uygulanarak k tane küme elde edilmektedir. Küme merkezleri sahip oldukları veri miktarlarına göre ağırlıklandırılmaktadır. Bu noktada asıl veriler göz ardı edilerek ağırlıklandırılmış veri merkezleri veri olarak kabul edilmektedir. İkinci aşamada ise ağırlıklandırılmış veri merkezleri kümelenerek daha az sayıda küme elde edilmektedir [44].

4.4.2. CluStream algoritması

CluStream, birçok zaman penceresi üzerinden akış yapma fikrini benimseyerek kümeleme sürecinde kümelerin davranışlarını ve gelişimlerini daha iyi anlayan bölünmeli bir kümeleme algoritmasıdır. CluStream algoritması veri özeti istatistiklerini çevrimiçi bileşeninde saklayarak kullanıcıya zaman açısından esneklik sağlamaktadır. Bu istatistikleri çevrimdışı kümedeki diğer parametrelerle birlikte nihai kümeleme sonuçlarını elde etmek için kullanmaktadır. Çevrimiçi bileşendeki veri mikro kümeler yapısında özetlenirken çevrimdışı bileşende k-means uygulanarak mikro kümeleri daha büyük kümelere dönüştürmektedir [45]. CluStream dairesel olmayan şekilleri bulmakta çok başarılı bir algoritma değildir. Ayrıca yapısı gereği büyük veri kümelerinde başarısı düşüktür ve sapan verilerden etkilenmektedir.

CluStream algoritmasının gelişmiş bir versiyonu olan StreamSamp algoritması ise temel anlamda verimli bellek kullanımı ile öne çıkmaktadır. StreamSamp'te sabit boyutlu bellek kullanımı benimsenmiştir. İki çeşit bellekleme yapılmaktadır; kısa boyutlu bellekleme küçük boyutlu periyodik depolama yaparken, uzun boyutlu bellekleme büyük boyutlu periyodik depolama yapmaktadır [44].

4.4.3. ClusTree algoritması

ClusTree algoritması da kümeleme işlemini çevrimiçi ve çevrimdışı olmak üzere iki aşamada gerçekleştiren hiyerarşik bir kümeleme algoritmasıdır. Sınırlı bellek ve zaman gibi birçok probleme yönelik çözümler üreten bir algoritmadır. Kümeleme işlemini veri akış hızına göre uyarlamaktadır. Buna ek olarak, ClusTree yavaş akışları işlemek için iniş stratejileri, hızlı akışları işlemek için ise toplama mekanizmaları kullanmaktadır. Ayrıca ClusTree, daha yeni verilere daha fazla önem vermek için üssel bir işlev kullanmaktadır [45].

4.4.4. DenStream algoritması

Akış verisi kümelemede, tüm verilerin elde edilmesi mümkün değildir. Bu nedenle, tek geçiş, bilinmeyen parametreler ve değişen veriler ile kümeleme yapabilen algoritmaların kullanımı önem arz etmektedir. DenStream yoğunluk tabanlı bir kümeleme yaklaşımıdır. Çevrimiçi ve çevrimdışı olmak üzere iki bölümden oluşmaktadır. DBSCAN algoritmasının

geliştirilmiş bir versiyonudur ve mikro kümeler kullanmaktadır [44]. Mikro kümeleme tekniği, akış nesneleri hakkında gerekli bilgileri kaydederek verileri etkili bir şekilde sıkıştırmaktadır. DenStream algoritması her veri noktası için bir ağırlık belirlemektedir. Birleştirme gerçekleştirilmeden önce, eski veri noktalarını ve sapan verileri silme yeteneği nedeniyle hızlı bir algoritmadır. Veri noktalarını bir mikro kümede birleştirmediğinden sapan verilerin tespit edilmesini ve zamandan tasarruf edilmesini sağlamaktadır [45].

4.4.5. D-Stream algoritması

D-Stream, akış verilerini kümelemek için kullanılan ızgara (grid) yapısından yararlanan yoğunluk tabanlı bir yaklaşımdır. Çevrimiçi ve çevrimdışı olmak üzere iki bölümden oluşmaktadır. Çevrimiçi kısım, giriş verilerini sürekli olarak okuyarak bir yoğunluk ızgarasına eşlemektedir. Çevrimdışı kısım ise ızgara yoğunluğunu hesaplayarak, grid listesini ve kümelemeyi düzenlemektedir. D-Stream, rastgele şekillerin kümelerini etkili ve verimli bir şekilde oluşturmak için yoğunluk azaltma tekniği kullanarak dinamik değişiklikleri yakalamaktadır [45].

4.4.6. HPStream algoritması

HPStream, eski verilerin ağırlığını azaltan CluStream algoritmasının gelişmiş bir versiyonudur. Çok boyutluluğu desteklediğinden CluStream'e göre daha avantajlıdır. Yeni gelen verilere önem verilen uygulamalarda kullanım için oldukça uygundur. Bu nedenle, veri güncelliğini ön planda tutan sensör uygulamalarında çokça kullanılan bir yaklaşımdır. Çok boyutluluğu desteklemesine rağmen çok boyutluluk konusunda optimum bir sonuç verememektedir. Çoğu zaman küme şekillerini belirlemede çok başarılı olamamaktadır. Daha iyi performans elde edebilmek için verinin tamamı hakkında yeterli bilgiye sahip olmak gerekmektedir. Bu da akış verisi için mümkün olmamaktadır [44].

5. BÜYÜK BOYUTLU DOKÜMANLARDA DERİN ÖĞRENME İLE KÜMELEME

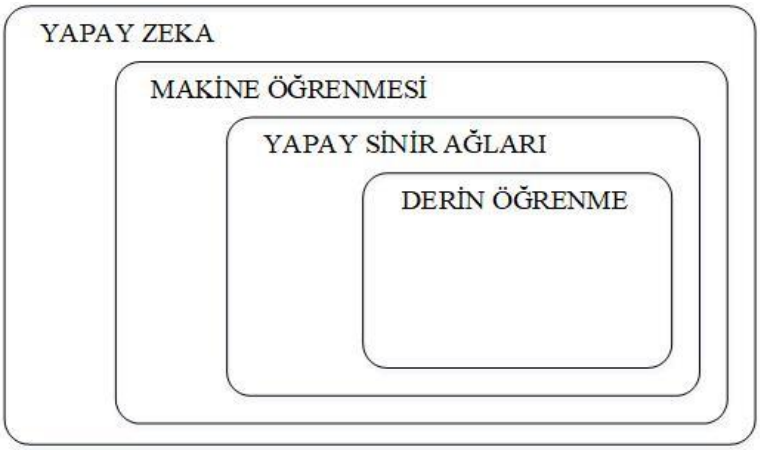
İnternet kullanımının yaygınlaşması ve teknolojide yaşanan hızlı gelişmeler ile verinin çeşitliliği ve hacmi gün geçtikçe artmaktadır. Basit ve açık olmayan, önceden bilinmeyen ancak yararlı olabilecek bilginin bu büyük miktardaki veriden çıkarılması gerekmektedir. Büyük veri üzerindeki bu anlamlı örüntülerin çıkarılmasında başarılı sonuçlar veren yapay zeka, makine öğrenmesi ve derin öğrenme yöntemleri oldukça önemli hale gelmiştir.

Yapay zeka, makinelere insan zekasını kazandırmayı amaçlamaktadır. İlk yıllardaki çalışmalarda, belirlenmiş bir komut listesine ve belirlenen kurallara dayanarak kararlar alan bilgisayarlar, daha sonra makine öğrenmesi yöntemlerinin gelişmesiyle birlikte katı kurallara sahip kurallar yerine büyük veri kümelerini kullanarak öğrenerek kararlar vermeye başlamıştır. Makine öğrenmesi, yapay zekanın deneysel verilere dayalı davranışlar geliştirmesine izin veren bir dalı olarak bilgisayarların kendi başlarına öğrenmelerini sağlamaktadır [46].

Makine öğrenmesi algoritmalarının en güçlü olanı ise, insan beyninin öğrenme sürecinden etkilenerek oluşturulan yapay sinir ağıdır (Artificial Neural Networks - ANN). Bu yaklaşım ilk kez 1943 yılında insan beynindeki hücrelerin yapısının matematiksel modellemesi yapılarak gerçekleştirilmiştir. Buradaki temel amaç insan beynindeki nöronların çalışmasından faydalanılarak benzer bir yaklaşımla makinenin öğrenmesini ve buna göre davranmasını sağlamaktır [46].

Bu yapay sinir hücre modeli zaman içerisinde geliştirilerek makine öğrenmesi yaklaşımlarında sıklıkla kullanılmaya başlanmıştır. Bir ANN, biyolojik beyindeki nöronları modelleyen ve yapay nöron adı verilen bağlı düğümler kümesine dayanmaktadır. Biyolojik beyindeki sinapslar gibi her bağlantı bir yapay nörondan diğerine sinyal iletmektedir. Sinyal alan yapay bir nöron onu işleyip daha sonra diğer nöronlara çıkış olarak gönderebilmektedir. Günümüzde bu yaklaşım daha ileri seviyelere taşınarak derin öğrenme modelleri kullanılmaya başlanmıştır. Derin öğrenme geniş makine öğrenme yöntemleri ailesinin yapay sinir ağlarına dayanan bir parçasıdır. Günümüzde, tek nöron yapısını esas alan perceptron (algılayıcı) ile oluşturulan tek katmanlı yapay sinir ağı, çok katmanlı hale

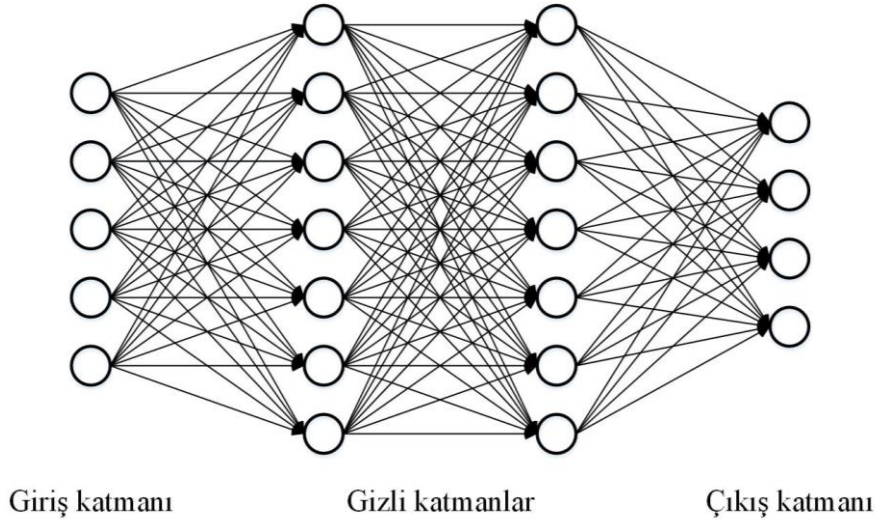
getirilerek derin yapay sinir ağları haline gelmiştir [47]. Yapay zeka, makine öğrenmesi, sinir ağları ve derin öğrenme kavramları arasındaki ilişki, basitçe Şekil 5.1 ile gösterilmiştir.



Şekil 5.1. Kavramlar arasındaki ilişki

Her ne kadar temelleri çok daha öncelere dayansa da, derin öğrenme ifadesi 2006 yılında Geoffrey Hinton ve Ruslan Salakhutdinas tarafından yayınlanan makale ile ortaya atılmış ve o tarihten sonra derin öğrenme çalışmaları başlamıştır [46]. Derin öğrenme mimarisi öğrenmeye bağlı olan ve birçoğu doğrusal olmayan giriş-çıkış eşleşmelerini hesaplayan çok katmanlı bir modül yığınıdır. Yığındaki her modül, çıktının hem seçiciliğini hem de değişmezliğini artırmak için kendi girişi üzerinde birtakım hesaplamalar yapmaktadır [47]. Bu doğrusal olmayan işlem birimlerinden oluşan katmanlar, özellik çıkarma ve dönüştürme için kullanılmaktadır. Her ardışık katman, önceki katmandaki çıktıyı giriş olarak almaktadır [48]. Bu katmanları gösteren basit bir sinir ağı yapısı Şekil 5.2'de sunulmuştur.

Derin öğrenme temel olarak verinin temsilinden öğrenmeye dayalıdır. Üst düzey özellikler, alt düzey özelliklerden türetilerek hiyerarşik bir temsil oluşturulmaktadır. Bu özelliklerin içinden bazıları veriyi daha iyi temsil etmektedir. Derin öğrenme yöntemleri, elle çıkarılan özellikler yerine veriyi en iyi temsil eden hiyerarşik özellik çıkarımı yapmaktadır. Alt katman ile üst katman arasında bağlantılı hiyerarşik bir yapı bulunmaktadır. Özellik çıkarımı için özel bir safha bulunmamakta, sinir ağı içerisinde otomatik olarak yapılmaktadır. Bu da derin öğrenmenin en büyük avantajlarındandır [48].



Şekil 5.2. Basit bir sinir ağı

5.1. Derin Öğrenme Modelleri

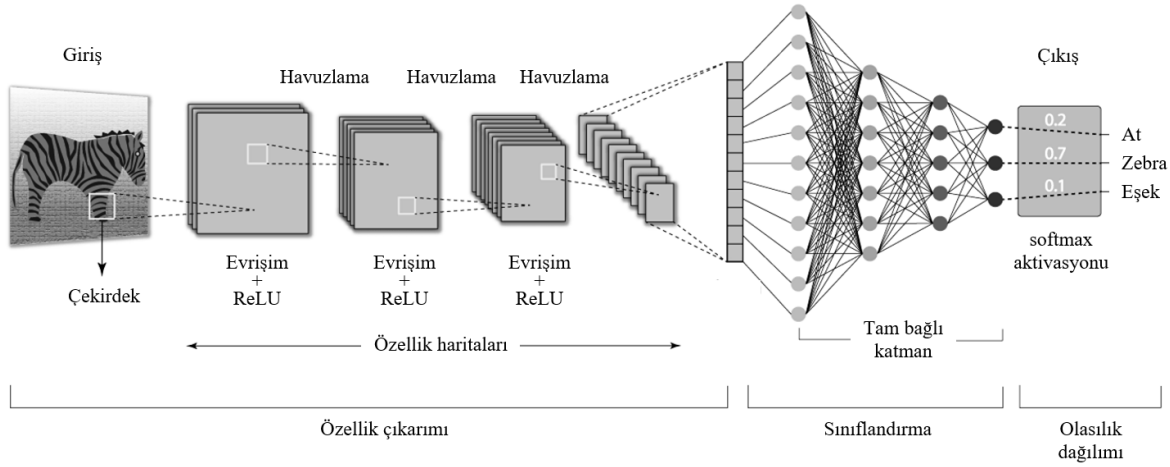
Derin öğrenmede her problemin çözümü için aynı yapay sinir ağı modeli kullanılmamaktadır. Yapay sinir ağları kullanım yerlerine ve amacına göre farklılık göstermektedir. Kullanılması gereken model, problemin yapısına ve veri türüne göre değişmektedir. Evrişimli Sinir Ağı (Convolutional Neural Network - CNN), Tekrarlayan Sinir Ağı (Recurrent Neural Network - RNN), Uzun Kısa Süreli Bellek (Long Short Term Memory - LSTM), Derin İnanç Ağı (Deep Belief Network - DBN) ve Çekişmeli Üretken Ağlar (Generative Adversarial Networks - GANs) en yaygın kullanılan modellerdendir [46].

5.1.1. Evrişimli sinir ağı

CNN, hayvanların görme merkezinden esinlenilerek ortaya atılmış ileri yönlü bir sinir ağıdır. Çok katmanlı algılayıcıların (Multi Layer Perceptron - MLP) bir türüdür. CNN, özellikle görüntü analizlerinin yapılması için kullanılmaktadır. Filtrelemeye dayalı yapısı görüntünün özelliğini ifade eden öznitelikleri belirgin hale getirmektedir. Filtreler, farklı boyut ve değerlerde kullanılarak baskınlık düzeyi az olan özniteliklerin ortaya çıkmasını ve sınıflandırıcı işlemlerinde başarılı sonuçlar elde edilmesini sağlamaktadır [46, 48].

CNN'ler bir veya birden fazla evrişim (convolution) ve havuzlama (pooling) katmanlarından oluşmaktadır. Bu aşamaların ardından sıradan yapay sinir ağı katmanı olan tam bağlı (fully

connected) katmana sahiptir. Tam bağlı katmanın ardından son katman olan sınıflandırma katmanı yer almaktadır [47]. Her katman kendi kendisine atanmış farklı bir işlevi gerçekleştirmektedir ve sınıflandırıcı katmanda ise sonuç üretilmektedir [46]. Şekil 5.3'te CNN'e ait örnek bir mimari görülmektedir.



Şekil 5.3. CNN mimarisi

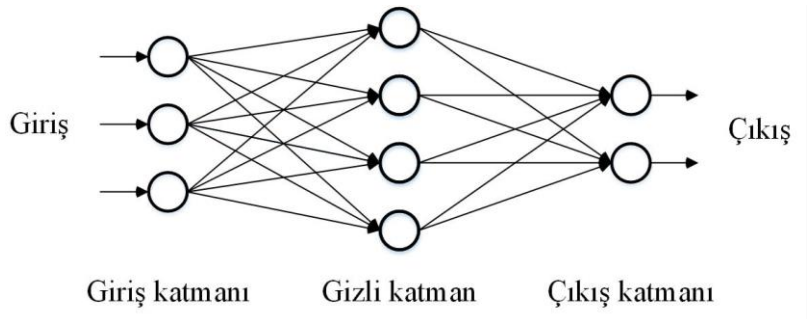
Bu mimaride giriş katmanı, işlenecek verinin ağa verildiği katmandır. Giriş verisinin boyutu büyük olduğunda veriden daha çok özellik çıkarılabildiği için modelin daha iyi öğrenmesi sağlanmaktadır. Ancak, veri miktarındaki artış gerekli donanım ihtiyacının ve eğitim süresinin artmasına sebep olmaktadır. Evrişim katmanında yer alan filtreler, giriş katmanından alınan veriler üzerinde dolaştırılarak her bir filtre için özellik haritası elde edilmektedir. Filtrelere ait özellik haritasındaki değerler veriler işlendikçe güncellenmektedir. Evrişim işlemi, bir nöronun kendi uyarı alanından uyaranlara verdiği cevaptır [48].

Eğitim verisinin boyutları çok büyük olduğunda havuzlama katmanında boyutunun azaltılması gerekmektedir. İşlenecek verinin boyutu azaldığında işlemler daha hızlı gerçekleşmektedir. Evrişim ve havuzlama katmanlarında çıkarılan özellikler kullanılarak tam bağlı katmanda sınıflandırma işlemi gerçekleştirilmektedir. Tam bağlı katman, kendinden önceki katmandan verileri alarak çıkışta belirlenmiş sınıf sayısına dönüştürmektedir [47, 49]. En son katmanda yer alan softmax fonksiyonu ile çıkış değerlerinin olasılık dağılımı elde edilmektedir.

Derin öğrenme yapıları içinde en çok kullanılan CNN [46], çoğunlukla görüntü ve ses işleme alanlarında daha başarılı olmaktadır. Görüntü işleme alanında elde edilen en iyi sonuçlar CNN mimari kullanılarak elde edilmiştir. Ancak, doğal dil işleme, biyomedikal gibi birçok farklı alanda da uygulanabilmektedir. Arama sorgusu elde etme, cümle modelleme, sınıflandırma, tahmin problemleri gibi çalışmalarda da başarılı sonuçlar elde edilmiştir [48].

5.1.2. Tekrarlayan sinir ağı

İleri beslemeli (feedforward) mimari, giriş verilerini ağdan geçirerek çıktı üreten ve sadece ileri doğru işleyen bir yapıya sahiptir. Bu yapıda öğrenme, kayıp fonksiyonunu (loss function) en aza indirmek için nöronlar arasındaki ağırlıkların sürekli güncellenmesiyle gerçekleştirilmektedir. Elde edilen çıktı ile hedef arasındaki mesafe hata olarak nitelendirilmektedir. Her iterasyonda elde edilen hataya göre ağırlıklar değiştirilerek daha doğru sonuçlara ulaşmaya ve hata en aza indirilmeye çalışılmaktadır. Böyle bir yapıda, girişin bir önceki veya bir sonraki giriş ile herhangi bir bağı bulunmamaktadır. Zaman veya sıraya bağlı bir kavram yoktur ve sadece mevcut giriş ile ilgilenilmektedir [50]. Basit bir ileri beslemeli ağ yapısı, Şekil 5.4'te sunulmuştur.

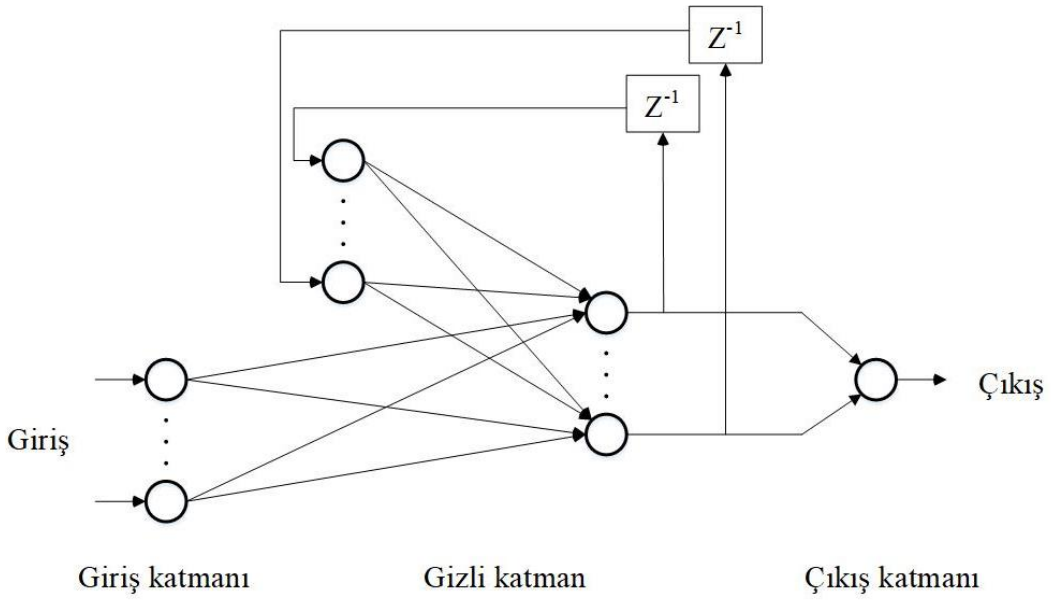


Şekil 5.4. Basit bir ileri beslemeli ağ yapısı

İlk tekrarlayan sinir ağı (Recurrent Neural Network-RNN), Jeff Elman tarafından 1990 yılında bir çalışmada uygulanmıştır. Bu çalışmada, cümle yapısı simülasyonunda kullanılan her bir kelime için gizli kalıpların üzerinde ortalama örüntü kümeleme sonucunda isim ve fiil kategorileri doğru bir şekilde ayrıştırılabilmiştir [48].

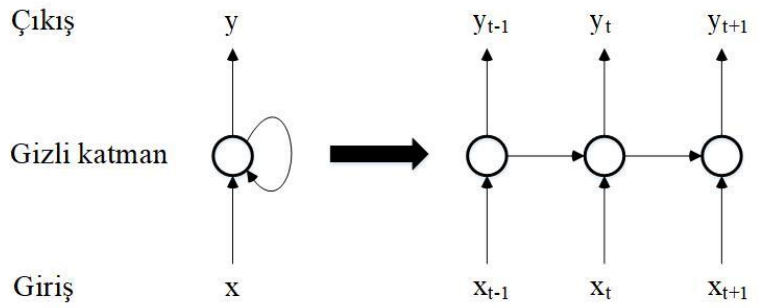
RNN, ileri beslemeli sinir ağını döngülü bağlantılar ile genişleten bir yapay sinir ağı türüdür [51]. Bitişik zaman adımlarını kapsayan kenarları dahil ederek modele sıralı zaman

kavramını kazandırmaktadır. İleri beslemeli ağlarda olduğu gibi, tekrarlayan sinir ağlarında da geleneksel kenarlar arasında devir olmayabilmektedir. Bununla birlikte, yinelenen (recurrent) kenarlar adı verilen ve bitişik zaman adımlarını birleştiren kenarlar model içerisinde döngüler oluşturabilmektedir. Zaman içinde bir düğümden kendisine bağlanan döngüler de bu duruma dahildir [50]. Bu şekilde ağ, dinamik zamansal davranış sergileyebilmektedir. İleri beslemeli sinir ağının aksine, RNN sıralı girişleri işleyebilmektedir. Kendi giriş belleğini girişlerin rastgele dizilerini işlemek için kullanmaktadır [51, 52]. Tekrarlayan sinir ağı yapısı Şekil 5.5'te verilmiştir.



Şekil 5.5. Tekrarlayan sinir ağı

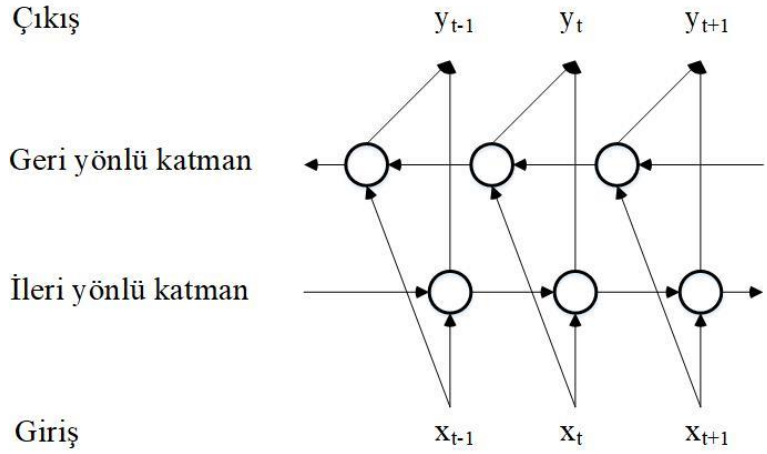
Tekrarlayan sinir ağındaki temel düşünce sıralı ve birbiriyle ilişkili bilgileri kullanabilmektir. Sadece ağa giren giriş örnekler değil daha önce zaman serisi içerisindeki giriş örneklerini de giriş olarak alabilir. Görüntü tabanlı verilerde tüm girişler birbirinden bağımsızdır, ancak doğal dil işleme gibi zaman değişkeni olan alanlar için bu durum farklıdır. RNN mimarisinin yinelenen olarak adlandırılmasının sebebi, her veri için aynı görevi önceki hesaplamalara bağlı olarak yerine getirmesidir. Her adım önceki adımlardaki veri ve çıkışları da değerlendirerek çıkışını hesaplamaktadır [46, 48]. Tekrarlayan sinir ağında bulunan döngünün zaman içindeki açılımı Şekil 5.6'da verilmiştir.



Şekil 5.6. Tekrarlayan bir sinir ağı ve hesaplamanın zaman içindeki açılımı

RNN, daha çok dil özelliklerini içeren problemlerde kullanılmakla birlikte arka arkaya meydana gelen bir dizi yapıda, bir sonrakinin tahmin edilmesini gerektiren problemler için başarılı bir şekilde uygulanmaktadır [46]. Metin işleme, metindeki bir sonraki karakteri tahmin etme gibi problemlerde yüksek başarı elde etmektedir [47].

En çok kullanılan RNN mimarilerinden birisi 1997 yılında Schuster ve Paliwal tarafından tanıtılan Çift Yönlü Tekrarlayan Sinir Ağıdır (Bidirectional RNN - BRNN). Bu mimaride, iki gizli düğüm katmanı vardır. Her iki gizli katman da giriş ve çıkışa bağlanmaktadır. Birincisi, geçmiş zamandan tekrarlayan bağlantılara sahip iken, ikincisinde tekrarlayan bağlantıların yönü ters çevrilerek aktivasyon dizi boyunca geriye doğru iletilmektedir [50]. Basit bir çift yönlü tekrarlayan sinir ağı yapısı Şekil 5.7'de sunulmuştur.



Şekil 5.7. Çift yönlü tekrarlayan sinir ağı

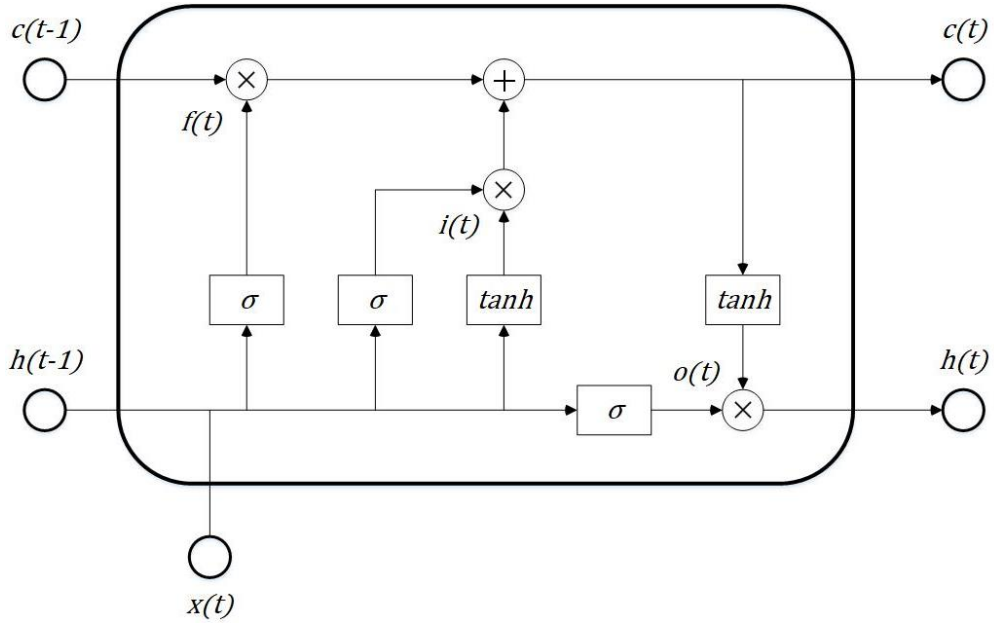
5.1.3. Uzun kısa süreli bellek

Derin öğrenme ailesinin önemli bir türü olan RNN, birçok makine öğrenmesi ve bilgisayarla görme uygulamalarında başarılı sonuçlar vermiştir [51]. RNN mimarilerinde önceki bilginin kullanımına dayalı yaklaşım bulunmaktadır. Ancak, uzun vadeli ardışık verilerle başa çıkmak için RNN'leri eğitmenin zor olduğu görülmüştür. Teorik olarak mümkün olan uzun-vadeli bağımlılıklar, pratikte büyük problemlere yol açmıştır. Bağlamlar arası boşluk arttığında modelin geçmişten gelen bilgiyi kullanarak tahminde bulunması oldukça zorlaşmaktadır [46, 48]. Bu sorunun üstesinden gelmek için daha çok sayıda ve farklı işleve sahip bileşenden oluşan bir tekrarlayan ağ tasarlanmıştır [51].

Uzun süreli bağımlılıkları öğrenebilecek özel bir RNN türü olarak LSTM önerilmiştir. LSTM ağları, Hochreiter ve Schmidhuber tarafından 1997 yılında tanıtılmıştır [48]. Bu model, gizli katmana sahip standart bir RNN ağına benzemektedir. Ancak, gizli katmandaki her sıradan düğüm bir bellek hücresi ile değiştirilmiştir [50]. Bellek hücreleri, önceki durumu ve giriş bilgisini tutmaktadır. Ağ mimarisi içerisinde yer alan bu hücreler, tutulması ve silinmesi gereken verilere karar vermektedir [46].

Örneğin, bir metinde, “Türkiye’de okudum.” cümlesi ile başlanıp başka birçok cümle geldikten sonra “Çok güzel Türkçe konuşurum.” cümlesinde “Türkçe” kelimesinin tahmin edilebilmesi için metnin en başındaki cümlede bulunan yer bilgisinin hafızada tutulması gerekmektedir. RNN, içinde bulunduğu cümleden yola çıkarak bir dil adı olacağını tahmin edebilmesine rağmen, bağlamlar arası mesafe uzun olduğundan hangi dil olduğunu tahmin etmekte zorlanmaktadır. Bu durumlar için geliştirilen LSTM, RNN ile benzer olmakla beraber gizli katmandaki sıradan düğümlerin yerine hafıza hücreleri kullanmaktadır. Böylece daha uzun süreli bilgi kullanımını sağlayabilmektedir.

LSTM ünitesinin ayrıntılı yapısı Şekil 5.8’de gösterilmektedir. LSTM ünitesinde giriş, unutma ve çıkış kapıları olmak üzere üç kontrol kapısı bulunmaktadır. Bloğun çıktısı sonraki iterasyonda bloğun girişine ve diğer kapılara tekrar bağlanmaktadır. Temel LSTM yapısında üç farklı harici giriş bulunmaktadır. Şekil 5.8’de $c(t-1)$ ile gösterilen giriş önceki hücre durumunu, $h(t-1)$ önceki gizli durumu, $x(t)$ ise mevcut girişi ifade etmektedir [53].



Şekil 5.8. LSTM mimarisi

Burada, $f(t)$ unutma kapısını, $i(t)$ giriş kapısını, $o(t)$ çıkış kapısını, σ sigmoid fonksiyonunu, $tanh$ hiperbolik tanjant fonksiyonunu, $c(t)$ mevcut hücre durumunu, $h(t)$ ise mevcut gizli durumu ifade etmektedir.

Hücre durumu, tahmin için kullanılacak anlamlı bilgileri sonraki hücrelere taşımaktadır. Taşınacak bilgilere kapılar aracılığı ile karar verilmektedir. Hangi bilgilerin unutulması gerektiğine unutma kapısında karar verilmektedir. Önceki hücrenin gizli durumu ve mevcut giriş sigmoid aktivasyon fonksiyonuna sokulmaktadır. Fonksiyonun sonucunda 0 değerine sahip olan bilgiler unutulmakta, 1 değerine sahip olanlar ise hücre durumu ile birlikte sonraki hücrelere aktarılmaktadır.

Hücre durum güncellemesi giriş kapısında gerçekleşmektedir. Önceki gizli durum ve mevcut girişin sigmoid fonksiyonu sonucunda elde edilen değere göre güncelleme yapılmaktadır. Bu kapıda aynı zamanda ağı düzenlemek için tanh aktivasyon fonksiyonu kullanılmaktadır. Bu iki fonksiyonun çıkışları çarpılarak hangi bilginin güncelleneceği belirlenmektedir. Çıkış kapısında ise sonraki hücrenin girişinde kullanılacak olan gizli durum belirlenmektedir. Bunun için, önceki hücreye ait gizli durum ve mevcut hücreye ait giriş bilgisi sigmoid fonksiyona sokulmakta ve elde edilen sonuç hücre durum bilgisinin tanh fonksiyonu sonucu ile çarpılmaktadır.

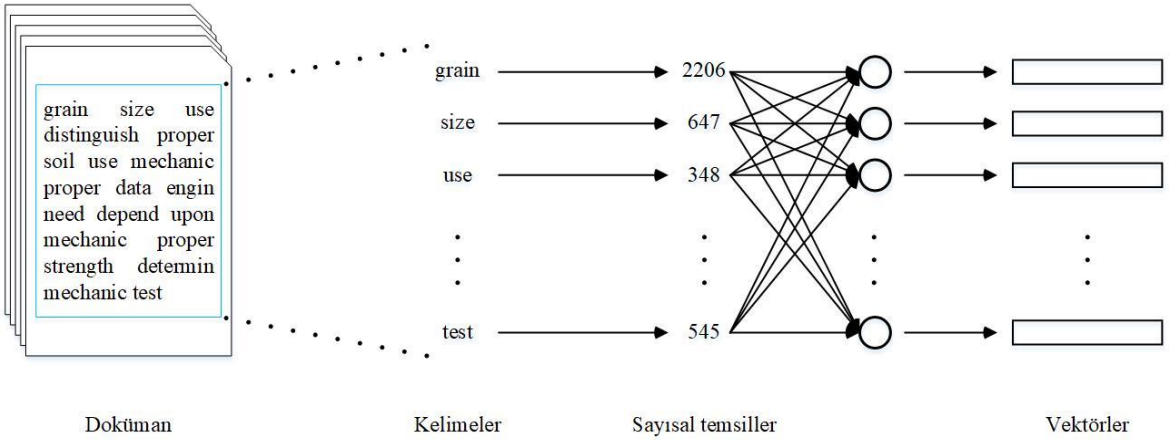
LSTM ayrıca sabit hata döngüsü, çıkış aktivasyon fonksiyonu ve gözetleme bağlantılarına sahiptir. Gözetleme bağlantıları ve unutma kapısı ilk geliştirilen mimaride bulunmamasına rağmen durumunu sıfırlamak ve kesin zamanlamaları, bağlantıları öğrenmeyi kolaylaştırmak için sonradan eklenmiştir. LSTM ağı, kapıların anahtarında geçici bellek uygulayarak unutulması ve hatırlanması gereken bilgileri belirlemekte [53] ve hücre durumuna bilgi ekleme veya çıkarma yapabilmektedir. Kapılar sigmoid sinir ağ katmanı ve matematiksel çarpımlardan oluşmaktadır. Böylelikle hangi verilerin hafızaya nasıl etki edeceği belirlenerek LSTM modelinin hafızası oluşturulmaktadır [49].

LSTM ağlarının özellikle her bir zaman adımı için birkaç katman bulundurduklarında geleneksel RNN'lere göre çok daha etkili olduğu görülmüştür [54]. LSTM mimarileri konuşma ve metin işleme konularında oldukça başarılı sonuçlar vermektedir [55, 56]. LSTM ağları, makine çevirisinde çok iyi performans gösteren kodlayıcı ve kod çözücü ağlar için de kullanılmaktadır [47].

5.2. Geliştirilen Model

Büyük boyutlu dokümanlar gerek işlem yükü ve süresi gerekse de başarı oranları açısından üzerinde çalışılması zor olan veri kümeleridir. Bu dokümanlar, boyutlarının bir sonucu olarak geniş kelime haznelerine de sahiptir. Bu durum, hem işlem sırasında veri boyutunu artırmakta hem de benzerlik ölçümlerinde zorluklara yol açmaktadır.

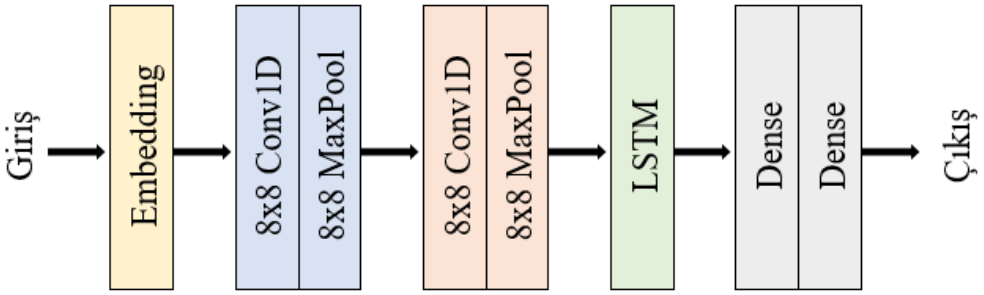
Bu çalışmada, gerekli önişlemlerden geçirilen dokümanlar öncelikle vektör haline getirilerek derin öğrenme modelinde kullanıma uygun hale getirilmiştir. Öncelikle tüm doküman koleksiyonu tokenize edilerek bütün dokümanlar kelimelere ayrılmıştır. Elde edilen kelimelere sayısal değerler verilmiştir. Her kelimenin sayısal karşılığı belirlendikten sonra dokümanlar içerdikleri kelimelerin sayısal karşılıklarının bütünü olarak ifade edilmiştir. Bu dokümanlara ait kelimeler daha sonra derin öğrenme modeli içerisinde birer vektör haline getirilmiştir. Dokümanların vektörlerle temsil edilme yöntemine ait blok şeması Şekil 5.9’da verilmiştir.



Şekil 5.9. Dokümanların vektörlerle ifade edilmesi

Vektörlerle temsil edilmiş dokümanlardan derin öğrenme yöntemleri kullanılarak özellik çıkarımı yapılmıştır. Bunun için CNN ve LSTM ağlarından yararlanılan hibrit bir model geliştirilmiştir. Modelde, giriş katmanı olarak Keras kütüphanesine ait embedding katmanı kullanılmıştır. Embedding katmanının çıkışı sırasıyla CNN ve LSTM katmanlarına bağlanmıştır.

Modelin CNN kısmında iki evrişim katmanı kullanılmıştır. 50 filtreye sahip olan bu evrişim katmanlarının her birinden sonra havuzlama katmanı yerleştirilmiştir. LSTM katmanının çıkışı ise Rectifier Linear Unit (ReLU) aktivasyon fonksiyonuna sahip, art arda yerleştirilmiş dense katmanlarına iletilmiştir. Geliştirilen modelin katmanları Şekil 5.10'da sunulmuştur.



Şekil 5.10. Geliştirilen modelin katmanları

Veri kümesindeki dokümanlarda bulunan toplam farklı sözcük sayısı 553.698'dir. Bu nedenle, embedding katmanının giriş boyutu bu sayıya eşitlenmiştir. Çıkış boyutu ise 50 olarak ayarlanmıştır. Yani her bir sözcüğün temsil edildiği vektör boyutu 50'dir. Evrişim katmanlarında aktivasyon fonksiyonu olarak Eş. 5.1'de verilen ReLU kullanılmıştır.

$$R(x) = max(0, x) \tag{5.1}$$

Burada, x nöron girişi, $max()$ en büyük değeri bulma fonksiyonu, $R(x)$ ise ReLU fonksiyonunun hesapladığı değerdir. LSTM katmanının aktivasyon fonksiyonu ise Sigmoid'dir. Sigmoid fonksiyonuna ait eşitlik Eş. 5.2'de verilmiştir.

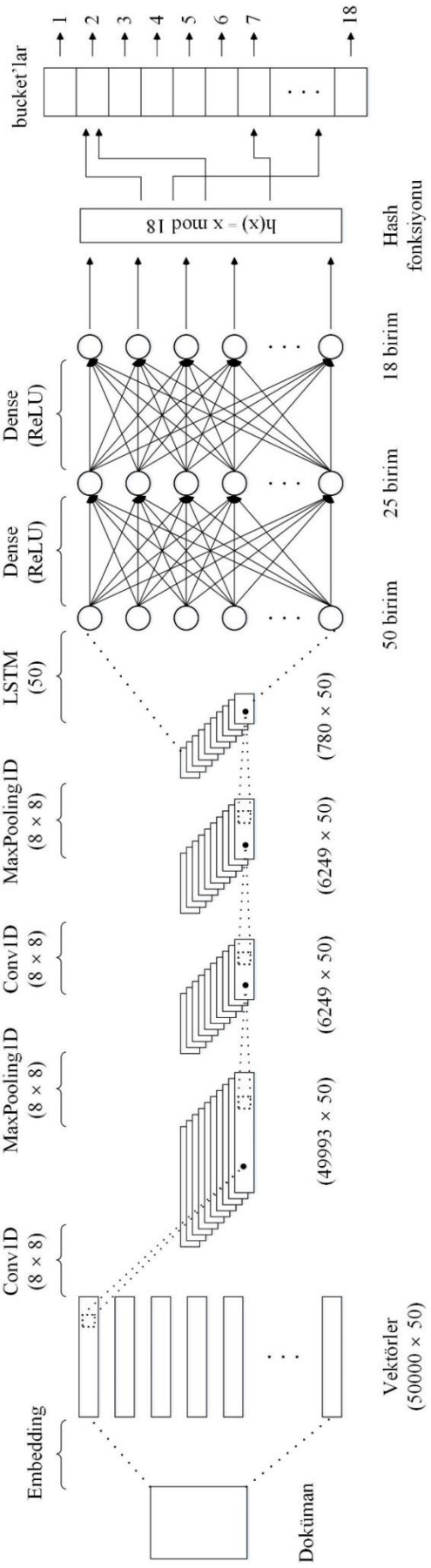
$$S(x) = \frac{1}{1 + e^{-x}} \tag{5.2}$$

Burada, x nöron girişi, e Euler sayısı, $S(x)$ ise Sigmoid fonksiyonu tarafından hesaplanan değerdir.

LSTM katmanından sonra gelen iki dense katmanı ReLU aktivasyonuna sahiptir. Bu katmanlar sırasıyla 25 ve 18 birim kullanılarak oluşturulmuştur. Derin öğrenme modeli toplamda 27.746.943 eğitilebilir parametreden oluşmaktadır. Modele ait katmanların ayrıntılı bilgisi Çizelge 5.1'de sunulmuştur. Geliştirilen derin öğrenme modelinin blok şeması ise Şekil 5.11'de verilmiştir.

Çizelge 5.1. Geliştirilen modelin katmanlarına ait ayrıntılar

Katman	Çıkış Şekli	Parametre Sayısı
Embedding	(None, 50000, 50)	27.684.900
Conv1D	(None, 49993, 50)	20.050
MaxPooling1D	(None, 6249, 50)	0
Dropout	(None, 6249, 50)	0
Conv1D_1	(None, 6242, 50)	20.050
MaxPooling1D_1	(None, 780, 50)	0
LSTM	(None, 50)	20.200
Dense	(None, 25)	1.275
Dense_1	(None, 18)	468
Toplam parametre sayısı:		27.746.943

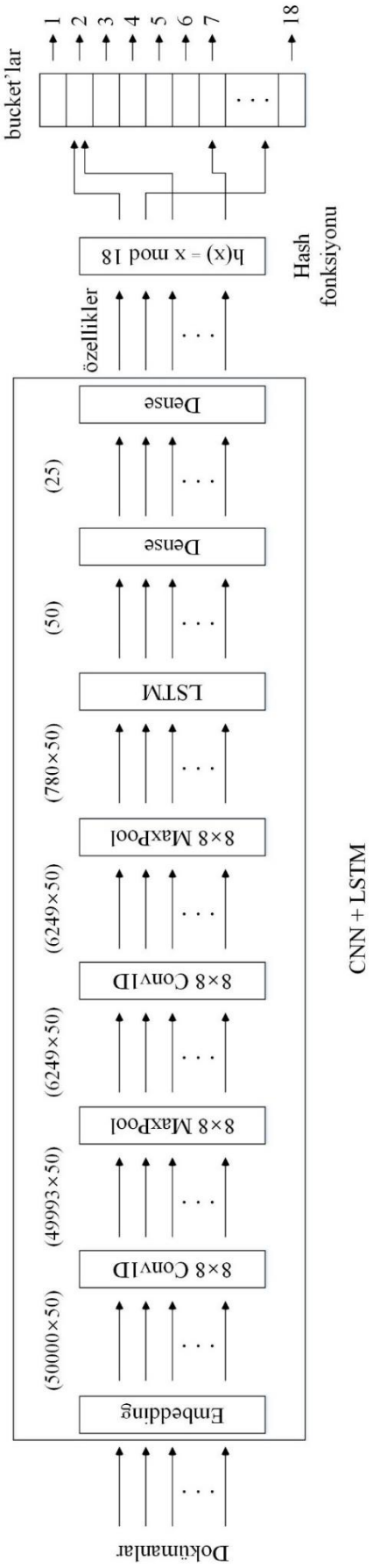


Şekil 5.11. Geliştirilen modelin blok şeması

Derin öğrenme modelinin eğitimi tamamlandıktan sonra modelin son katmanına (Dense_1) ait çıktılar kullanılarak modelden özellik çıkarımı gerçekleştirilmiştir. Elde edilen bu özellikler Eş. 5.3'te verilen hash fonksiyonundan geçirilmiş ve dokümanlar bucket'lara atanmıştır.

$$h(x) = x \; mod \; 18 \tag{5.3}$$

Bucket'lara ayrılan dokümanlar küme özellikleri açısından değerlendirilmiştir. Çalışmada kümeleme işlemi için tasarlanan algoritma Şekil 5.12'de görsel olarak sunulmuştur.



Şekil 5.12. Kümeleme algoritması

6. DENEYSEL SONUÇLAR

Bu çalışmadaki deneysel sonuçlar, Python programlama dili kullanılarak Jupyter Notebook ortamında geliştirilen uygulama ile elde edilmiştir. Kullanılan veri kümesi ve elde edilen deneysel sonuçlar bu bölümde sunulmuştur.

6.1. Veri Kümesi

Çalışmada geliştirilen derin öğrenme tabanlı model, Kaggle platformundan elde edilmiş 'COVID19-Engineering-Books-NLP-Dataset' isimli bir veri kümesi [57] üzerinde test edilmiştir. Kullanılan veri kümesi Springer'daki 386 İngilizce ders kitabını içeren bir kitap koleksiyonudur. Dokümanlar (kitaplar), toplam 7.61 GB boyutunda ve pdf formatındadır. Ayrıca, bu kitaplara ait meta verinin bulunduğu bir Excel dosyası içermektedir. Bu dosyada her kitabın adı, yazarları, basım tarihi, kitabın ilgili olduğu bazı ana ve alt alanlar gibi bilgiler sunulmuştur. Çalışmada dokümanlar bir pdf çıkarıcı kullanılarak okunmuş ve önişlemler yapıldıktan sonra txt formatında kaydedilmiştir. Veri kümesi hakkında detaylı bilgi vermek amacıyla en çok kelime içeren, en az kelime içeren, en büyük boyutlu, en küçük boyutlu ve farklı alanlardan toplam 10 dokümana ait detaylar Çizelge 6.1'de sunulmuştur.

Çizelge 6.1. Veri kümesinden 10 örnek doküman

Kitap Adı	Kelime Sayıları		Özellikler	
	Toplam	Tekil	Boyut (MB)	Sayfa Sayısı
The ASCRS Textbook of Colon and Rectal Surgery	514641	30782	77	1269
Handbook of Biological Confocal Microscopy	449716	22711	458	1014
Advanced Organic Chemistry	219965	14088	58	1213
Clinical Neuroanatomy	193271	10814	125	704
Algebra	142157	9974	66	923
Robotics, Vision and Control	108987	9285	153	572
Plate Tectonics	85445	6926	152	214
An Introduction to Soil Mechanics	57057	3252	82	407
Epidemiological Research: Terms and Concepts	40612	3998	2	181
Modelling Computing Systems	6679	844	15	507

Veri kümesindeki dokümanların haricinde sağlanan meta veride kitapların yazarları, baskı sayıları, eğitim düzeyi ve konu dağılımları gibi bilgiler paylaşılmıştır. Çizelge 6.1'de verilen örnek dokümanların konu başlıkları Çizelge 6.2'de sunulmuştur.

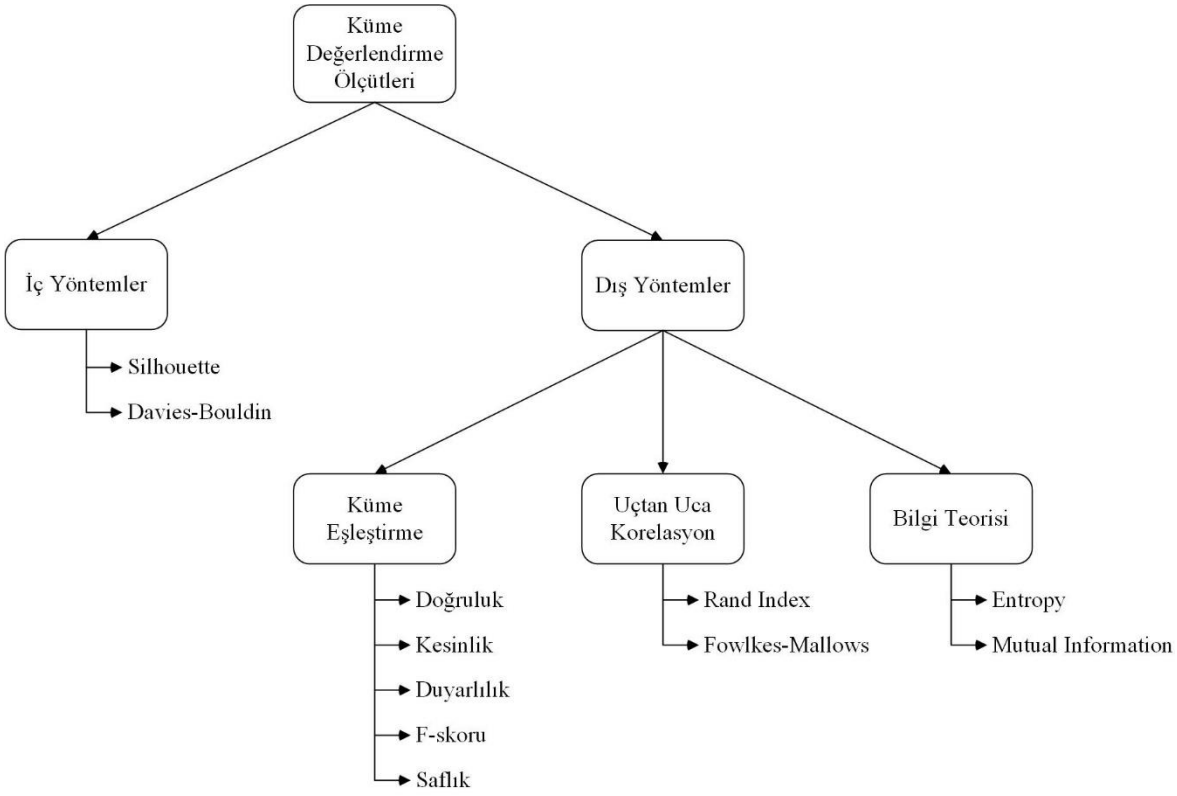
Çizelge 6.2. Örnek dokümanlara ait konular

Kitap Adı	Konular
The ASCRS Textbook of Colon and Rectal Surgery	Medicine and Public Health, Colorectal Surgery, General Surgery, Surgical Oncology
Handbook of Biological Confocal Microscopy	Life Sciences, Biochemistry, General, Biotechnology
Advanced Organic Chemistry	Chemistry, Organic Chemistry, Physical Chemistry, Medicinal Chemistry
Clinical Neuroanatomy	Psychology, Neuropsychology, Psychiatry, Anatomy, Psychology, General
Algebra	Mathematics, Algebra, Commutative Rings and Algebras, Linear and Multilinear Algebras, Matrix Theory, Associative Rings and Algebras, Group Theory and Generalizations
Robotics, Vision and Control	Engineering, Robotics and Automation, Control, Image Processing and Computer Vision, Signal, Image and Speech Processing, Cognitive Psychology
Plate Tectonics	Earth Sciences, Geology, Structural Geology, Planetology
An Introduction to Soil Mechanics	Engineering; Civil Engineering, Hydrogeology, Soil Science and Conservation, Geotechnical Engineering and Applied Earth Sciences
Epidemiological Research: Terms and Concepts	Biomedicine, General, Epidemiology, Public Health, Medicine/Public Health, Biometrics, Biostatistics
Modelling Computing Systems	Computer Science, Logics and Meanings of Programs, Mathematical Logic and Formal Languages, Discrete Mathematics in Computer Science, Math Applications in Computer Science

Veri kümesinde sunulan bu konular arasından her bir doküman için ana konu başlığı belirlenerek tek bir etiket oluşturulmuştur. Bu etiketler derin öğrenme modelinin eğitiminde kullanılmıştır.

6.2. Değerlendirme Ölçütleri

Bu yüksek lisans tez çalışmasında literatürde yaygın olarak kullanılmakta olan değerlendirme ölçütleri kullanılmıştır. Bu çalışmada geliştirilen modelin başarısı Şekil 6.1'de hiyerarşik olarak gösterilmiş olan değerlendirme ölçütlerine göre test edilmiştir [58].



Şekil 6.1. Değerlendirme ölçütlerinin sınıflandırması

Doğruluk (accuracy), yapılan tahminin hangi oranda doğru olduğunu göstermektedir. Bu oran Eş. 6.1'de verildiği şekilde hesaplanabilmektedir.

$$\text{Doğruluk} = \frac{\text{Doğru tahmin sayısı}}{\text{Toplam tahmin sayısı}} \tag{6.1}$$

Doğruluk değerinin hesaplanma şekli matematiksel olarak, Eş. 6.2'de daha detaylı bir şekilde sunulmuştur.

$$\text{Doğruluk} = \frac{\text{DP} + \text{DN}}{\text{DP} + \text{DN} + \text{YP} + \text{YN}} \tag{6.2}$$

Burada, DP doğru pozitif sayısını, DN doğru negatif sayısını, YP yanlış pozitif sayısını ve YN yanlış negatif sayısını göstermektedir.

Kesinlik (precision) ölçütü, pozitif olarak tahmin edilenlerin ne kadarının gerçekte pozitif olduğunu belirtmektedir. Matematiksel olarak, Eş. 6.3'te sunulduğu gibi, doğru pozitif sayısının toplam pozitif sayısına oranı alınarak hesaplanmaktadır.

$$\text{Kesinlik} = \frac{\text{DP}}{\text{DP} + \text{YP}} \tag{6.3}$$

Duyarlılık (recall), pozitiflerin hangi oranda pozitif olarak tahmin edilebildiğini göstermektedir. Bunun için doğru pozitiflerin, doğru pozitifler ve pozitif olması gereken ancak negatif olarak tahmin edilen yanlış negatiflerin toplamına oranı alınmaktadır. Bu oran Eş. 6.4'te verilmiştir.

$$\text{Duyarlılık} = \frac{\text{DP}}{\text{DP} + \text{YN}} \tag{6.4}$$

F1-skoru ise, Eş. 6.5'te sunulduğu üzere, kesinlik ve duyarlılık değerlerinin harmonik ortalamasına eşittir.

$$\text{F1} - \text{skoru} = 2 \times \frac{\text{Kesinlik} \times \text{Duyarlılık}}{\text{Kesinlik} + \text{Duyarlılık}} \tag{6.5}$$

Saflık (purity), etiketli verilere dayanarak kümenin kalitesini ölçen bir yöntemdir. Bu yöntemde her kümenin sadece tek bir sınıfa ait verileri bulundurup bulundurmadığı kontrol edilmektedir. 1 değeri tüm verilerin mevcut etikete göre mükemmel şekilde kümelendiği anlamına gelmektedir. 0 ise bir küme içerisinde farklı sınıflardan üyeler bulunduğunu göstermektedir. Saflık değeri Eş. 6.6'da verildiği şekilde hesaplanmaktadır.

$$\text{Saflık} = \frac{1}{N} \sum_{i-1}^{k} max_j \left| C_i \cap t_j \right| \tag{6.6}$$

Burada N veri sayısını, k küme sayısını, C_i i sayılı kümeyi ve t_j C_i kümesinde bulunan aynı etiketteki verilerin en büyük sayı değerini ifade etmektedir [59].

Rand Index (RI), iki veri kümesi arasındaki benzerliğin doğruluk ile ilgili bir ölçüsüdür. Doğruluk değerinin hesaplanmasında kullanılan yöntemden faydalanmaktadır. Ancak, bu işlemi kümelerdeki ikili elemanları göz önüne alarak yapmaktadır. Aynı kümede olması gerektiği halde olmayan çiftler YN, farklı kümelerde olmaları gerektiği halde aynı kümeye yerleştirilen çiftler ise YP olarak değerlendirilmektedir. *RI* aşağıdaki eşitlikle hesaplanmaktadır.

$$RI(X,Y) = \frac{a+b}{a+b+c+d} \tag{6.7}$$

Burada, a X ve Y'de aynı alt kümede bulunan eleman çiftlerinin sayısını, b hem X hem de Y'de farklı alt kümelerde bulunan eleman çiftlerinin sayısını, c X'te aynı alt kümede olup Y'de farklı alt kümede olan çiftlerin sayısını, d ise X'te farklı alt kümede olup Y'de aynı alt kümede olan çiftlerin sayısını ifade etmektedir. Adjusted Rand Index (ARI), *RI* için beklenen değere göre ayarlamaktadır. *ARI*, Eş. 6.8'de verildiği şekilde hesaplanabilmektedir.

$$ARI(X,Y) = \frac{RI(X,Y) - E\{RI(X,Y)\}}{\max\{RI(X,Y)\} - E\{RI(X,Y)\}} \tag{6.8}$$

Burada $E\{RI(X,Y)\}$, $RI(X,Y)$'nin beklenen değerini ifade etmektedir. *ARI*, 0 ve 1 arasında değer almaktadır. 1 değeri kümelemenin ideal ile aynı olduğunu, 0 değeri ise kümelemenin rastgele olduğunu göstermektedir [24, 60].

Mutual Information (MI), iki ayrı rastgele değişken arasındaki karşılıklı bağımlılığın bir ölçüsüdür. Bir ayrık rastgele değişkendeki belirsizliğin, diğer değişkenin bilgisi göz önüne alındığındaki azalışını hesaplamaktadır. Yüksek *MI*, belirsizlikte büyük bir azalma olduğunu göstermektedir. Ortak olasılık dağılımı $p(x, y)$ olan X ve Y değişkenleri için *MI* değerinin hesaplanması Eş. 6.9'da verilmiştir.

$$MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{6.9}$$

NMI, *MI* ölçütünün 0 ile 1 arasındaki değerleri alacak şekilde normalleştirilmiş halidir. 0 değişkenler arasında karşılıklı bilgi olmadığını, 1 ise ilişkili olduklarını göstermektedir. *NMI* değerinin hesaplanmasında Eş. 6.10'da verilen entropi kullanılmaktadır.

$$H(X) = -\sum_{x \in X} p(x) \log(p(x)) \tag{6.10}$$

NMI, *MI* değeri kullanılarak *H(X)* ve *H(Y)* marjinal entropiler olmak üzere Eş. 6.11'de sunulduğu şekilde hesaplanmaktadır.

$$NMI(X,Y) = \frac{MI(X,Y)}{\sqrt{H(X)H(Y)}} \tag{6.11}$$

AMI ise *MI* ölçütünün bir başka türevi olarak yine 0 ile 1 arasındaki değerleri almaktadır. 1 birebir bölünmeyi temsil etmekte ve kullanılan bölünme sayısına göre ayarlanmaktadır [24]. *AMI* aşağıdaki eşitlikle hesaplanabilmektedir.

$$AMI(X,Y) = \frac{MI(X,Y) - E\{MI(X,Y)\}}{\max\{H(X),H(Y)\} - E\{MI(X,Y)\}} \tag{6.12}$$

Burada $E\{MI(X,Y)\}$, $MI(X,Y)$'nin beklenen değerini $H(X)$ ve $H(Y)$ ise entropi değerlerini ifade etmektedir.

Fowlkes-Mallows (FM) ölçütü, gerçekleştirilmiş iki farklı kümeleme işleminin benzerliğini ölçmektedir. Bunlardan birinin beklenen kümeleme sonucu olduğu varsayılmaktadır. Eş. 6.13'te verildiği gibi, kesinlik ve duyarlılık değerlerinin geometrik ortalaması alınarak hesaplanmaktadır.

$$FM = \frac{TP}{\sqrt{(TP+FP)(TP+FN)}} = \sqrt{\frac{TP}{TP+FP}\,\frac{TP}{TP+FN}} = \sqrt{\text{Kesinlik x Duyarlılık}} \tag{6.13}$$

Burada, *TP* doğru pozitif sayısını, *FP* yanlış pozitif sayısını, *TN* doğru negatif sayısını ve *FN* yanlış negatif sayısını ifade etmektedir.

Yöntemler arasında yapılan karşılaştırmaların daha yararlı olabilmesi için değerlendirme ölçütlerinin sabit sınır ve temel değere sahip olması gerekmektedir. *MI* ve *RI* gibi sabit bir temele sahip olmayan ölçütler, farklı küme sayılarına sahip kümeleme yöntemlerinde karşılaştırılamamaktadır. Bu nedenle bu ölçütler için düzeltilmiş türevler önerilmiştir. Literatürde, karşılaştırılabilir değerlendirme sağlamak için en iyi ölçütlerin *AMI* ve *ARI* olduğu belirtilmektedir [24]. Veri kümesi dengesiz ve küçük kümelerden oluştuğunda *AMI*, dengeli ve büyük kümelerden oluştuğunda ise *ARI* tercih edilebileceği belirtilmektedir [61].

Dış değerlendirme ölçütlerinin yanı sıra iç değerlendirme ölçütlerinden literatürde yaygın olarak kullanılmakta olan Silhouette ve Davies-Bouldin ölçütlerine yer verilmiştir. Bu ölçütler, kohezyon (cohesion) yani küme içi yakınlığı ve kümeler arası uzaklığı ölçerek kümenin kalitesini belirlemektedir [62]. Silhouette ölçütünde kohezyon, aynı kümedeki tüm noktalar arasındaki mesafeye göre, kümeler arası uzaklık ise en yakın komşuya olan uzaklığa göre ölçülmektedir. Silhouette hesaplama yöntemi Eş. 6.14'te verilmiştir.

$$S = \frac{1}{\mathrm{N}} \sum_{i=1}^{N} \sum_{j=1}^{C_i} \frac{b_j^i - a_j^i}{\max\{a_j^i, b_j^i\}} \tag{6.14}$$

Burada a_j^i, C_i kümesindeki j noktası ile aynı kümedeki diğer noktalar arasındaki ortalama mesafeyi, b_j^i ise C_i kümesindeki j noktası ile diğer tüm noktalar arasındaki minimum ortalama mesafeyi temsil etmektedir. Bu ölçüt, her nokta için kümeleme kalitesini göstermekte ve [-1, 1] aralığında değerler alabilmektedir. 1'e yakın olan *S* değeri, puanların iyi dağıtıldığını göstermektedir. Öte yandan, *S* değeri -1'e yakınlaştıkça belirtilen noktanın farklı bir kümeye atanması gerektiği anlaşılmaktadır.

Davies-Bouldin (DB) ölçütü, kohezyonu bir kümedeki noktaların küme merkezlerine olan uzaklığına göre tahmin etmektedir. Uzaklık ise küme merkezleri arasındaki uzaklığa göre tahmin edilmektedir. Bu ölçüt farklı varyantlara sahip olmakla birlikte yaygın kullanılan varyantlarından biri Eş. 6.15'te tanımlanmıştır.

$$DB = \frac{1}{\mathrm{N}} \sum_{i=1}^{N} max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \tag{6.15}$$

Burada N toplam küme sayısı, σ_i ise i kümesindeki tüm noktaların c_i küme merkezine olan ortalama mesafesidir. Kümeleme kalitesinin daha düşük DB değerleri için daha iyi olduğu kabul edilmektedir. Bu ölçüt için üst sınır yoktur ancak minimum değeri sıfırdır.

6.3. Yapılan Önişlemler

386 tane pdf formatındaki kitaptan oluşan veri kümesini kümeleme işlemine hazır hale getirmek, yanlış pozitif ve yanlış negatif sayısını azaltmak için önişlemler uygulanmıştır. Öncelikle dokümanlarda gerekli önişlemlerin gerçekleştirilebilmesi için bir pdf çıkarıcı yardımı ile dokümanlar metin dosyası haline dönüştürülmüştür.

İşlenebilir hale getirilen dokümanlar, daha yüksek kümeleme başarısı ve doğru benzerlik sonuçları elde edebilmek amacıyla, çoğu dil işleme çalışmasında uygulanmakta olan bazı önişlemlerden geçirilmiştir. Bu işlemler aşağıda özetlenmiştir:

- Dokümanların sadece metin kısımları alınarak şekil, tablo ve denklem gibi diğer kısımlar kaldırılmıştır.
- Metindeki anlamsal içeriğe katkısı olmayan noktalama işaretleri ve etkisiz kelimeler temizlenmiştir.
- Tüm metin küçük harfe çevrilmiştir.
- Metindeki gereksiz beyaz boşluklar ve işaretler kaldırılmıştır.
- Metin üzerinde gövdeleme işlemi gerçekleştirilmiştir.

Etkisiz kelimeler listesi, İngilizce dilindeki yaygın kullanımı ve başarısından dolayı Natural Language ToolKit (NLTK) kütüphanesi kullanılarak elde edilmiştir. Gövdeleme işlemi için de yine NLTK kütüphanesine ait Snowball stemmer kullanılmıştır. Bu yöntem, literatürde başarılı bir şekilde sıklıkla kullanılmakta olan Porter gövdeleme yönteminin geliştirilmiş versiyonu olarak bazı problemleri ele alan ek kurallar tanımlamaktadır. Bu işlemler tamamlandıktan sonra dokümanlar metin belgesi olarak kaydedilerek geliştirilen modele giriş olarak kullanıma hazır hale getirilmiştir.

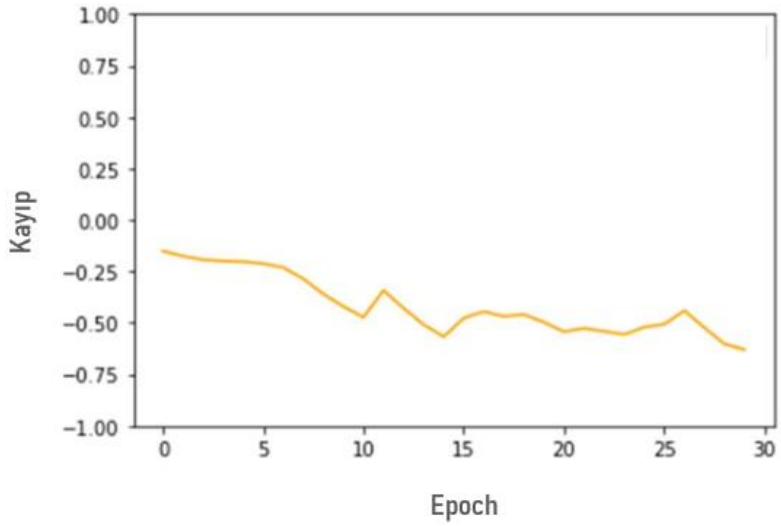
6.4. Deneysel Sonuçlar

Geliştirilen derin öğrenme modeli, önişlemlerden geçirilmiş ve vektörleştirilmiş dokümanlar ile eğitilmiştir. Modelden elde edilen özellikler hash fonksiyonu kullanılarak normalize edilmiş ve kümeler oluşturulmuştur. Bunun için, 386 doküman üzerinde yapılan deneylerde %66 oranında kümeleme başarısı sağlanmıştır. Toplam 18 küme elde edilmiştir. Modelin eğitimi sırasında kayıp fonksiyonu olarak [-1, 1] aralığında değerler alan Eş. 6.9'da verilen kosinüs benzerliğinden yararlanılmıştır.

$$SIM([x_1, \dots, x_n], [y_1, \dots, y_n]) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2}\ \sqrt{\sum_{i=1}^{n} {y_i}^2}} \tag{6.9}$$

Burada, $[x_1, x_2, \dots, x_n]$ ve $[y_1, y_2, \dots, y_n]$ hedef ve tahmin vektörlerini, n bu vektörlerin boyutunu göstermektedir.

Eğitim süresince kayıp değerinin değişimini gösteren grafik Şekil 6.2'de sunulmuştur. Model, eğitim sonucunda -0,63 kayıp değerine ulaşmıştır. Optimizasyon algoritması olarak ise Adam kullanılmıştır.



Şekil 6.2. Kayıp değeri grafiği

Modelin eğitimi -0,15 kayıp değeri ile başlamıştır. Bu değer Şekil 6.1'de verilen grafikte de görüldüğü üzere aralarda yaşanan iniş çıkışlar ile birlikte genel itibariyle kademeli olarak düşmüştür. 30 epoch ile tamamlanan eğitimin sonunda -0,63 kayıp değerine ulaşılmıştır.

Optimum küme sayısının bulunması için farklı küme sayılarının kullanıldığı deneyler gerçekleştirilmiştir. Farklı küme sayıları için elde edilen deneysel sonuçlar Çizelge 6.3'te sunulmuştur.

Çizelge 6.3. Farklı küme sayıları için elde edilen sonuçlar

Ölçüt	10 Küme	15 Küme	18 Küme	20 Küme	30 Küme
Saflık	0,27	0,29	*0,61*	0,45	0,45
ARI	0,14	0,15	*0,34*	0,23	0,21
NMI	0,37	0,42	*0,65*	0,54	0,54
AMI	0,34	0,39	*0,59*	0,49	0,48
Fowlkes-Mallows	0,31	0,36	*0,42*	0,36	0,37
Davies-Bouldin	3,58	2,30	*0,95*	2,49	2,75
Silhouette	0,04	0,22	*0,81*	0,13	0,10

Yapılan bu deneylerde optimum küme sayısının 18 olduğu sonucuna varılmıştır. Çizelge 6.3'te sunulduğu üzere 10 kümeden 18 kümeye doğru çıktıkça küme kaliteleri artmıştır. Küme sayısı artırılmaya devam edildiğinde 18 kümeden sonra belirgin bir değişim olmuştur. Daha sonra 30 kümeye kadar küme kaliteleri yaklaşık olarak aynı kalmıştır. 18 küme ile yapılan kümeleme işlemi, küme kalitesi için kullanılan tüm ölçütlerde daha başarılı sonuçlar vermiştir.

Kümeleme başarısı farklı ölçütler kullanılarak değerlendirilmiştir. Her bir kümede bulunan ve bulunması gereken dokümanlar göz önüne alınarak elde edilen başarı ölçütleri Çizelge 6.4'te ayrıntılı olarak sunulmuştur. Çizelgenin son sütununda verilen destek değeri, her bir kümede kaç tane eleman bulunduğunu göstermektedir.

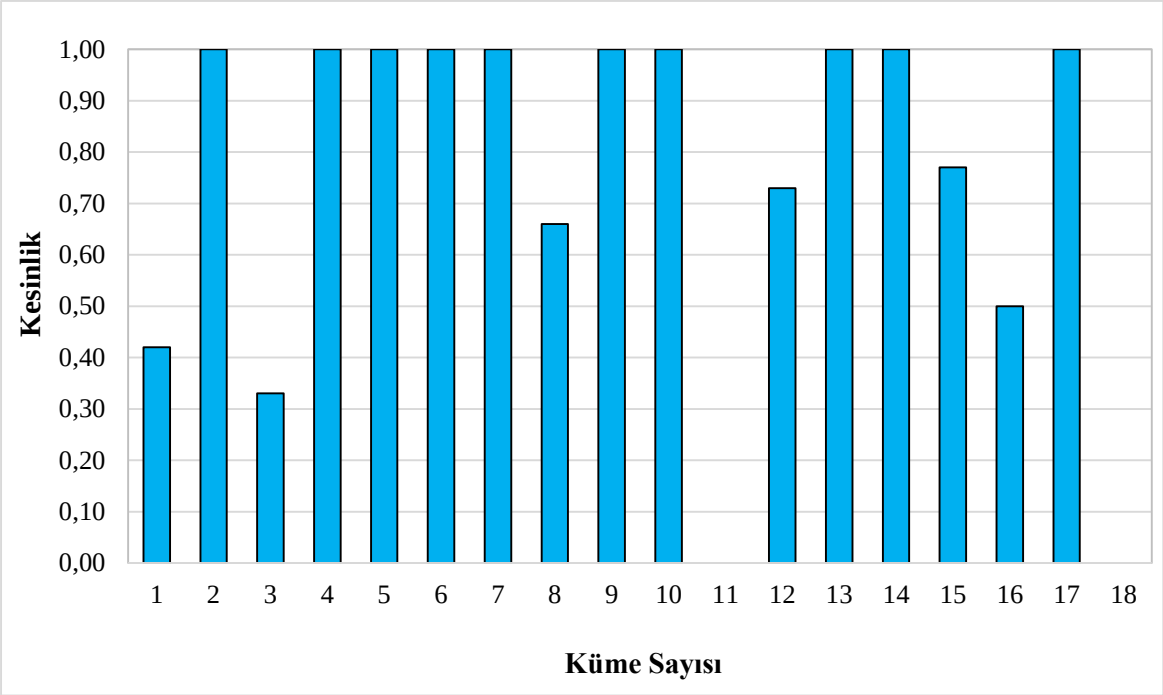
Çizelge 6.4. Geliştirilen modelden elde edilen deneysel sonuçlar

Küme #	Kesinlik	Duyarlılık	F1-skoru	Destek
1	0,42	0,67	0,51	39
2	1,00	0,73	*0,84*	26
3	0,33	*0,97*	0,49	34
4	1,00	0,58	0,74	12
5	1,00	0,79	*0,88*	24

Çizelge 6.4. (devam) Geliştirilen modelden elde edilen deneysel sonuçlar

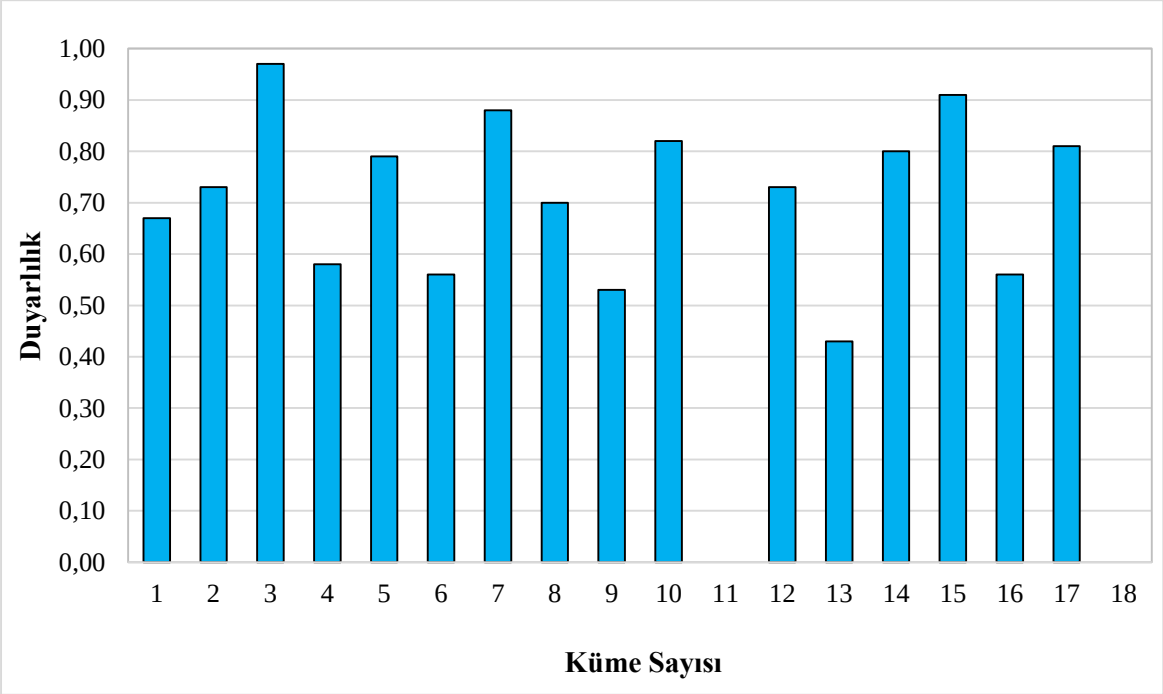
Küme #	Kesinlik	Duyarlılık	F1-skoru	Destek
6	1,00	0,56	0,72	41
7	1,00	0,88	*0,93*	16
8	0,66	0,70	0,68	50
9	1,00	0,53	0,69	34
10	1,00	0,82	*0,90*	11
11	0,00	0,00	*0,00*	26
12	0,73	0,73	0,73	11
13	1,00	0,43	0,60	7
14	1,00	0,80	*0,89*	10
15	0,77	0,91	*0,83*	11
16	0,50	0,56	0,53	9
17	1,00	0,81	*0,89*	21
18	0,00	0,00	*0,00*	4

Çizelge 6.4'te her bir küme için ayrı ayrı sunulan kesinlik, duyarlılık ve F1-skorları incelendiğinde öncelikle 2, 5, 7, 10, 14, 15 ve 17 numaralı kümelerde elde edilen yüksek başarılar fark edilmektedir. Bu kümelerde hem kesinlik hem de duyarlılık için tatmin edici sonuçlar elde edilmiş ve dolayısıyla bu sonuçlar F1-skorlarına da yansımıştır. 4, 6, 9 ve 13 numaralı kümelerde ise kesinlik değerleri 1,00 iken duyarlılık değerlerindeki daha düşük başarılardan dolayı F1-skorları ortalama olarak 0,70 civarında kalmıştır. 3 numaralı kümede ise tam ters bir şekilde 0,97 duyarlılık sağlanmış ancak 0,33 değerindeki kesinlik bu kümenin F1-skorunu düşürmüştür. Matematik kategorisine ait bu kümede neredeyse tüm ilgili kitapları bulunduğu ancak istatistik gibi matematik içerikli kategorilere ait bazı kitapların da bu kümeye dahil edilmesinden dolayı kesinlik değerinin düştüğü görülmüştür. Diğer yandan skorlarının 0,00 olması sebebiyle dikkat çeken 11 ve 18 numaralı kümeler incelendiğinde 11 numaralı kümenin tıp alanına ait kitaplardan oluştuğu fark edilmiştir. Bu kitapların çoğunlukla beyin cerrahisi üzerine olduğu ve içerdikleri terimlerden kaynaklı olarak bilgisayar bilimlerindeki yapay zeka kitaplarından ayırt edilmede zorlanıldığı tespit edilmiştir. 18 numaralı kümede ise bu kategoriye ait çok az sayıda kitap bulunmasından dolayı zorlanıldığı düşünülmektedir. Küme bazlı kesinlik değerleri Şekil 6.3'te karşılaştırılarak sunulmuştur.



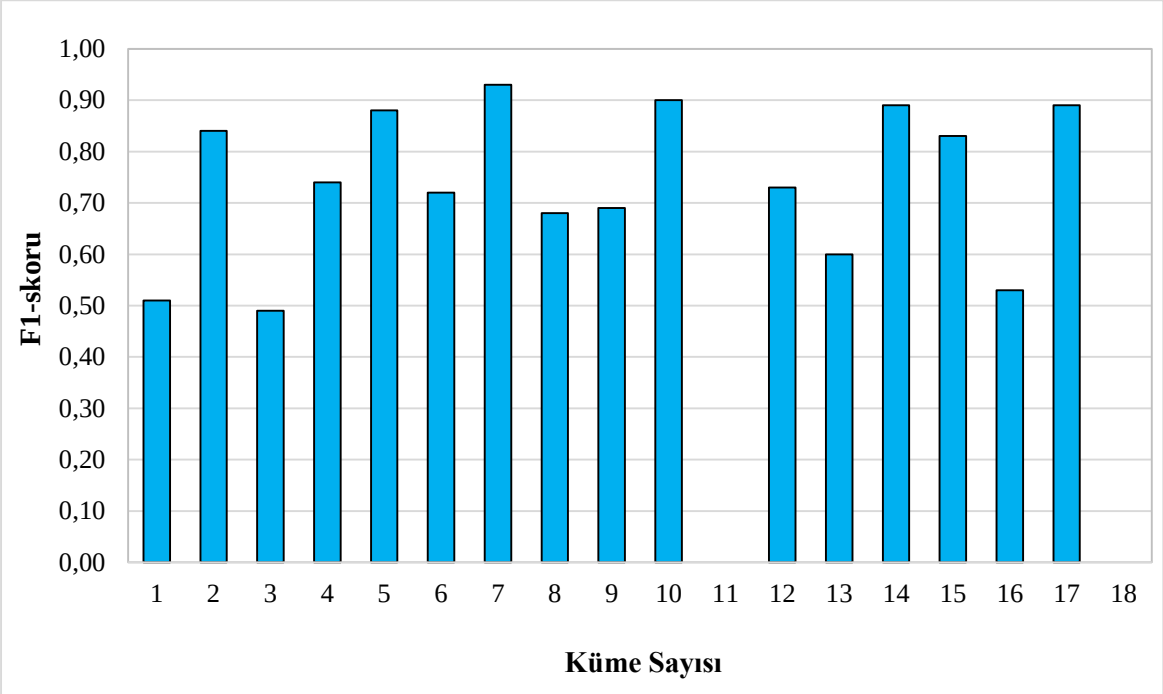
Şekil 6.3. Kümelerin kesinlik değerleri

Şekil 6.3'te verilen grafik incelendiğinde çoğu kümenin 0,70 ve üzeri hatta 1,00 değerine sahip olduğu görülmektedir. Ancak 1, 3, 11, 16 ve 18 numaralı kümeler diğerlerinden daha düşük skorlara sahiptir. Küme bazlı duyarlılık değerleri ise Şekil 6.4'te sunulmuştur.



Şekil 6.4. Kümelerin duyarlılık değerleri

Şekil 6.4'te verilen grafik incelendiğinde kümelere ait duyarlılık değerlerinin kesinlik değerlerinden daha düşük olduğu görülmektedir. 1, 3 ve 8 numaralı kümeler gibi kesinlik değeri düşük kümelerin duyarlılık değerleri yüksektir. Çoğu küme 0,70 ve üzeri değere sahip olmakla birlikte 11, 13 ve 18 numaralı gibi bazı kümeler diğerlerinden daha düşük skorlara sahiptir. Kesinlik ve duyarlılık ölçütlerinden elde edilen F1-skorları Şekil 6.5'te küme bazlı karşılaştırılarak sunulmuştur.



Şekil 6.5. Kümelerin F1-skorları

Şekil 6.5'te elde edilen 18 kümeye ait F1-skorları incelendiğinde çoğu kümenin 0,70 ve üzeri bir değere sahip olduğu görülmektedir. Ancak az sayıdaki bazı kümeler diğerlerinden daha düşük skorlara sahiptir. Özellikle 11 ve 18 numaralı kümelerin doğru bir şekilde ayırt edilemediği fark edilmektedir.

Elde edilen başarı oranlarına ait nedenlerin daha detaylı bir şekilde irdelenebilmesi için küme bazlı farklı ve ortak kelime sayıları incelenmiştir. Her kümedeki tekil kelime sayıları ve içerdiği dokümanlardaki ortak kelime sayıları küme bazlı olarak Çizelge 6.5'te sunulmuştur.

Çizelge 6.5. Küme bazlı tekil kelime ve ortak kelime sayıları

Küme #	Doküman Sayısı	Tekil Kelime Sayısı	Ortak Kelime Sayısı
1	39	69388	18458
2	26	79977	27284
3	34	58233	14485
4	12	64360	20268
5	24	77875	22526
6	41	69245	22244
7	16	37061	10615
8	50	95499	22858
9	34	80970	27140
10	11	39658	11254
11	26	*113948*	35128
12	11	23055	8475
13	7	35986	10103
14	10	21017	8050
15	11	33263	11083
16	9	38802	16532
17	21	63359	21637
18	4	*20830*	6795

Küme bazlı kesinlik, duyarlılık ve F1-skorunda düşük başarı gösteren 11 ve 18 numaralı kümelerin Çizelge 6.5'teki değerleri incelendiğinde her iki kümenin de doküman sayılarına göre diğer kümelere kıyasla oldukça fazla sayıda tekil kelime içerdiği görülmüştür. Bu kategorilere ait dokümanların bu kadar fazla çeşit kelime içermesi benzerliklerin bulunmasını zorlaştırarak kümeleme başarısını olumsuz etkilemiştir.

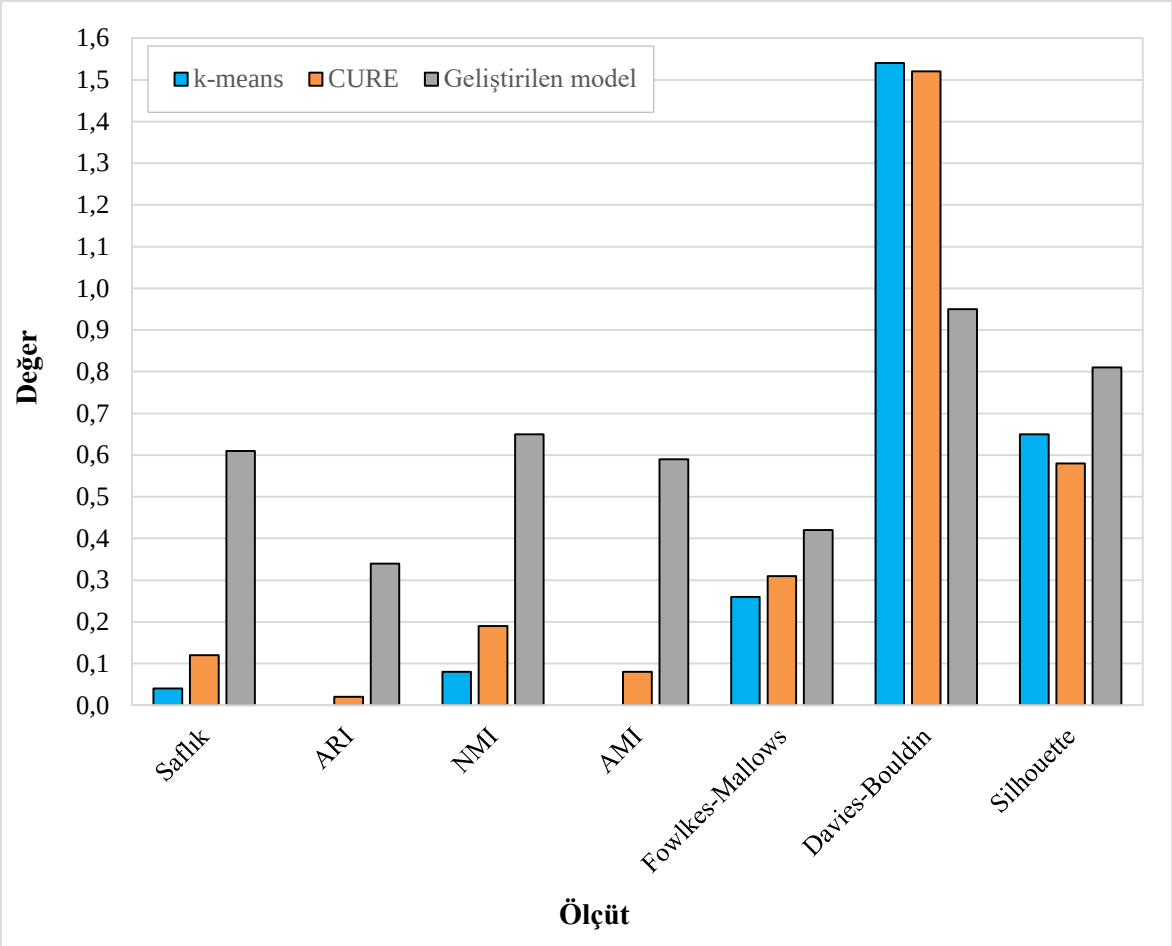
Çalışmada kullanılan veri kümesi, literatürde yaygın olarak kullanılmakta olan k-means ve CURE algoritmaları ile de kümeleme işlemine tabi tutulmuştur. Geliştirilen model ile elde edilen kümeler, k-means ve CURE algoritmaları kullanılarak elde edilenler ile kalite yönünden karşılaştırılmıştır. Bunun için kümeleme performansını değerlendiren bazı ölçütler kullanılmıştır. Bu ölçütler için elde edilen değerler, k-means algoritması, CURE algoritması ve geliştirilen model için Çizelge 6.6'da verilmiştir.

Çizelge 6.6. Geliştirilen modelin diğer algoritmalar ile karşılaştırması

Ölçüt	k-means	CURE	Geliştirilen Model
Saflık	0,04	0,12	0,61
ARI	0,00	0,02	0,34
NMI	0,08	0,19	0,65
AMI	0,00	0,08	0,59
Fowlkes-Mallows	0,26	0,31	0,42
Davies-Bouldin	1,54	1,52	0,95
Silhouette	0,65	0,58	0,81

Geliştirilen model saflık, ARI, NMI, AMI ve Fowlkes-Mallows ölçütlerinin hepsinde literatürde yaygın olarak kullanılmakta olan iki algoritmadan daha iyi performans sergilemiştir. Özellikle NMI ve AMI gibi sıkça kullanılan ölçütlerde elde edilen 0,65 ve 0,59 değerleri oluşturulan kümelerin yüksek kalitede olduğunu göstermiştir. Yine ayarlanmış bir ölçüt olarak sıkça kullanılan ARI için diğer iki algoritmadan daha başarılı bir sonuç elde edilmiş olmasına rağmen diğer ölçütlere göre daha düşük bir sonuç olarak 0,34 değeri elde edilmiştir. Bunun nedeni olarak da ARI'nın büyük ve dengeli kümelerde kullanım için daha uygun olduğu görülmektedir. Elde edilen bu sonuçlar Şekil 6.6'da karşılaştırılarak sunulmuştur.

Seçilen veri kümesi üzerinde farklı birçok ölçüt kullanılarak gerçekleştirilen deneylerde, geliştirilen model yaygın olarak kullanılmakta olan diğer iki kümeleme algoritmasına kıyasla önemli ölçüde daha başarılı sonuçlar vermiştir. Dokümanların büyük boyutunun bir sonucu olarak veri kümesinde çok fazla sayıda farklı sözcük bulunması, farklı türdeki dokümanların bazen aynı sözcükleri içerebilmesi, aynı türdeki dokümanların ise geniş sözcük hazinesinden dolayı benzer ancak farklı sözcükler içerebilmesi kümeleme kalitesini negatif yönde etkileyebilmektedir. Ancak kümeleme işleminde derin öğrenmenin kullanılması, dokümanlardan farkına varılamayacak bazı özelliklerin çıkarılmasını sağlamaktadır. Bu özelliklerin kümeleme işleminde kullanılması elde edilen kümelerin daha kaliteli ve ideal kümelemelere yakın olmasını sağlamaktadır. Ayrıca derin öğrenmenin veriden beslenen bir yöntem olduğu göz önünde bulundurulduğunda daha geniş veri kümelerinde daha başarılı sonuçlar da elde edilebileceği düşünülmektedir.



Şekil 6.6. Geliştirilen modelin diğer yöntemlerle karşılaştırması

7. SONUÇ VE ÖNERİLER

Bu çalışmada, büyük boyutlu dokümanları içerik benzerliğine göre kümelemek için derin öğrenme tabanlı hibrit bir model geliştirilmiştir. Geliştirilen model, 386 İngilizce kitaptan oluşan bir veri kümesi üzerinde test edilmiştir.

Öncelikle doğru bir içerik benzerliği ölçebilmek için metin üzerinde bazı önişlemler gerçekleştirilmiştir. Bu önişlemler ile metinlerin standartlaştırılması sağlanmıştır. Daha sonra, verinin derin öğrenme modelinde kullanılabilmesi için vektör halinde gösterimi gerçekleştirilmiştir. Veri, kümeleme işlemine hazır hale getirildikten sonra kümeleme için derin öğrenme tabanlı bir model geliştirilmiştir.

Geliştirilen derin öğrenme modelinde dokümanlardan özellik çıkarımı gerçekleştirilmiştir. Bunun için, literatürde yaygın olarak kullanılan CNN ve LSTM ağları hibrit bir şekilde kullanılmıştır. CNN, özellik çıkarımında yaygın olarak kullanılan ve başarılı sonuçlar elde edilen bir yapay sinir ağı modelidir. Ancak, bu çalışmada kullanılan dokümanların büyük boyutlu olmasından dolayı tek başına başarılı olamamıştır. Bu nedenle, uzun süreli bellek konusunda oldukça başarılı olan LSTM ağları kullanılmıştır. Derin öğrenme modelinden elde edilen özellikler hash fonksiyonundan geçirilerek normalize edilmiştir. Böylece her doküman bir bucket'a eşlenmiştir. Her bucket birer küme olarak düşünülerek gerekli küme analizleri yapılmıştır.

Yapılan deneysel çalışmalarda elde edilen kümelerin yüksek kalitede olduğu görülmüştür. Ancak, bazı düşük kalitede oluşturulmuş kümeler, ortalama başarıyı etkilemiş ve %66 oranında bir doğruluk değeri elde edilmiştir. Bu düşük kalitedeki kümelerin başında tıp alanındaki dokümanlar gelmektedir. Veri kümesinde bulunan tıp kitaplarının büyük bir bölümü beyin cerrahisi üzerinedir. Bu kitaplar, içerdikleri terimler dolayısıyla bilgisayar bilimlerindeki yapay zeka kitapları ile benzerlik göstermektedir. Bu nedenle model, bu kitapları ayrıştırmada yeterli düzeyde başarı sağlayamamıştır.

Geliştirilen modelin literatürde yaygın olarak kullanılan diğer modeller ile karşılaştırılabilmesi için aynı veri kümesi üzerinde k-means ve CURE algoritmaları da uygulanmıştır. Bu algoritmaların karşılaştırılması için beş farklı kümeleme değerlendirme ölçütü kullanılmıştır. Deneysel sonuçlar geliştirilen modelin bu algoritmalardan çok daha

kaliteli kümeler oluşturduğunu göstermiştir. Bu çalışmada, büyük boyutlu dokümanların kümeleme işlemi sonucunda literatürde çokça kullanılmakta olan NMI ve AMI ölçütleri için sırasıyla 0,65 ve 0,59 değerleri elde edilmiştir. Bu sonuçlar, derin öğrenme yöntemleri kullanarak büyük boyutlu dokümanları kümeleme konusunda literatürde fazla çalışma bulunmaması nedeniyle oldukça önemlidir. Geliştirilen model, Silhouette ve Davies-Bouldin gibi yaygın olarak kullanılmakta olan iç değerlendirme ölçütlerinde ise sırasıyla 0,81 ve 0,95 değerleri elde etmiştir. Çalışmada karşılaştırma için kullanılan diğer algoritmalara kıyasla daha başarılı sonuçlar elde edilmesi derin öğrenme yöntemleri ile dokümanlardan çıkarılan özelliklerin, kümeleme işlemini kolaylaştırarak daha başarılı hale getirdiğini göstermiştir.

Gelecek çalışmalarda bu model, kullanılan hash fonksiyonu ya da metin önişleme işlemleri geliştirilerek daha başarılı sonuçlar elde edilebilir.

KAYNAKLAR

1. Chen, G. (2015). Deep learning with nonparametric clustering. *arXiv preprint*, 1501.03084.

2. Subramani, S., Sridhar, V. and Shetty, K. (2018, November). *A Novel Approach of Neural Topic Modelling for Document Clustering.* Paper presented at the IEEE Symposium Series on Computational Intelligence, Bengaluru, India.

3. Xu, J., Xu, B., Wang, P., Zheng, S., Tian, G. and Zhao, J. (2017). Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88, 22-31.

4. Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Joo, E. M., Weiping, D. and Lin, C. T. (2017). A review of clustering techniques and developments. *Neurocomputing*, 267, 664-681.

5. Luo, K., Liu, Y. and Wang, X. (2007, August). *A Dynamic SOM Algorithm for Clustering Large-Scale Document Collection.* Paper presented at the Sixth International Conference on Advanced Language Processing and Web Information Technology, Luoyang, Henan, China.

6. Feng, Z., Bao, J. and Shen, J. (2010, July). *Dynamic and adaptive self organizing maps applied to high dimensional large scale text clustering.* Paper presented at the 2010 IEEE International Conference on Software Engineering and Service Sciences, Beijing, China.

7. Wang, L. and Dong, M. (2011, July). *On the clustering of large-scale data: a matrix-based approach.* Paper presented at the 2011 International Joint Conference on Neural Networks, San Jose, CA, 139-144.

8. Xiufeng, S. and Wei, C. (2011, December). *Improved CURE algorithm and application of clustering for large-scale data.* Paper presented at the 2011 IEEE International Symposium on IT in Medicine and Education, Guangzhou.

9. Spasojevic, N. and Poncin, G. (2011, September). Large scale page-based book similarity clustering. Paper presented in the 2011 International Conference on Document Analysis and Recognition, Beijing, China.

10. Young, S. R. and Arel, I. (2012, December). Recurrent clustering for unsupervised feature extraction with application to sequence detection. Paper presented at the 11th International Conference on Machine Learning and Applications, Boca Raton, Florida, USA.

11. Forsati, R., Mahdavi, M., Shamsfard, M. and Meybodi, M. R. (2013). Efficient stochastic algorithms for document clustering. *Information Sciences*, 220, 269-291.

12. Yang, Q., Wang, H., Li, T. and Yang, Y. (2015, November). *Deep Belief Networks Oriented Clustering*. Paper presented at the 10th International Conference on Intelligent Systems and Knowledge Engineering, Taipei, Taiwan.

13. Tang, D., Qin, B. and Liu, T. (2015, September). *Document modeling with gated recurrent neural network for sentiment classification.* Paper presented at the Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal.

14. Karaa W.B.A., Ashour A.S., Sassi D.B., Roy P., Kausar N. and Dey N. (2016). MEDLINE Text Mining: An Enhancement Genetic Algorithm Based Approach for Document Clustering. In AE. Hassanien, C. Grosan, M. Fahmy Tolba (Eds.) *Applications of Intelligent Optimization in Biology and Medicine*. Intelligent Systems Reference Library, Springer, Cham.

15. Saiyad, N. Y., Prajapati, H. B. and Dabhi, V. K. (2016, March). *A survey of document clustering using semantic approach.* Paper presented at the International Conference on Electrical, Electronics, and Optimization Techniques, Chennai, Tamilnadu, India.

16. Kulkarni, M., Karande, S. S. and Lodha, S. (2016, April). *Unsupervised word clustering using deep features.* Paper presented at the 12th IAPR Workshop on Document Analysis Systems, Santorini, Greece.

17. Jahromi, A. N. and Hashemi, S. (2017, October). *A deep super-vector based representation for clustering.* Paper presented at the 9th International Conference on Information and Knowledge Technology, Tehran, Iran.

18. Yang, B., Fu, X., Sidiropoulos, N. D. and Hong, M. (2017, August). *Towards k-means-friendly spaces: Simultaneous deep learning and clustering.* Paper presented at the 34th International Conference on Machine Learning, Sydney, Australia.

19. Sreedhar, C., Kasiviswanath, N. and Reddy, P. C. (2017). Clustering large datasets using K-means modified inter and intra clustering (KM-I2C) in Hadoop. *Journal of Big Data*, 4(1), 27.

20. Mei, J. P., Wang, Y., Chen, L. and Miao, C. (2017). Large scale document categorization with fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 25(5), 1239-1251.

21. Sardar, T. H. and Ansari, Z. (2018). Partition based clustering of large datasets using MapReduce framework: An analysis of recent themes and directions. *Future Computing and Informatics Journal*, 3(2), 247-261.

22. Lv, B., Hou, W., Liu, G., Gao, J., Yuan, X., Li, P. and Chen, Z. (2018, July). *A Deep CFS Model for Text Clustering.* Paper presented at the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada.

23. Janani, R. and Vijayarani, S. (2019). Text document clustering using spectral clustering algorithm with particle swarm optimization. *Expert Systems with Applications*, 134, 192-200.

24. Curiskis, S. A., Drake, B., Osborn, T. R. and Kennedy, P. J. (2020). An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit. *Information Processing & Management*, 57(2), 102034.

25. Sardar, T. H. and Ansari, Z. (2020). An Analysis of Distributed Document Clustering Using MapReduce Based K-Means Algorithm. *Journal of The Institution of Engineers (India): Series B*, 101(6), 641-650.

26. Vijaymeena, M. K. and Kavitha, K. (2016). A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3(2), 19-28.

27. Huang, A. (2008, April). *Similarity measures for text document clustering.* Paper presented at the sixth new zealand computer science research student conference, Christchurch, New Zealand.

28. Shirkhorshidi, A. S., Aghabozorgi, S. and Wah, T. Y. (2015). A comparison study on similarity and dissimilarity measures in clustering continuous data. *PloS one*, 10(12), e0144059.

29. Uysal, A. K. and Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, 50(1), 104-112.

30. Anandarajan, M., Hill, C. and Nolan, T. (2019). Text Preprocessing. In M. Anandarajan, C. Hill and T. Nolan (Eds.), *Practical Text Analytics*. Springer, Cham, pp. 45-59.

31. Singh, J. and Gupta, V. (2017). A systematic review of text stemming techniques. *Artificial Intelligence Review*, 48(2), 157-217.

32. Singh, J. and Gupta, V. (2016). Text stemming: Approaches, applications, and challenges. *ACM Computing Surveys (CSUR)*, 49(3), 1-46.

33. Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2), 165-193.

34. Shah, N. and Mahajan, S. (2012). Document clustering: a detailed review. *International Journal of Applied Information Systems*, 4(5), 30-38.

35. Nisha, Kaur, P. J. (2015, March). *A survey of clustering techniques and algorithms.* Paper presented at the 2nd international conference on computing for sustainable global development, New Delhi.

36. Bisht, S. and Paul, A. (2013). Document clustering: a review. *International Journal of Computer Applications*, 73(11).

37. Sahu, S. K. and Srivastava, S. (2016, March). *Review of Web Document Clustering algorithms.* Paper presented at the 3rd International Conference on Computing for Sustainable Global Development, New Delhi.

38. Ng, R. T. and Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5), 1003-1016.

39. Halkidi, M., Batistakis, Y. and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of intelligent information systems*, 17(2), 107-145.

40. Mahfouz, M. A. and Ismail, M. A. (2009). Fuzzy relatives of the CLARANS algorithm with application to text clustering. *International Journal of Electrical and Computer Engineering*, 4(6), 370-377.

41. Yafooz, W. M., Bakar, Z. A. and Mithun, A. M. (2018, December). *Textual Document Clustering in Traditional and Modern Approaches.* Paper presented at the IEEE Conference on Systems, Process and Control, Melaka, Malaysia.

42. Karypis, M. S. G., Kumar, V. and Steinbach, M. (2000, August). *A comparison of document clustering techniques.* Paper presented at the TextMining Workshop at KDD2000, Boston, MA, USA.

43. Dutta, M., Mahanta, A. K. and Pujari, A. K. (2005). QROCK: A quick version of the ROCK algorithm for clustering of categorical data. *Pattern Recognition Letters,* 26(15), 2364-2373.

44. Şenol, A. ve Karacan, H. (2018). Akan Veri Kümeleme Teknikleri Üzerine Bir Derleme. *Avrupa Bilim ve Teknoloji Dergisi,* (13), 17-30.

45. Ahmed, R. D., Dalkılıç, G., & Erten, M. (2018, September). *Survey: Running and comparing stream clustering algorithms.* Paper presented at the 12th Turkish National Software Engineering Symposium, Istanbul, Turkey.

46. Doğan, F., & Türkoğlu, İ. (2019). Derin Öğrenme Modelleri ve Uygulama Alanlarına İlişkin Bir Derleme. *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi*, 10(2), 409-445.

47. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

48. Şeker, A., Diri, B., & Balık, H. H. (2017). Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme. *Gazi Mühendislik Bilimleri Dergisi*, 3(3), 47-64.

49. Kurt, M.S., (2018). *Derin Öğrenme Modelleri ile Web Sayfası Sınıflandırma,* Yüksek Lisans Tezi, İstanbul Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

50. Lipton, Z. C., Berkowitz, J. and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint,* 1506.00019.

51. Mou, L., Ghamisi, P. and Zhu, X. X. (2017). Deep recurrent neural networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7), 3639-3655.

52. Alanis, A. Y. and Sanchez, E. N. (2017). Discrete-Time Neural Observers: Analysis and Applications. *Academic Press,* 14-15.

53. Yuan, X., Li, L. and Wang, Y. (2019). Nonlinear dynamic soft sensor modeling with supervised long short-term memory network. *IEEE Transactions on Industrial Informatics,* 16(5), 3168-3176.

54. Graves, A., Mohamed, A. R. and Hinton, G. (2013, May). *Speech recognition with deep recurrent neural networks.* Paper presented at the IEEE international conference on acoustics, speech and signal processing, Vancouver, Canada.

55. Li, J., Mohamed, A., Zweig, G. and Gong, Y. (2015, December). *LSTM time and frequency recurrence for automatic speech recognition*. Paper presented at the IEEE workshop on automatic speech recognition and understanding, Arizona, USA.

56. Graves, A., Jaitly, N. and Mohamed, A. R. (2013, December). *Hybrid speech recognition with deep bidirectional LSTM*. Paper presented at the IEEE workshop on automatic speech recognition and understanding, Olomouc, Czech Republic.

57. İnternet: Praveen (2020). COVID19-Engineering-Books-NLP-Dataset. Web: https://www.kaggle.com/praveengovi/covid19engineeringbooksnlpdataset adresinden 24 Kasım 2020'de alınmıştır.

58. Palacio-Niño, J. O. and Berzal, F. (2019). Evaluation metrics for unsupervised learning algorithms. *arXiv preprint*, 1905.05667.

59. Marutho, D., Handaka, S. H. and Wijaya, E. (2018, September). *The determination of cluster number at k-mean using elbow method and purity evaluation on headline news.* Paper presented at the International Seminar on Application for Technology of Information and Communication, Semarang, Indonesia.

60. Yang Y. (2017). Temporal Data Clustering. In Y. Yang (Eds.), *Temporal Data Mining Via Unsupervised Ensemble Learning*. Elsevier.

61. Romano, S., Vinh, N. X., Bailey, J. and Verspoor, K. (2016). Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research*, 17(1), 4635-4666.

62. Özdem K. and Akcayol M. A. (2021, March). *Locality Sensitive Hashing Based Clustering for Large Scale Documents*. Paper presented at the International Conference on Mathematics and Artificial Intelligence, Chengdu, China.

GAZİ GELECEKTİR...