KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

SORU DOKÜMANLARININ ANLAMSAL BENZERLİKLERİNE DAYALI DERİN ÖĞRENME TABANLI KÜMELEME ANALİZİ

ERAY YELMEN

KOCAELİ 2020

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

SORU DOKÜMANLARININ ANLAMSAL BENZERLİKLERİNE DAYALI DERİN ÖĞRENME TABANLI KÜMELEME ANALİZİ

ERAY YELMEN

Prof. Dr. Nevcihan DURU		
Danışman,	**Kocaeli Üniversitesi**	**.....**
Doç. Dr. Sevinç İLHAN OMURCA		
Jüri Üyesi,	**Kocaeli Üniversitesi**	**.....**
Dr. Öğr. Üyesi Fatma BÜYÜKSARAÇOĞLU SAKALLI		
Jüri Üyesi,	**Trakya Üniversitesi**	**.....**

Tezin Savunulduğu Tarih: 17.08.2020

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışmasında soru dokümanlarının anlamsal benzerliklerine dayalı derin öğrenme tabanlı kümeleme analizi yapılmıştır.

Yüksek lisans eğitimim süresince bilgi ve tecrübeleriyle çalışmalarıma katkıda bulunan saygıdeğer hocam, tez danışmanım Sayın Prof. Dr. Nevcihan DURU'ya teşekkürlerimi sunarım.

Eğitim hayatım boyunca maddi ve manevi desteklerini esirgemeyen ve her zaman yanımda olan aileme teşekkürü borç bilirim.

Temmuz – 2020 Eray YELMEN

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

C : Kovaryans matrisi
$D(x)^2$: En yakın küme merkezine olan uzaklık
S_t : Temel bileşenler analizi yöntemindeki küçük boyutlu veri seti
U : Üyelik matrisi
X_t : Temel bileşenler analizi yöntemindeki yüksek boyutlu veri seti
λ_i : Kovaryans matrisindeki öz değerler
u_i : Kovaryans matrisi öz vektörleri
$p\ (x_i \mid \theta_j)$: Yoğunluk fonksiyonu
α_0 : Bileşenlerin karıştırma oranları

Kısaltmalar

BoW : Bag of Words (Kelime Torbası)
CBoW : Continuous Bag of Words (Sürekli Kelime Torbası)
CH : Calinski Harabasz Index
DB : Davies Bouldin Index
DBoW : Distributed Bag of Words (Dağıtılmış Kelime Torbası)
DM : Distributed Memory (Dağıtılmış Bellek)
EM : Expectation Maximization (Beklenti Maksimizasyonu)
HCA : Hybrid Clustering Approach (Hibrit Kümeleme Yaklaşımı)
H-FCM : Hyperspherical Fuzzy C-Means (Hiper Küresel Bulanık C-Means)
IR : Information Retrieval (Bilgi Edinme)
PCA : Principal Component Analysis (Temel Bileşenler Analizi)
RDF : Resource Description Framework (Kaynak Açıklama Çerçevesi)
TF : Term Frequency (Terim Frekansı)
TF-IDF : Term Frequency–Inverse Document Frequency (Terim Frekansı ve Ters Belge Frekansı)

SORU DOKÜMANLARININ ANLAMSAL BENZERLİKLERİNE DAYALI DERİN ÖĞRENME TABANLI KÜMELEME ANALİZİ

ÖZET

İnternet ortamında metinsel dokümanların miktarının büyük boyutlara ulaşması ile birlikte aranan doğru dokümana kolay ve hızlı bir şekilde ulaşmak zorlaşmıştır. Metin dokümanlarının benzerliklerine göre kümelenmesi manuel yöntemlerle oldukça zahmetlidir. Bu durumu otomatik hale getirerek kolaylaştırmak için gelişmiş yöntemlere ihtiyaç vardır. Belge kümelemede metin verileri yakınlık ve benzerlik ölçüsüne göre gruplandırılır. Kümelemede yüksek başarı elde etmek, belgelerin doğru bir şekilde keşfedilmesi için oldukça önemlidir.

Kelimelerin anlamsal özelliklerini yoğun vektörler kullanarak temsil etmek için kelime temsil yöntemlerinin kullanımı yaygınlaşmıştır. Yapay sinir ağı tabanlı ve semantik bir yapı içeren kelime temsil yöntemleri, kelimeler arasındaki anlamsal ilişkileri tespit etmekte oldukça başarılıdır. Bu temsil yöntemleri, geleneksel yöntemlere göre daha etkili bulunmuştur. Özellikle anlambilimsel analizlerde başarılı çalışmalar yapılmıştır. Doküman kümeleme çalışmaları kelime temsil yöntemleri veya geleneksel yöntemlerle yapılmış olup tek başına yeterli başarıya ulaşılamamıştır. Bundan dolayı başarıyı artırmak için benzer dokümanları kümelemede kelime temsil yöntemleri ve öznitelik boyut indirgeme yöntemlerinin birlikte kullanılmasına ihtiyaç bulunmaktadır.

Bu tez çalışmasında soru dokümanları üzerinde kümeleme çalışması kelime temsil yöntemleri ve öznitelik boyut indirgeme yöntemlerine odaklanılarak yapılmıştır. Kelime temsili için word2vec, doc2vec ve fasttext yöntemleri kullanılmıştır. Temel Bileşenler Analizi yöntemi ise boyut indirgeme için bu 3 kelime temsil yöntemi ile birlikte kullanılmıştır. Kümeleme için ise k-means++, k-medoids, gaussian mixture ve bulanık c-means algoritmaları üzerinde deneysel çalışmalar yapılmış olup, en yüksek başarı doc2vec, temel bileşenler analizi (PCA) ve bulanık c-means algoritmasının birlikte kullanılması ile elde edilmiştir.

Anahtar Kelimeler: Bulanık C-means, Derin Öğrenme, Doc2Vec, Doğal Dil İşleme, Doküman Kümeleme.

DEEP LEARNING BASED CLUSTERING ANALYSIS BASED ON THE SEMANTIC SIMILARITY OF QUESTION DOCUMENTS

ABSTRACT

With the increase of the amount of textual documents on the Internet, it had difficult to reach the searched correct document easily and quickly. Clustering of text documents according to their similarities is very troublesome by manual methods. Advanced methods are needed to automate this situation by facilitating it. In document clustering, text data is grouped according to proximity and similarity. Achieving high success in clustering is very important for accurate discovery of documents.

The use of word embedding methods has become widespread to represent the semantic properties of words using dense vectors. Word embedding methods based on artificial neural network and containing a semantic structure are very successful in finding semantic relationships between words. These embedding methods were found to be more effective than traditional methods. Successful studies have been carried out especially in semantic analysis. Document clustering studies were made using word embedding methods or traditional methods, and not enough success was achieved alone. Therefore, in order to increase success, it is necessary to use word embedding methods and attribute dimension reduction methods together in clustering similar documents.

In this thesis, clustering on question documents was done by focusing on word embedding and dimension reduction methods. Word2vec, doc2vec and fasttext methods are used for word embeddings. Principal Component Analysis (PCA) was used with this 3 word embedding methods for dimension reduction. For clustering, experimental studies have been conducted on k-means++, k-medoids, gaussian mixture and fuzzy-c means algorithms, and the highest success has been achieved by using doc2vec, PCA and fuzzy-c means algorithm.

Keywords: Fuzzy C-means, Deep Learning, Doc2Vec, Natural Language Processing, Document Clustering.

GİRİŞ

İnternetin gelişmesi ve dijital ortamlardaki bilgilerin çeşitliliğinin ve boyutunun artmasıyla aranan bilgiye kolay bir şekilde erişmek zorlaşmıştır. Bu zorluk kullanıcıların istedikleri bilgiye daha doğru ve hızlı bir şekilde ulaşma ihtiyacını ortaya çıkarmıştır. Bu amaçla dijital ortamdaki dokümanların kümelenmesinde ve aranan bilgiye kolay ulaşmak için yeni yaklaşımlar geliştirilmektedir.

Metinsel dokümanların temsilinde uygun bir model oluşturmak zorlu bir işlemdir. İçeriğe duyarlı olan yapısal olmayan metinsel dokümanların çoğu doğal dil kullanılarak yazılmaktadır. Bir cümledeki sözcüklerin sıralaması ile cümlenin içerdiği anlam birbiriyle ilişkilidir. Bundan dolayı dokümanların temsilinde kullanılacak modelin ilgili dokümandaki sözcüklerin sırasını da dikate alması gerekmektedir [1].

Metinsel dokümanları yönetmenin en faydalı çözümlerinden biri, onları benzerliklerine göre otomatik olarak kümelemektir. Kümeleme, metin belgelerinin kategorik hale getirilmesi ve düzenlenmesi için önemli bir veri madenciliği yöntemidir [2]. Doküman kümeleme işlemi belgeleri ayrı kümelere ayırmaya odaklanır, bu nedenle bir kümeye ait dokümanlar birbirine çok benzer ve diğer doküman kümelerinden farklıdır [3].

Bir cümlede aynı anlamı ifade etmek için farklı kelimeler kullanılabilmektedir. Metinsel dokümanların kelime anlamlarının dikkate alınmayarak kümelenmesi sonucunda ise hata oranı artmaktadır [4].

Küme sayısı kümeleme işlemi yapıldıktan sonra belli olmaktadır. Eğer ilgili dokümanın içeriği de bilinmiyorsa küme sayısını doğru tahmin etmek zordur. Oluşacak küme sayısını kümeleme algoritması çalışmadan vermek yerine bu sayıyı kümeleme algoritmasının bulması daha doğrudur. Metin dokümanlarının kümelenmesinde karşılaşılan bu gereksinimler, metin dokümanlarının doğası da hesaba katılarak yeni yöntem ve algoritmaların geliştirilmesini gerekli hale getirmiştir [1].

Doküman kümeleme işlemi doküman setlerinin benzerliklerine göre ayrıştırılmasıdır. Sonuç olarak bir doküman seti içerisindeki verilerin benzer bir konuda olması gerekmektedir. Bu alanda yapılan çalışmalar 1980'lerin ortalarında başlamış olup, günümüze kadar birçok gelişim göstermiştir [5]. K-means algoritması bu alanda en fazla bilinen ve kullanılan algoritmadır [6].

Bulanık c-means ve k-means algoritmalarını kullanarak yapılan bir doküman kümeleme çalışmasında, web belgeleri kullanılmış olup, belgedeki kelimeler çok boyutlu vektörler olarak temsil edilmiştir. Yapılan deneysel çalışmalar sonucunda bulanık c-means algoritmasının daha az hata oranına sahip olduğu tespit edilmiştir. Bunun yanı sıra kümelerin saflıkları ve entropi değerleri k-means algoritması ile oluşan kümelerin değerlerinden daha iyi olduğu saptanmıştır [7].

Bir diğer çalışmada, e-postaların ilişkili uygulama alanı yapısal ve alana özgü özellikler kullanılarak gruplandırılmıştır. Bu çalışmada K-means, Bisecting k-means ve EM (Beklenti Maksimizasyonu) kümeleme algoritmaları kullanılmıştır [8].

V. Tunalı, çalışmasında çok boyutlu dokümanların yüksek başarı ile kümelenebilmesi amacıyla, k-means algoritmasında değişiklik yaparak, esnek kümeleme ile örtüşen kümeler oluşabilmesi fikrini k-means algoritmasına uygulanmıştır. Bu çalışmada dokümanların birden çok kümeye dahil olmasına imkan tanıyan bulanık k-means algoritması geliştirilmiştir [1].

2010 yılında Shin-Jye Lee ve arkadaşları, bulanık sistemi tanımak için hibrit bir kümeleme tabanlı modüler yöntem önermişlerdir. Kümelerin sayısını ve konumunu bulmak, böyle bir model geliştirmek için en önemli zorluklardan birisi olduğunu belirtmişlerdir. Böylece, girdi, çıktı, uzmanlaşma ve genelleme dikkate alınarak bir HCA (Hibrit Kümeleme Yaklaşımı) tasarlanmıştır. Bu üç bölümden oluşan kümeleme yöntemi, sorunu tanımlamak için bir çok kümeleme özelliğini birlikte kabul eder. [9].

Heterojen metin belgesi içeren [10,11] çalışmada 20 haber grubundan belge koleksiyonunun bir alt kümesi seçilerek yazarların önerdiği yöntemle, DBSCAN ve k-means algoritmaları denenmiş olup, önerilen yaklaşımın DBSCAN ve k-means algoritmalarına göre daha iyi sonuç verdiği tespit edilmiştir. Kalogeratos ve Likas tarafından K-Means algoritmasında küme prototipi olarak küme ağırlık merkezlerinin

yüksek boyutlu ve seyrek doküman verileri için kullanılmasının özellikle az sayıda doküman içeren kümeler söz konusu olduğunda iyi bir seçim olmayabileceği, sentetik küme prototipleri fikrinin uygulanabileceği bu duruma bir çözüm olarak öne sürülmüştür. Bu amaçla MedoidKNN adı verilen sentetik küme prototipi oluşturma yaklaşımında ilk önce küme içerisinde baskın olan sınıftan dokümanlar belirlenir, bunlardan küme temsilcisi hesaplanır ve bu temsilciye göre hatalı olarak bu kümeye atanmış dokümanların daha uygun kümelere atanması sağlanır [12].

Bulanık kümelemenin doküman kümeleme alanında uygulanabilirliği ilk olarak Mendes ve Saks tarafından araştırılmıştır. Bulanık c-means algoritmasındaki amaç ve uzaklık fonksiyonunda gerekli değişiklikler yapılarak, hiperküresel bulanık c-means algoritması geliştirilmiştir. Bu algoritma geleneksel kümeleme yapan yöntemlerle karşılaştırılmış olup, H-FCM'nin (Hiper Küresel Bulanık C-Means) çoğu durumda k-means algoritmasından daha iyi sonuç verdiği belirtilmiştir [13].

Bünyamin Dursun ve A. Coskun Sönmez yaptığı çalışmada Türkçe metin benzerliklerinin tespit edilmesi için yeni bir yöntem önermişlerdir. Yapılan çalışmada Türkçe'de sık yapılan hatalar tespit edilerek bu hataların düzeltilmesi için çözümler önerilmiştir. Sonuçların başarısı Levenshtein Edit Distance benzerliği ve Jaro-Winkler benzerliği ile karşılaştırılmış ve değerlendirilmiştir [14].

Metin tanımlama problemine ek olarak, ağırlıklandırma şemasının tasarımı da dikkate alınması gereken bir diğer sorundur. Yapılan çalışmada ağırlıklandırma şemasının rolünü, metnin belgede sağladığı bilgi miktarına göre daha fazla organize etmek olduğu belirtilmiştir [15]. Metin kümelemedeki TF (Terim Frekansı) ve TF-IDF (Terim Frekansı ve Ters Belge Frekansı) gibi, her bir cümlenin veya her paragrafın yerine yalnızca belgelerdeki her kelimenin önemini vurgulayan terim özelliklerine dayandığı tespit edilmiştir [16].

Basit bir vektör kullanmak, vektör uzaylarında belgeleri temsil etmenin geleneksel bir yoludur. Ancak bu temsil, metinlerin sözdizimsel yapısını göz ardı etme eğilimindedir. Doc2Vec sözcük yerleştirme yöntemi tarafından sunulan matris gösterimi dokümanların sözdizimsel yapısını korur. Belgeleri temsil eden matrisi bir kümeleme algoritmasına entegre etmek için, Frobenius tabanlı bir yöntem kullanılmıştır. Bu yaklaşımın geleneksel kümeleme algoritmalarına göre avantajlar sağladığını ve Bilgi

Edinme (IR) 'de kullanılan kelime torbası (BoW) temsillerinden daha iyi performans gösterdiği belirtilmiştir [17].

Makine öğrenimi alanındaki en son değişiklik, derin öğrenme olarak da bilinen sinir ağlarının yaygın kullanımı olmuştur. [18,19] Özellikle, tek bir belgenin veya kelimenin gizli vektör temsili için kelime temsil yöntemlerinin geliştirilmesi, derin öğrenme tekniklerinin metin belgelerine uygulanmasını hızla genişletmektedir. Bu, durum belge kümeleme ve sınıflandırma gibi çalışmalarda performans artışına yardımcı olur [20].

Karasoy ve Ballı yaptıkları çalışmada SMS leri spam olup olmadığına göre word2vec yöntemi kullanarak sınıflandırmıştır. Word2vec kelime temsil yöntemi ile rastgele orman (random forest) algoritması kullanılmış ve SMS leri başarılı bir şekilde sınıflandırmışlardır [21]. Benzer bir çalışmada ise eğitim seti büyüklükleri farklı 7 kategoride yapılan sınıflandırma çalışmasında word2vec yaklaşımının daha başarılı olduğu gözlemlenmiştir.

Doc2Vec, belge temsil yöntemlerinde kelime-vektör (Word2Vec) gösteriminin bir uzantısıdır. Word2Vec varsayımı, bir sözcüğün öğe değerlerinin, hedef sözcüğü çevreleyen diğer sözcüklerin değerlerinden etkilenmesidir. Bu varsayım, skip-gram gibi bir sinir ağı yapısı olarak kodlanır. Ağ ağırlıkları ve CBOW (Sürekli Kelime Torbası) örneklerinin öğrenilmesiyle ayarlanır [22]. Doc2Vec, Word2Vec'i genişletir, çünkü cümleler de belge olarak kabul edilebilir [23].

Sosyal medya metin verileri üzerinde yapılan kümeleme çalışmasında TF-IDF ve doc2vec yöntemleri ile dokümanlar vektörel hale getirilmiştir. K-means kümeleme algoritmasının bu yöntemlerle kullanılması ile yapılan deneysel çalışmada Doc2vec yönteminin TF-IDF'e göre daha iyi performans gösterdiği belirtilmiştir [18].

Veri kümeleme, birçok web tabanlı uygulama için çok önemli bir görev olarak tanımlanır. RDF (Kaynak Açıklama Çerçevesi) veri kümelemesi için sinir dili modellerini kullanan bir yaklaşımda ilk önce, TF-IDF yöntemi ile Doc2vec ve Word2vec modellerini birlikte kullanarak kullanarak çeşitli grafik alt yapılarından çıkarılan varlık dizileri üretilmiştir. Sonra kümeleme için K-means algoritması uygulanmıştır. Gerçek veri kümeleri üzerinde yapılan deneylerde, RDF verilerinin

kümelenmesinde vektör temsili için Doc2vec ile TF-IDF'in birlikte kullanılmasının iyi sonuçlar verdiği belirtilmiştir [24].

Bir diğer çalışmada öznitelik boyutu yüksek olduğunda, geleneksel kümelenme yöntemleri, mesafe ölçümlerini hesaplamada düşük performans gösterme eğiliminde olduğu öne sürülmüştür [25]. Bu nedenle, öznitelikler üzerinde boyut indirgeme yapılabilmesi için PCA (Temel Bileşenler Analizi) ve doğrusal olmayan dönüşüm yöntemlerinden kernel yöntemleri [26] gibi boyutsallık azaltma ve öznitelik yeniden yapılandırma yaklaşımları önerilmiştir. K-means algoritması üzerine yapılan daha sonraki çalışmalar, özniteliklerin yüksek boyutluluğuna ilişkin bu konunun, boyutsallığı birlikte azaltarak ve kümeleme yaparak ele alındığını ileri sürmüştür [27].

Literatürdeki doküman kümeleme konusunda yapılan çalışmalar incelediğinde özellikle geleneksel kümeleme yöntemlerinin doküman kümeleme de yetersiz olduğu belirtilmiştir. Çalışmalarda, öznitelik seçiminden ziyade daha çok sınıflandırma yöntemleri üzerinde odaklanıldığı ve farklı tekniklerle çeşitli sonuçların alındığı tespit edilmiştir. Bu çalışmada, bir soru cevap sitesi olan Stackoverflow web sitesi üzerinden Ocak ve Ekim 2019 ayları arasında toplam 617.347 örnek İngilizce veri toplanmıştır. Bu veriler üzerinden en fazla soru sorulan 10 anahtar kelime belirlenmiştir. Bu 10 anahtar kelimenin hepsi programlama dilidir. En çok sorulan ilk 3 programlama dili ile ilgili belgeler veri seti olarak hazırlanmıştır. Toplanan veriler ön işlemeden geçirilmiş sonrasında etkisiz kelime silme, kelime düzeltme ve köklerine ayırma çalışmaları yapılmıştır. Bu aşamadan sonra Word2Vec, Doc2Vec ve Fasttext temsil yöntemleri PCA boyut indirgeme yöntemiyle ve bu yöntem kullanmadan, K-means, k-medoids, gaussian mixture ve bulanık c-means algoritmaları kullanılarak deneysel çalışmalar yapılmıştır. En yüksek başarı Doc2Vec, PCA ve bulanık c-means algoritmalarının birlikte kullanılmasıyla elde edilmiştir.

Tez 7 bölümden oluşmakta olup 1. bölümde veri setinin alındığı stackoverflow web sitesi hakkında bilgiler, çalışma kullanılan veri ön işleme aşamaları ile ilgili bilgiler ve öznitelik boyut indirgeme yöntemi olan Temel Bileşenler Analizi hakkında bilgiler verilmiştir. 2. bölümde çalışmada kullanılan Word2Vec, Doc2Vec ve FastText kelime temsil yöntemleri hakkında açıklamalar yapılmıştır. 3. bölümde çalışma kapsamında kullanılan K-means, K-medoids, Bulanık c-means ve gaussian mixture algoritmaları,

4. bölümde Davies - Bouldin, Silhouette ve Calinski - Harabasz değerlendirme metrikleri, 5. bölümde ise dokümanların birbirlerine olan benzerliklerini ölçebilmek için kullanılan Pearson Korelasyon benzerliği ve Kosinüs benzerliği anlatılmıştır. 6. bölümde yazılım ve donanım ortamı, veri kümesinin oluşturulması, veri ön işleme ve deneysel sonuçlardan bahsedilmiştir. Son olarak 7. bölümde ise sonuçlar ve önerilere yer verilmiştir.

1. VERİ KÜMESİ VE METİNLERİN HAZIRLANMASI

Bu bölümde çalışmada kullanılan veri kümesi, veri ön işleme teknikleri ve öznitelik boyut indirgeme teknikleri anlatılmıştır.

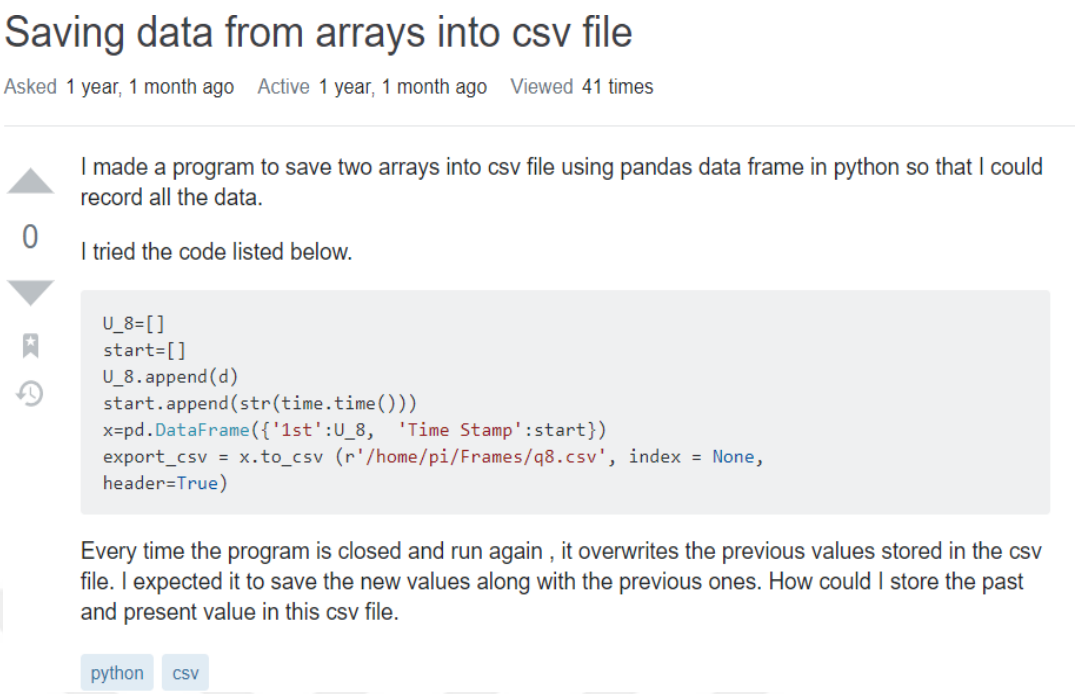
1.1. Stackoverflow

Stackoverflow Jeff Atwood ve Joel Spolsky tarafından 2008 yılında kurulmuş olup, profesyoneller ve programcıların işleri için yararlandığı bir soru cevap sitesidir. Stackoverflow bilgisayar programcılığında çok çeşitli konularda sorular ve cevaplar içerir [28].

Stackoverflow'un kurucuları mevcut soru cevap web sitelerindeki sorulara verilen cevapların yanlış olması, kullanıcıların yanlış veya eksik yanıtlarını düzenleyememesi, karmaşık programlama sorunlarına karşı kapsamlı cevapların bulunamamasından dolayı bu siteyi kurmaya karar vermişlerdir [29].

Bu sitede kullanıcılar sorulara veya cevaplarına verdikleri oylar sayesinde rozet sistemiyle ödüllendirilirler. Sistem, kullanıcıları siteye geri dönmeye ve siteye olumlu katkılar yapmaya teşvik ederek, sorulara verilen yanıtların kalitesini büyük ölçüde artırmaktadır [29]. Bu sayede site üzerinde milyonlarca soru verisi oluşmuş ve veri madenciliğiyle ilgili çalışmaya açık bir hale gelmiştir.

Kullanıcı siteye üye olduktan sonra soru sorabilmekte ve başka kullanıcılara sorulan sorulara cevap verebilmektedir. Şekil 1.1'de stackoverflow üzerinden gönderilmiş bir soru örneği ve Şekil 1.2'de de bir cevap örneği gösterilmiştir.



Saving data from arrays into csv file

Asked 1 year, 1 month ago Active 1 year, 1 month ago Viewed 41 times

I made a program to save two arrays into csv file using pandas data frame in python so that I could record all the data.

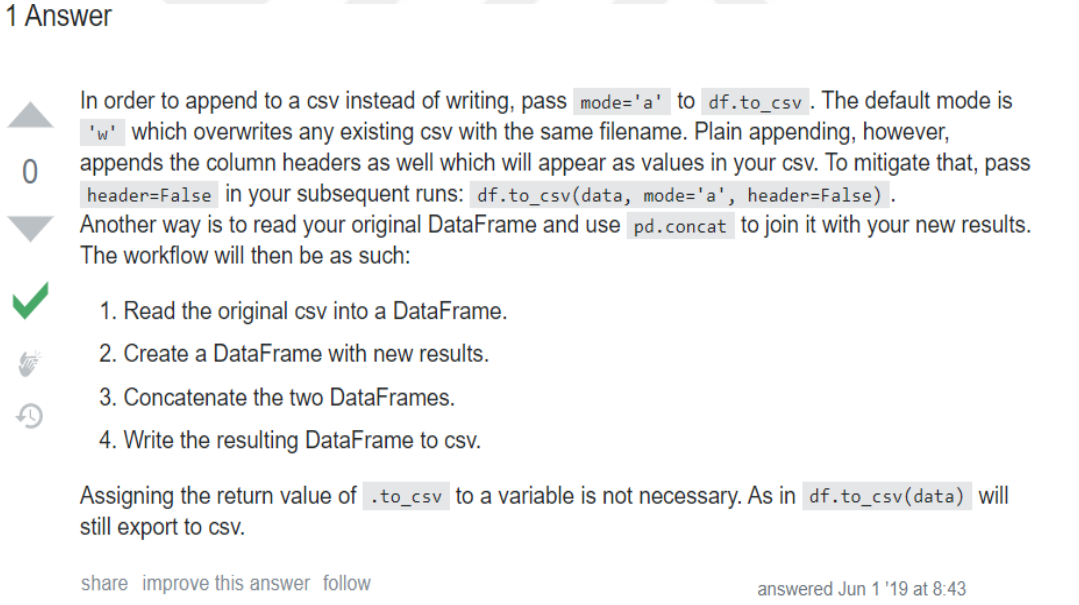
I tried the code listed below.

```
U_8=[]
start=[]
U_8.append(d)
start.append(str(time.time()))
x=pd.DataFrame({'1st':U_8,  'Time Stamp':start})
export_csv = x.to_csv (r'/home/pi/Frames/q8.csv', index = None,
header=True)
```

Every time the program is closed and run again , it overwrites the previous values stored in the csv file. I expected it to save the new values along with the previous ones. How could I store the past and present value in this csv file.

python csv

Şekil 1.1. Stackoverflow soru örneği



1 Answer

In order to append to a csv instead of writing, pass `mode='a'` to `df.to_csv`. The default mode is `'w'` which overwrites any existing csv with the same filename. Plain appending, however, appends the column headers as well which will appear as values in your csv. To mitigate that, pass `header=False` in your subsequent runs: `df.to_csv(data, mode='a', header=False)`.
Another way is to read your original DataFrame and use `pd.concat` to join it with your new results. The workflow will then be as such:

1. Read the original csv into a DataFrame.
2. Create a DataFrame with new results.
3. Concatenate the two DataFrames.
4. Write the resulting DataFrame to csv.

Assigning the return value of `.to_csv` to a variable is not necessary. As in `df.to_csv(data)` will still export to csv.

share improve this answer follow

answered Jun 1 '19 at 8:43

Şekil 1.2. Stackoverflow cevap örneği

Kullanıcılar soru sorma işleminde konuya uygun olarak etiket seçmek zorundadır. Diğer kullancılar bu etiketler aracılığıyla kendi sorularına yanıt bulabilmekte veya sorulara yanıt veren kullanıcılar bu etiketler aracılığıyla filtreleme yaparak sorulara yanıt vermektedirler. Şekil 1.1'de gösterilen soru örneğindeki 'python', 'pandas', 'replace', 'dataframe' ve 'rename' kelimeleri bu soruya ait etiketlerdir.

Tez kapsamında Ocak ve Ekim 2019 arasında toplam 617,347 veri toplanmış olup bu verilerden python, javascript ve java etiketli veriler kullanılmıştır. Bu verilerden 101,111 tanesi python, 74,357 tanesi javascript, 52,885 tanesi java etiketiyle etiketlenmiş verilerden oluşmaktadır.

1.2. Veri Ön İşleme

Metinlerden anlam çıkarımı yapabilmek ve öğrenme başarısını artırmak amacıyla verilerin ön işleme aşamasında geçirilmesi gerekmektedir. Veriler bazı özel formatlar, sayı biçimleri, tarih biçimleri, ön ekler, özel ifadeler içermesinden ötürü işleme tabi tutularak metinden ayıklanabilirler. Ayrıca köke indirgeme ve kelime düzeltme de yapılan diğer işlemlerdir. Bu işlemler metin madenciliğinde başarılı çıkarım yapılabilmesi için önemlidir. Metin ön işleme yapılmasıyla, madencilik ve kümeleme performansı artar [30]. Ayrıca yapısal olmayan veriden yapısal bir metin elde edilebilmesi amacıyla bazı dönüştürme işlemleri yapılır [31]. Veri ön işlemede kullanılan ve bu tez kapsamında da kullanılan teknikler aşağıda belirtilmiştir.

1.2.1. Ayrıştırma

Herhangi bir anlam içermeyecek karakterleri ayrıştırmak için doküman üzerinde uygulanan bir işlemdir. Bu işlemde, ilgili dokümanda tüm sekmeler, noktalama işaretleri, boşluklar ve metin olmayan diğer karakterler yok sayılarak bir kelime dizisi oluşturulur. Buna ayrıştırma (tokenization) işlemi denir [32].

1.2.2. Tüm karakterlerin küçük harfe çevrilmesi

Dokümanda, büyük ve küçük harflerden dolayı ifadelerin farklı algılanmasını önlemek için tüm metin yazıldığı dilin alfabesi dikkate alınarak küçük harfe dönüştürülür.

1.2.3. Köke indirgeme

Öğrenmeyi kolaylaştırmak ve model başarısını artırmak amacıyla ilgili doküman kelimelere ayrıldıktan sonra anlamca farklı olsa da aynı kök veya gövdeye sahip kelimeler üzerinde kök veya gövde bazında indirgeme işlemi yapılır [33]. Yapılacak işlem ile birlikte kelime vektörü de benzer anlamlı kelimelerden arındırılmış olur. Kök

seviyesine indirgeme işlemi yapılmadığında kelimelerin hepsi farklı bir kelime olarak yorumlanır ve modelin başarısını düşürür.

1.2.4. Etkisiz kelimeler

Modelin öğrenme başarısını yükseltmek amacıyla, her dokümanda bulunabilecek ancak kümeleme veya sınıflandırma için bir anlam ifade etmeyecek etkisiz kelimeler kaldırılır. Bu kelimeler kullanılan dile bağlı olarak değişiklik göstermektedir. Örnek olarak İngilizce'de "and", "or", "before", "so", "as" gibi kelimeler veri kümesinden atılırlar [32]. Dokümanların tamamında geçebilecek olan bu kelimeler, vektör boyutunu artırarak kümeleme veya sınıflandırmaya katkıları olmayacağı için dokümandan kaldırılırlar.

1.2.5. Kelime düzeltme

Bağlama duyarlı yazım düzeltmesi, genellikle girdinin bağlamına bağlı olarak bir kelime algılama ayrıştırma mekanizması olarak ifade edilir. Örneğin, bir kelime ilk bakıldığında düzgün yazılmış olarak görülebilir ama anlam bakımından hatalı yazılmıştır. İşte bu durumda cümleyi anlamsal olarak algılama ve düzeltme ihtiyacı bulunmaktadır. Örneğin, "It is to cold in the winter" ifadesinde "to" tek başına bağlaç olsa da cümlenin tamamı için bir anlam ifade etmemektedir. Doğrusu "to" yerine "too" yazılmasıdır [34].

1.3. Boyut İndirgeme

Öznitelik boyutu geniş doküman koleksiyonlarında yüzbinleri bulabilir. Dolayısıyla performans ve model başarısı açısından sorun yaşanmaması için öznitelik seçim yöntemleri oldukça önemlidir. Makine öğreniminde çeşitli öznitelik seçim yöntemleri bulunmaktadır. Bu yöntemler sınıflandırma sisteminin eğitimi için kullanılan dokümanlardaki özniteliklerin dağılım ve istatistiklerinden yararlanmaktadırlar. Fakat metin kümelemede önceden böyle bir istatistiksel bilgi bulunmamaktadır. Bundan dolayı kümeleme için farklı boyut indirgeme teknikleri geliştirilmiştir [6,46]. Bu tez çalışmasında ise Temel Bileşenler Analizi yöntemi kullanılarak boyut indirgeme yapılmış olup, bu yöntem ile ilgili detaylara aşağıda değinilmiştir.

1.3.1. Temel bileşenler analizi (Principal component analysis)

Temel Bileşenler Analizi yöntemi, tanıma, sınıflandırma, görüntü sıkıştırma alanlarında kullanılan, bir değişkenler setinin varyans kovaryans yapısını, bu değişkenlerin doğrusal birleşimleri vasıtasıyla açıklayarak, boyut indirgenmesi ve yorumlanmasını sağlayan, çok değişkenli bir istatistik yöntemidir. Diğer adı Karhunen-Loeve metodudur. Temel Bileşenler Analizi, verideki gerekli bilgileri ortaya çıkarmada oldukça etkilidir. Bu yöntem ile yüksek boyutlu verilerdeki genel özellikler bulunarak boyut sayısının azaltılmasını, verinin sıkıştırılmasını sağlanmaktadır. Boyut indirgemeyle bazı özelliklerin kaybedileceği kesindir. Ancak bu kaybolan özellikler veri seti hakkında çok az bilgi içermektedir [47].

Temel Bileşenler Analizi yönteminin amacı, verinin çeşitliliğini daha iyi yakalayacak yeni bir öznitelik grubunun bulunmasıdır [48]. Bu yöntemde ilk boyut, mümkün olduğunca çok çeşitliliği gösterecek şekilde seçilir. 2. boyutta ise ilk boyuta dikey olacak ve yine mümkün olduğunca çok çeşitliliği gösterecek şekilde seçim yapılır.

PCA, veri varyansını belirlemek için temel bileşenler (PCs) olarak bilinen yeni bir ortogonal özellik oluşturur. PCA, bir veri uzayındaki gereksiz özellik boyutlarının azaltılması amacıyla kullanılmaktadır [49,50]. Veri setinin boyutunun azaltılmasıyla makine öğrenmesi süreçlerinde hesaplamada kolaylığı sağlanır. PCA, yüksek boyutlu veri setlerini (X_t), kovaryans matrisinin (C) özdeğerlerini ve öz vektörlerini belirleyerek daha küçük boyutlu bir veri seti (S_t) haline dönüştürür. PCA ile ilgili denklemler şu şekildedir:

$$c(x)= \sum_{i=1}^{N} \frac{(x_i x^T)}{N} \tag{1.1}$$

$$\lambda_i u_i = c u_i \quad , \qquad i=1,2,3, \ldots,m \tag{1.2}$$

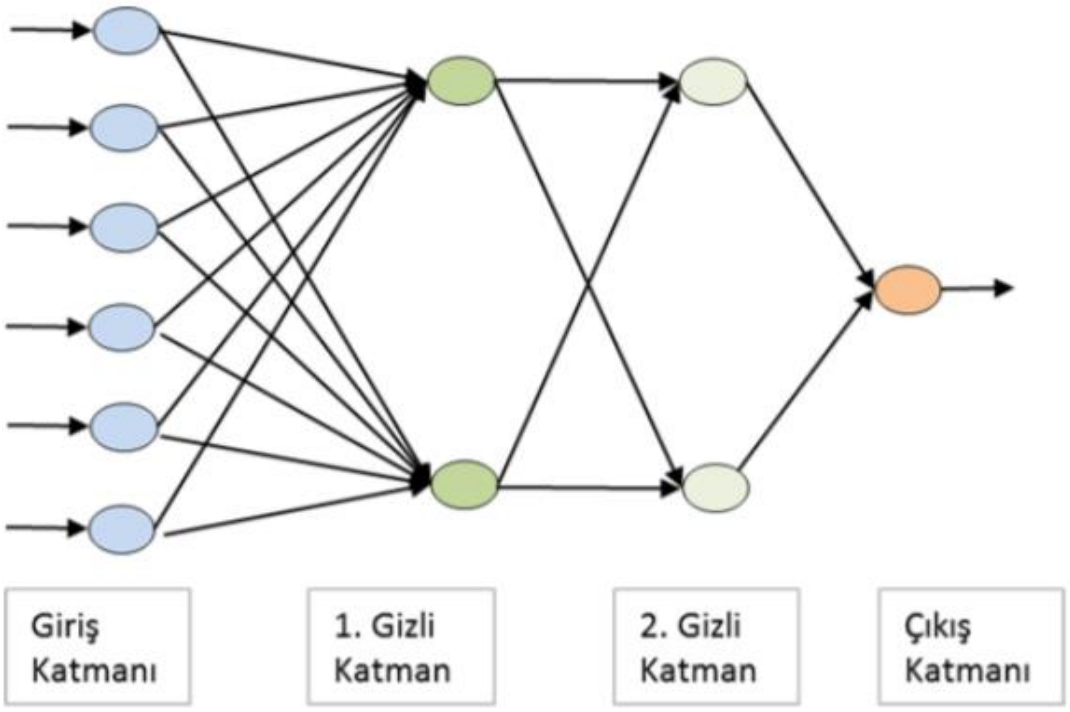
Denklemlerde λ_i kovaryans matrisinde özdeğerleri ifade ederken, u_i ise aynı matrisin özvektörlerini belirtir.

$$S_t(i) = u_i^T x_t \,, \qquad i=1,2,3, \ldots,m \tag{1.3}$$

Denklem 1.3'de, $S_t(i)$, veri seti (X_t)'nin temel bileşenlerini ifade eder [49,51].

2. DERİN ÖĞRENME

Derin öğrenme, veride saklı olan bilgilerden en fazla şekilde yararlanmaya olanak sağlayan, gerçek dünyadaki karmaşık verilerle çalışabilen, bu karmaşık verileri içi içe geçmiş hiyerarşiler ile ele alabilen ve bu hiyerarşileri basit ilişkiler kurarak daha başarılı sonuçlar elde eden makine öğrenmesi tekniğidir. Bir başka tanıma göre; derin öğrenme, yapay sinir ağları, yapay zeka, grafiksel modelleme, optimizasyon, model tanıma ve sinyal işlemenin bir araya gelmesiyle meydana gelen daha güçlü tahminler yapmak için kullanılabilecek makine öğrenimi tekniğidir. Derin öğrenme, temel makine öğrenmesi tekniklerinden farklı olarak daha fazla nöron içerirler, yapay sinir ağlarına göre hiyerarşik biçimde daha karmaşık bir yapıdan oluşurlar ve eğitim aşamasında daha fazla güç tüketirler [82]. İki katmalı bir derin öğrenme yapısı Şekil 2.1'de gösterilmiştir.



Şekil 2.1. İki gizli katmanlı derin öğrenme yapısı

2.1. Kelime Temsil Yöntemleri

Kelime temsil yöntemleri, Frekans Bazlı ve Tahmin Bazlı olmak üzere iki bölümden oluşmaktadır. Frekans bazlı kelime temsilleri, geleneksel yöntemler olarak bilinmekte olup, dokümanlar içerisinde bulunan kelimeler ve bu kelimelerin frekanslarının belirlenmesi yöntemine dayanmaktadır [35].

Frekans bazlı kelime temsilinde en çok kullanılan yöntem Kelimeler Çantası (Bag of Words) dır. Bu yöntemde, doküman içerisindeki her cümle benzersiz kelimelere ayrılarak, n boyutlu bir matrise dönüştürülür. Matristeki sütunlar dokümandaki kelimelerden, satırlar ise dokümanlardan oluşur [36]. Frekans bazlı temsil yöntemlerinden bir diğeri ise TF-IDF (Term Frequency-Inverse Document Frequency) yöntemidir. Terim frekansı, bir dokümanda geçen terim frekanslarını elde etmeye yarar. Ters Doküman Frekansı ise birden fazla dokümanda kelimenin geçme sayısını bulup, kelimenin terim olup olmadığını tespit etmeye çalışır [37].

Frekans bazlı temsil yöntemlerinin en önemli iki dezavantajı vardır. Bunlardan birincisi, oluşturulan matris yapısında birçok değer sıfır olacağından seyrek matris (sparse matrix) durumu meydana gelmektedir. İkinci dezavantaj ise, kelimeler arasındaki anlamsal yakınlıkların tespit edilememesidir [38].

Tahmin bazlı kelime temsil yöntemi ailesinden olan Word2Vec modeli Mikolov ve arkadaşları tarafından 2013 yılında geliştirilmiş olup, temelinde yapay sinir ağı ile kelimelerin eğitilmesi ilkesi bulunmaktadır [22]. Word2Vec modeli CBOW (Continous Bag of Words) ve Skip-gram olarak adlandırılan iki farklı algoritmadan oluşmaktadır.

Word2Vec modeli tahmin bazlı bir kelime temsil yöntemi olup, girdi olarak büyük bir derlem içerisindeki benzersiz kelimeleri alarak belirlenen bir temsil vektörü boyutunda matris oluşturur. Bu model her seferinde dokümandaki bir cümleyi pencere (window) denilen bir yapı içerisinde kaydırarak taramakta ve penceredeki hedef kelimeye göre genellikle çok sayıda boyuttan oluşan bir vektörü çıktı olarak üretmektedir [23].

Bir diğer temsil yöntemi olan Doc2Vec yönteminin temeli Word2Vec yöntemine dayanmaktadır. Word2Vec, kelimeleri uzaysal bir düzleme taşıyarak bir vektör

oluşturur. Doc2Vec benzer işlemi cümleler ya da paragraflar için gerçekleştirir. Doc2Vec yöntemi Paragraph2Vec olarakta ifade edilmektedir [39].

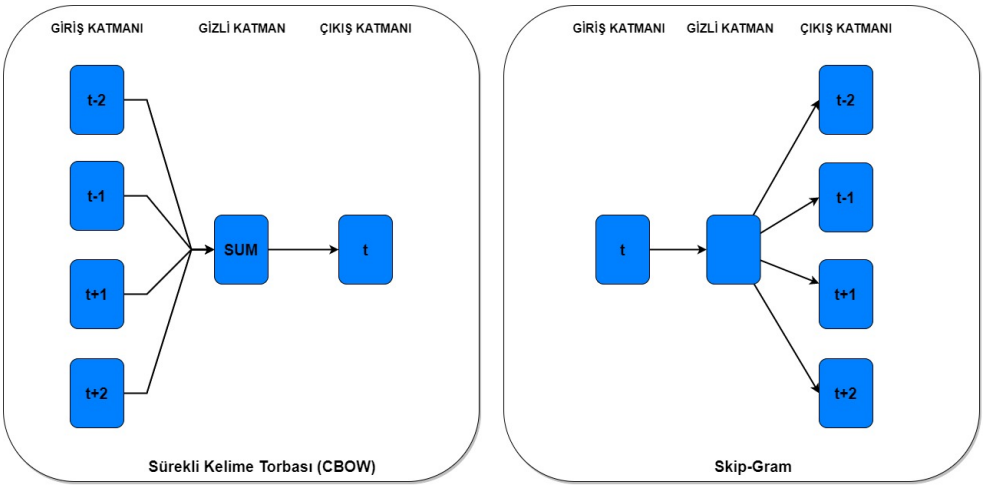
FastText ise Facebook'un yapay zeka laboratuvarı tarafından oluşturulan kelime temsilini ve metin sınıflandırmasını öğrenmek için üretilmiş bir kütüphanedir. Model, kelimeler için vektör temsillerini elde etmede gözetimsiz bir öğrenme veya denetimli öğrenme algoritması oluşturmaya izin verir. FastText, kelime temsili için sinir ağı kullanır [40].

Bu bölümde deneysel çalışmalarda kullanılan Word2Vec, Doc2Vec ve FastText temsil yöntemleri detaylı bir şekilde açıklanmıştır.

2.1.1. Word2Vec

Word2Vec kelimeleri vektör uzayında ifade etmek için kullanılan tahmin temelli ve denetimsiz bir sinir ağıdır. Bu model 2013 yılında Google'da çalışan araştırmacı Tomas Mikolov ve ekibi tarafından bulunmuştur [23]. Bu yöntemin amacı benzer kelimeleri vektör uzayında matematiksel olarak tanımlayarak gruplandırmaktır [41].

Kelimelerin temsilini oluşturmak için sürekli kelime torbası modeli (CBOW) ve skip-gram olmak üzere iki farklı Word2Vec modeli vardır. Sürekli kelime torbası modelinde merkezde olmayan kelimeler girdi olarak alınarak merkezdeki kelime tahmin edilmeye çalışılır. Skip-gram modelinde ise merkezdeki kelime girdi olarak alınarak merkezde olmayan kelimeler tahmin edilmeye çalışılır. CBOW ve skip-gram modelleri Şekil 2.2.'de gösterilmiştir [22].



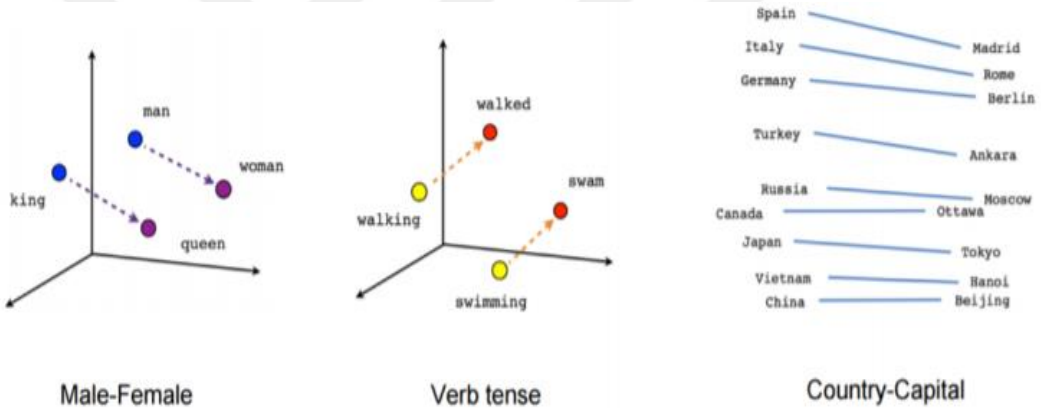
Şekil 2.2. CBOW ve Skip-Gram modelleri

Word2vec vektör uzayında bir kelimeye en yakın kelimeleri bulabilmek için kullanılabilir. Bu yöntem kelimelerin dokümanda kullanımına göre farklı kelimelerle ilişki oluşturup aynı zamanda vektörler arasındaki uzaklıları da hesaplama imkanı sağlar [22].

Şekil 2.3'de gösterilen "King" ve "Queen" arasında ilişki ile "Man" ile "Woman" arasındaki ilişki aynıdır [41]. Yine ülkeler ve başkentleri arasında bir ilişki olduğu da yeteri kadar metin işlediğinde görülebiliyor.

Bunun yanı sıra, word2vec yöntemi ile aritmetik işlemler veya fonksiyonlar kullanarak kelimeler arasındaki bağlamın oluşturulması da sağlanmaktadır.

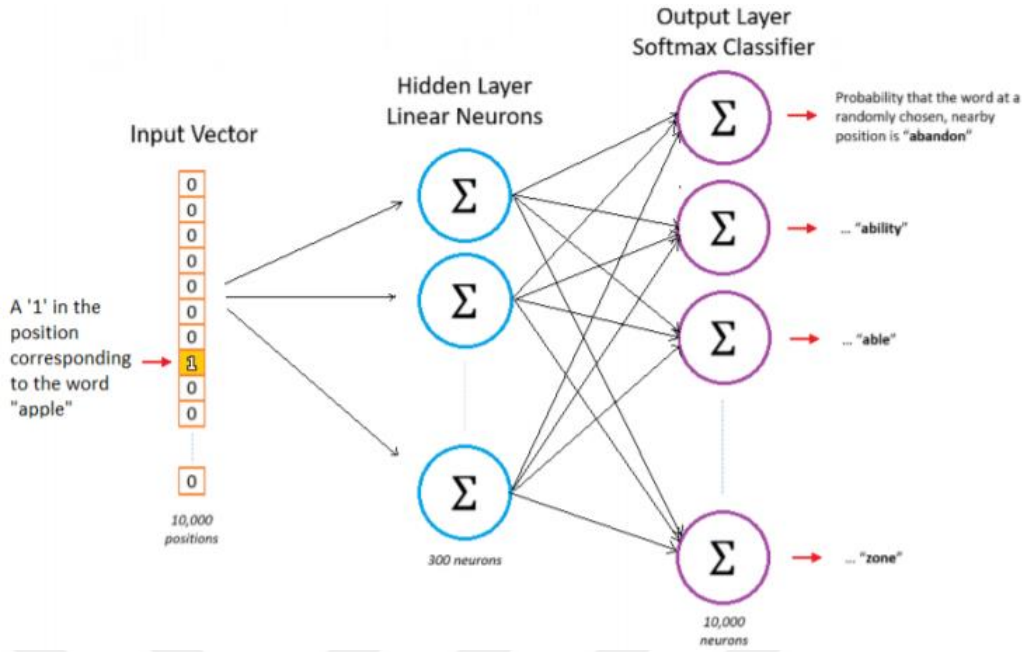
Örn: vec(Spain) – vec(Madrid) = vec(Turkey) – vec(Ankara)



Şekil 2.3. Kelimeler arasındaki ilişkiler

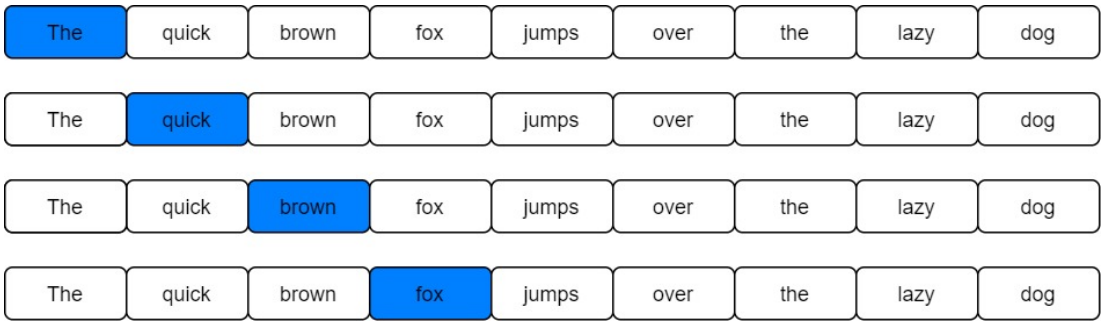
Şekil 2.4'te gösterilen word2vec mimarisine [41] göre 10,000 adet benzersiz kelimeden oluşan bir kelime sözlüğünde "elma" kelimesi giriş vektörü olarak kullanıldığında, ağın çıktısının sözlüğümüzdeki her kelimenin olasılığını içeren ve kelimeye yakın anlamda olan rastgele bir vektör olduğu görülmektedir.

Gizli katmanın nöronlar üzerinde aktivasyon fonksiyonu yoktur, ancak çıkış nöronları softmax kullanır. Eğer 300 özniteliğe sahip kelime vektörlerine ihtiyaç olursa gizli katman 10,000 satır ve 300 sütun içeren bir ağırlık matrisi ile temsil edilecektir [41].



Şekil 2.4. Word2Vec mimarisi

Word2Vec modelinde önemli parametrelerden bir tanesi pencere boyutudur. Pencere boyutu merkezde bulunan kelimenin sağında ve solunda kaç kelime olabileceğini ifade eder. Şekil 2.5'teki skip-gram modelinde "quick" kelimesini kullanılarak "the" ve "brown" kelimeleri tahmin edilmeye çalışılırken CBOW modelinde ise "quick" kelimesi "the" ve "brown" kelimeleri kullanılarak tahmin edilmektedir.



Şekil 2.5. Pencere boyutu örneği

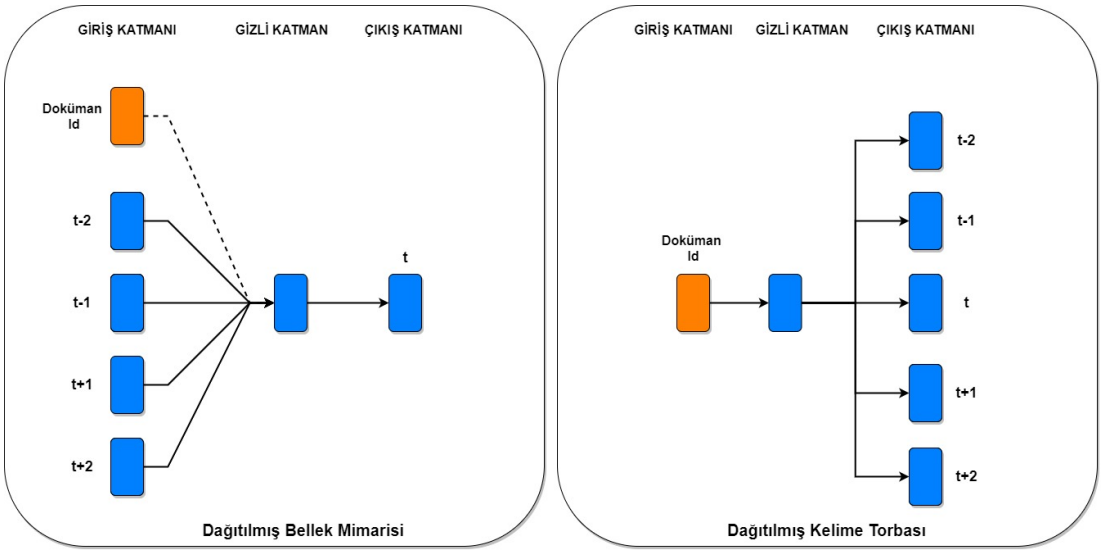
CBOW modeli skip-gram modeline göre çok daha hızlı olup, büyük veriler için daha uygundur. Ancak tahminlemede skip-gram CBOW a göre daha iyi performans gösterir [23]. Word2Vec yönteminde benzerlik terimleri, kosinüs benzerliği kullanılarak hesaplanabilmektedir [42]. Kelime vektörleri arasındaki kosinüs benzerliğinin formülü 2.1'de ifade edilmiştir [41].

$$\text{sim}(A,B) = \cos(\ ) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\Sigma_{i=1}^{n} A_i B_i}{\sqrt{\Sigma_{i=1}^{n} A_i^2}\sqrt{\Sigma_{i=1}^{n} B_i^2}} \quad (2.1)$$

2.1.2. Doc2Vec

Word2Vec algoritması, kelimeleri vektör uzayında ifade eden tahmin temelli bir yapıdadır. Doc2Vec yönteminde de ifadeler veya paragraflar word2vec'te olduğu gibi benzer işlemleri gerçekleştirir. Bazı kaynaklarda Doc2Vec'e Paragraph2Vec'te denilmektedir. Doc2Vec, Word2Vec algoritmasının geliştirilmiş halidir. Burada Word2Vec algoritmasına göre yapılan tek değişiklik, Şekil 2.6'te görülebileceği gibi benzersiz bir belge kimliğinin (ID) eklenmesidir. Doc2vec aynı zamanda kelimelerin aritmetik ortalamasını da dikkate alır [43].

Doc2Vec, paragraf vektörü üreten sinir ağına ek giriş nöronları uygulayarak sınırlamaları araştırmaktadır. Bu açıdan Doc2Vec temel olarak Word2Vec'e uygun değiştirilmiş 2 yöntem sunar [44]. Word2Vec'teki yöntemler, sürekli kelime torbası (CBoW) ve Skip-Gram'dır. Bu yöntemler Doc2Vec için yapılandırılarak Dağıtılmış Bellek (DM) ve Dağıtılmış kelime torbası (DBoW) olmak üzere iki farklı yönteme çevrilmiştir [43].



Şekil 2.6. Doc2Vec mimarisi

2.1.3. FastText

FastText'te [45], amaç her kelimeyi n-gram karaktere bölmektir. Ön ekleri ve son ekleri ayırt etmek için sözcüğün başına ve sonuna özel bir karakter eklenir. Örneğin, n, 3'e eşit olduğunda, 3-gramlık “this” kelimesi aşağıdaki gibidir:

<th, thi, his, is>

Eğitim sürecinde her kelimenin kelime karakterleri ve karakter gramları dikkate alınır. Bir kelimenin temsili, n-gram vektörlerinin toplanmasıyla hesaplanır. Kabul edilebilir sonuçlar elde etmek için üç ila altı karakter gramı yeterlidir [45]. En iyi n değeri dile bağlıdır.

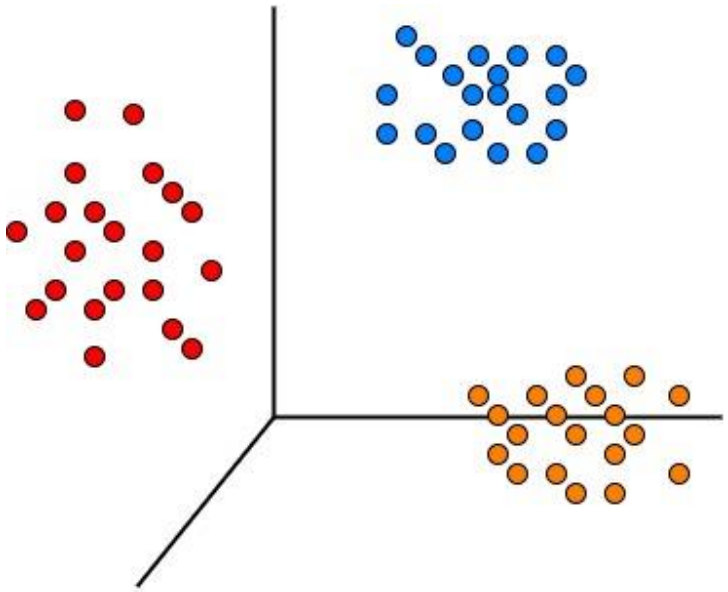
N değerinin yüksek olması karakter gram sayısı ile doğru orantılı ve tüm n-gramların hızlı bir şekilde erişilebilir olması için bir hashmapta tutulması önerilmektedir. Word2vec'e benzer şekilde, eğitim verilerinin boyutu arttıkça fastText modelinin performansı daha iyi olur [45]. Karakter n-gram kullanmanın bir diğer avantajı da modelin sözdizimsel özellikleri daha iyi yakalama yeteneğine sahip olmasıdır.

3. METİN KÜMELEME

Kümeleme, belgelerin özelliklerinin benzerliklerine göre gruplara ayrılması için denetimsiz bir veri madenciliği tekniğidir [52]. Amaç, benzer belgeleri, yüksek küme içi benzerlik ve düşük küme arası benzerlik olacak şekilde aynı grupta bir araya getirmektir [53].

Kümeleme gözetimsiz sınıflandırmaya girer. Gözetimsiz sınıflamada, başlangıçta verilen ve sınıflandırılmamış bir kümeden anlamlı alt kümeler oluşturulması amaçlanmaktadır. Kümeleme analizi veri setinin özelliklerine göre yapılır.

Kümeleme analizinde benzer uzaklıklar dikkate alınarak yararlanılabilecek yöntem ve metrikler bulunmaktadır. Birimler arası uzaklıklar için Euclidyen, Manhattan Mahalanobis, Kareli Euclidyen, Standardize Euclidyen, Canberra veya Minkowski metrikleri kullanılabilmektedir. Kümeleme algoritması veritabanını alt kümelere ayırmaktadır. Her bir kümede yer alan elemanların özellikleri diğer kümelerdeki elemanların özelliklerinden farklıdır. Şekil 3.1'de bir koordinat düzleminde küme üyelerinin birbirlerine çok benzediği, ancak özellikleri açısından birbirlerinden çok farklı olan kümelerin bulunduğu görülmektedir.



Şekil 3.1. Koordinat düzleminde kümeleme örneği

Kümeleme yöntemleri hiyerarşik ve hiyerarşik olmayan (bölümlemeli) kümeleme yöntemleri olmak üzere iki bölüme ayrılmaktadır. Ancak yapılan araştırmalar bu yöntemlerin daha alt bölümlere ayrılabileceğini de göstermektedir. Hiyerarşik kümeleme yönteminde en çok kullanılan metotlar, tek bağlantılı, tam bağlantılı, ortalama bağlantı, merkezi ve ward metotlarıdır. Küme sayısı hakkında bir ön bilginin olması veya araştırmacının anlamlı olacak küme sayısına karar vermiş olması durumunda hiyerarşik olmayan kümeleme yöntemi tercih edilmektedir. Hiyerarşik olmayan kümeleme yönteminde en çok tercih edilen yöntem ise k-means kümeleme yöntemidir [54].

Bu tez kapsamında doküman kümelemede Gaussian Mixture, Bulanık C-means, K-means++ ve K-medoids algoritmaları kullanılmış olup, bu algoritmalar ile ilgili detaylar aşağıda belirtilmiştir.

3.1. Gaussian Mixture

Gaussian Mixture, kümeleme problemlerini çözmede, bazı durumlarda çok etkili olan ancak en hızlı olmayabilecek bir modeldir [55]. Gauss Karışımı, varyans, ortalama ve önceki olasılık dahil olmak üzere her kümenin parametrelerine bağlı olarak belirli bir veri kümesini k sayıda kümeye böler. Bu parametrelerin her biri Expectation Maximization (EM) algoritması ile belirlenir. Algoritma adımları şu şekildedir: Yeni bir veri noktası verilirse, belirli bir kümeye ait olma olasılığı hesaplanır ve veri noktası en yüksek olasılık kümesine atanır [56].

Sonlu karışım modelleri kullanılarak yapılan kümeleme, [57]'de ayrıntılı olarak açıklanmaktadır. Bu modelde, verilerin k bileşeni yoğunluk fonksiyonlarının bir karışımından üretildiği varsayılmaktadır. Burada p ($x_i \mid \theta_j$), tüm j $_0$'lar için j bileşeninin yoğunluk fonksiyonunu temsil eder. θ_j ise j kümesi için parametredir. x_i'nin olasılık yoğunluğu şu şekilde ifade edilir:

$$P(x_i) = \sum_{j=1}^{k} \alpha_j P(x_i|\theta_j) \tag{3.1}$$

Burada α_0'lar bileşenlerin karıştırma oranlarıdır. ($\alpha_j \geq 0$ ve $\Sigma_{j=1}^{k} \alpha_j = 1$) N veri noktalarının log olasılığı, şu şekildedir:

$$L=\sum_{i=1}^{N} \ln \{ \sum_{j=1}^{k} \alpha_j P(x_i|\theta_j) \} \quad (3.2)$$

Burada 3.2'deki denklemi doğrudan optimize etmek zordur. Bu nedenle veri kümesi için parametrelerin (yerel) maksimum olasılığını veya maksimum posteriori (MAP) tahminini bulmak için Expectation Maximization (EM) [58] algoritması uygulanmaktadır. EM algoritması birleşme noktasına kadar bir tahminleme adımı ve bir maksimizasyon adımı arasında yinelenir.

3.2. Bulanık C-means

Bulanık c-means algoritması, bulanık kümeleme yöntemlerinden en yayın kullanılanıdır. 1973 yılında Dunn tarafından bulunan bulanık c-means algoritması 1981 yılında Bezdek tarafından geliştirilmiştir [59]. Bulanık c-means amaç fonksiyonu temelli bir algoritmadır. Bulanık c-means algoritması, nesnelerin iki veya daha fazla kümeye ait olabilmesine olanak sağlar. Bulanık mantık prensibi gereği her veri, kümelerin her birine [0,1] arasında değişen birer üyelik değeri ile aittir. Bir verinin tüm sınıflara olan üyelik değerleri toplamı "1" olmalıdır. Nesnenin küme merkezine yakın olması diğer kümelere göre üyelik değerinin büyük olması anlamına gelir. Amaç fonksiyonun belirlenen minimum ilerleme değerine yakınlaşmasıyla kümeleme işlemi tamamlanır.

Bulanık c-means algoritması, aşağıdaki amaç fonksiyonunu öteleyerek minimize etmeye çalışır [59].

$$J_m=\sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^{m} \left\| x_i - c_j \right\|^2 \quad , \quad 1 \leq m < \infty \quad (3.3)$$

Algoritma U üyelik matrisinin rasgele atanmasıyla başlatılır. İkinci aşamada ise merkez vektörleri hesaplanır. Merkezler 3.4 nolu denklem ile hesaplanır [59].

$$c_j=\frac{\sum_{i=1}^{N} u_{ij}^{m} x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \quad (3.4)$$

U matrisi 3.5 nolu denklem kullanılarak hesaplanan küme merkezlerine göre tekrar hesaplanır. Fark ε'dan küçük olana kadar eski U matrisi ile yeni U matrisi karşılaştırılır [60].

$$u_{ij}=\frac{1}{\sum_{k=1}^{c}\left(\frac{\|x_i - c_i\|}{\|x_i - c_k\|}\right)^{2/(m-1)}} \tag{3.5}$$

Kümeleme işlemi tamamlandığında bulanık değerler içeren U üyelik matrisi kümelemenin sonucunu gösterir. İstenirse, bu değerler yuvarlanarak 0 ve 1'lere dönüştürülebilir.

Bulanık c-means algoritmasının adımları:

Adım 1: Küme sayısı c, bulanıklık indeksi m, üyelik derecesi matrisi V ya da U küme prototipleri rastgele oluşturulur.

Adım 2: U küme prototipleri rastgele oluşturulduğu varsayılırsa bu değerler kullanılarak üyelik derecesi matrisi hesaplanır.

Adım 3: U küme prototipleri güncellenir.

Adım 4: $|U(t)- U(t-1)| < \alpha$ ise algoritma sonlandırılır aksi taktirde Adım 2'ye geri dönülür.

3.3. K-means++

K-Means isminde geçen k parametresinden de anlaşılacağı üzere giriş veri kümesini k adet merkezli kümelere ayıran bir algoritmadır [61]. Veriler k-means algoritmasında sadece bir kümeye ait olmalıdır. Algoritmanın ilk adımında n adet veri k adet kümeye bölünür. Bölme işlemindeki amaç, bölünme tamamlandıktan sonra oluşan kümelerin, küme içindeki benzerliklerini maksimum ve kümeler arasındaki benzerlikleri ise minimum yapmaktır [54].Veri kümesinde bulunan her nokta, kendisine komşu olduğu en yakın merkez noktanın bulunduğu kümeye atanmaktadır. Kümenin merkez değeri içerisinde bulunan noktaların ortalaması alınarak hesaplanmaktadır [63]. İlgili hesaplama, merkezin değeri değiştiğinde biter.

K-means ++ algortiması, David Arthur ve Sergei Vassilvitskii [64] tarafından önerilen ve k-means kümeleme algoritması için kümelerin merkezini başlatan bir algoritmadır. K-means ++ algoritması, düzgün olmayan bir dağılıma göre her bir yinelemede bir nokta seçer ve k yinelemelerini kullanır. K-means++ 'nın başlangıç noktası, merkez noktaları arasındaki mesafeyi olabildiğince uzağa dönüştürmek ve böylece global optimal değeri elde etmek için rastgele seçilir. K-means++ algoritması, kümeleme sonuçlarının doğruluğu ve kararlılığı bakımında geleneksel k-means algoritmasına göre daha başarılı olduğu söylenebilir [65]. K-means++ algoritmasının adımları şöyledir:

$X \subset R^d$ veri noktaları kümesi ve D (x), $x \in X$ noktası ile en yakın küme merkezi arasındaki mesafe olsun.

Adım 1: X'den rastgele bir küme merkezi olarak S_1 seçilir.

Adım 2: Sonraki küme merkezi 3.6'da gösterilen denklem kullanılarak S_i seçilir. $D(x)^2$ x veri noktasından en yakın küme merkezine olan uzaklığı temsil eder.

$$S_i = \frac{D(x')^2}{\sum_{x=X} D(x)^2} \tag{3.6}$$

Adım 3: K küme merkezi seçilene kadar adım 2 tekrar edilir.

Adım 4: Kümeleme, k-means algoritmasının ilk kümeleme merkezi olarak seçilen K küme merkezleri kullanılarak gerçekleştirilir.

3.4. K-medoids

K-Medoids algoritmasında küme merkezi seçilirken, noktaların birbirine olan uzaklıkları hesaplandıktan sonra her bir noktanın diger noktalara olan uzaklıkları toplanır. Diğer noktalara en yakın nokta minimum toplam mesafeye sahip nokta olduğu için bu nokta küme merkezi olarak belirlenir. Kümeleme yöntemleri tarafından kullanılan sorgu metinleri arasındaki uzaklık, her bir sorgu çiftinin benzerliği ile ters orantılıdır. Denklem 3.7'de sorgular arasındaki uzaklık gösterilmiştir.

uzaklık = 1 - benzerlik (3.7)

Aykırı verilere karşı K-Medoids algoritması K-Means algoritmasına göre daha toleranslıdır. Kümenin diğer elemanlarından çok daha uzak olan noktalar aykırı veri olarak ifade edilir. Küme merkezi hesaplarına dahil edilen aykırı veriler, Öklit uzaklığını kullanarak k-means algoritmasıyla kümenin gerçek merkezinden aykırı veriye doğru yön değiştirerek yeni bir merkez belirleyecektir. Bu durum, gerçekte o kümede olması gereken elemanların farklı kümelere atanmasına neden olacaktır. K-Medoids algoritmasında aykırı verilerden daha az etkilenecek olan nokta küme elemanları içerisinden seçilen orta noktadır.

K-Medoids, küme sayısını parametre olarak en başta alır. Veri setindeki n adet noktadan rastgele seçilen k adet nokta başlangıç küme merkezleri (medoid) olarak atanır. k adet küme, veri setinde kalan diğer tüm noktaların kendisine en yakın medoid ile birleştirilmesiyle oluşturulur. Aynı kümede bulunan noktaların diğer noktalara olan uzaklıkları hesaplanarak uzaklıklar toplamı alınır. Minimum uzaklıklar toplamına sahip olan küme elemanı yeni medoid olarak atanır. İterasyon sayısı tamamlanana kadar veya medoid değerleri değişmeyene kadar kümeleme adımları yinelenir. Bu yöntemde, k parametresi olarak önceden verilen küme sayısı, metinsel sınıflandırmalarda karmaşıklığa yol açmaktadır. Metin kümelemesi sonucunda oluşacak küme sayılarının önceden tahmin edilmesi zor olduğu için aynı veri seti üzerinde çalıştırılan yoğunluk tabanlı yöntem sonucu oluşan küme sayıları, k parametresinin belirlenmesinde referans alınmaktadır [66,67].

4. DEĞERLENDİRME METRİKLERİ

Kümeleme analizinde kullanılan farklı algoritmalarla veya bu algoritmalardaki farklı parametre değeri ile farklı sayıda ve özelliklerde kümeleme yapısı elde edilebilir. Aynı veri kümesinden elde edilen farklı kümeleme yapılarının anlamlı olup olmadığı kümeleme geçerliliği olarak adlandırılır [68].

Çoğu veri kümesi için başlangıçta küme sayısı belli değildir. Kümelenme sonuçları bir kümeleme algoritması ile elde edildiğinde, bir sonraki önemli adım, modelin kümelerin sayısı için optimal bir çözüm veya küme yapısını belirleyip belirlemediğinin değerlendirilmesidir [69].

Farklı küme yapılarının anlamlı olup olmadığının değerlendirilmesi kümeleme geçerliliği olarak adlandırılmış olup, analiz sonucu oluşan küme yapılarının başarısını ölçebilmek için çeşitli değerlendirme metrikleri geliştirilmiştir. Böylece küme yapısının ve sayısının ne olması gerektiği tespit edilebilmekte ve alınan kararlar küme geçerlilik kriterleri ile desteklenebilmektedir [70].

4.1. Kullanılan Kümeleme Değerlendirme Kriterler

Bu tez kapsamındaki yapılan çalışmada, Davies - Bouldin [71], Silhouette [72] ve Calinski - Harabasz [73] indekslerinden faydalanılmıştır.

4.1.1. Davies bouldin (DB) indeksi

DB indeksini hesaplamak için kullanılan formül 4.1'de verilmiş olup, burada N küme sayısını, S_i küme i'deki tüm veri noktalarının M_i'ye olan ortalama mesafesini göstermektedir. DB indeksini en aza indiren küme düzeni optimum küme sayısı olarak alınmıştır [74].

$$DB \equiv \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq i} \frac{\left(S_i + S_j\right)}{M_{i,j}} \tag{4.1}$$

Avantajları

1. Davies-Bouldin'in hesaplanması Silhouette indeksinin hesaplanmasından daha kolaydır.
2. İndeks değeri sadece veri setindeki miktar ve özellikllere göre hesaplanır [75].

Dezavantajları

1. Centroid mesafesinin kullanımı öklid uzayının uzaklık metriğini sınırlar.
2. Davies-Bouldin indeksi dışbükey kümeler için genellikle diğer küme kavramlarından daha yüksektir [75].

4.1.2. Silhouette indeksi

Silhouette katsayısı, her veri noktasının ne kadar iyi kümelendiğinin geometrik olarak ifade edilmesidir. Ayrıca bu yöntem, kümelerdeki tutarlılığın doğrulanması ve açıklanması için kullanılır [76]. Silhouette katsayısı -1 ile +1 arasındadır. Yüksek değer, kümedeki noktanın kendi kümesine çok iyi uyduğunu ve komşu kümelerle uyuşmadığını ifade eder. Veri noktalarının çoğu yüksek silhouette değerine sahipse, kümelenme yapısının uygun olduğunu söyleyebiliriz. Çoğunluk düşük veya negatif değere sahipse, kümeleme yapısı çok az sayıda küme veya tam tersi çok fazla küme içerebilir. Silhouette katsayısı, Manhattan veya Öklid mesafesi gibi herhangi bir metrik ile hesaplanabilir.

Silhouette değerini hesaplamak için kullanılan formül 4.2'de verilmiş olup, a(i), kendi kümesindeki diğer tüm noktalardan i noktasının ortalama mesafesini ve b(i) ise i'nin diğer kümelerdeki diğer tüm noktalara en küçük ortalama mesafesini gösterir.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i),b(i)\}} \tag{4.2}$$

Avantajları

1. Skor yanlış kümeleme için -1 ile çok yoğun kümeleme için +1 arasında sınırlandırılmıştır. Sıfır civarındaki puanlar çakışan kümeleri gösterir.
2. Kümeler yoğun ve iyi ayrıldığında skoru daha yüksektir. Bu durum standart bir küme kavramı ile ilgilidir [75].

Dezavantajları

1. Silhouette katsayısı dışbükey kümeler için genellikle diğer küme kavramlarından daha yüksektir.
2. Yüksek hesaplama karmaşıklığı: $O(n^2)$ [75].

4.1.3. Calinski harabasz (CH) indeksi

Calinski ve Harabasz k kümeye sahip bir kümelemenin geçerliliğini ölçebilmek için 6.3'deki formülü önermişlerdir.

$$CH(K) = \frac{iz(S_B)(n - K)}{iz(S_w)(K - 1)} \tag{4.3}$$

şeklinde ifade edilir. Burada $iz(S_B)$ ve $iz(S_W)$ sırasıyla kümeler arası ve küme içi matrislerinin iz değerleridir. Küme sayısı olarak K'nın bir seçimi için bu indeksi maksimum yapan değerin seçimi önerilir [77].

Avantajları

1. Kümeler yoğun ve iyi ayrıldığında skoru daha yüksektir. Bu durum standart bir küme kavramı ile ilgilidir.
2. Skorun hesaplanması hızlıdır [75].

Dezavantajları

1. Calinski-Harabasz indeksi dışbükey kümeler için genellikle diğer küme kavramlarından daha yüksektir [75].

5. BENZERLİK ÖLÇÜLERİ

İstatistik ve ilgili alanlarda, bir benzerlik ölçüsü veya benzerlik işlevi, iki nesne arasındaki benzerliği ölçen gerçek değerli bir işlevdir. Bir benzerlik ölçüsünün tek bir tanımı olmamasına rağmen, genellikle bu ölçüler bir anlamda mesafe metriklerinin tersidir. Benzer nesneler için katsayı değeri büyük olurken, çok farklı nesneler için bu değer sıfır veya negatif olabilir [62].

5.1. Pearson Benzerlik Katsayısı

Pearson benzerlik ölçütü sürekli değişkenler için yaygın olarak kullanılan bir benzerlik ölçüsüdür. i'nci ve j'nci nesneler arasındaki benzerliği ölçmek için kullanılan bu yöntem Denklem 5.1'deki gibi hesaplanmaktadır.

$$r_{ij} = \frac{\Sigma_{k=1}^{p}(x_{ik} - \bar{x}_{i.})(x_{jk} - \bar{x}_{j.})}{\left[\Sigma_{k=1}^{p}(x_{ik} - \bar{x}_{i.})^2 \Sigma_{k=1}^{p}(x_{jk} - \bar{x}_{j.})^2\right]^{1/2}} \tag{5.1}$$

Pearson benzerlik katsayısı -1 ile 1 arasında yer almaktadır. Maksimum değer olan 1 değeri pozitif en güçlü benzerliği gösterirken, -1 değeri ise en güçlü negatif benzerliği yansıtmaktadır.

Korelasyon katsayısı nesneler arasındaki benzerliği ölçmek için kullanılması durumunda veri matrisinin sütunları yerine satırları standartlaştırılmaktadır [78].

5.2. Cosine Benzerlik Ölçüsü

İki vektör arasındaki açısının kosinüsü hesaplanarak bulunur. Benzerliği ölçmek için kullanılan formül Denklem 5.2'de gösterilmiştir.

$$\text{sim}(A,B) = \cos(\ ) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\Sigma_{i=1}^{n} A_i B_i}{\sqrt{\Sigma_{i=1}^{n} A_i^2}\sqrt{\Sigma_{i=1}^{n} B_i^2}} \tag{5.2}$$

Bu benzerlik ölçütü -1 ile 1 arasında yer almaktadır. İki vektör arasındaki açının kosinüsü hesaplanarak nesneler arasındaki benzerlikler tespit edilebilir. Birbirlerine paralel olan iki vektörün kosinüs değeri büyük olacağından, bulunan benzerlik değeri de artacaktır. Bunun sonucunda iki nesnenin daha benzer olduğu söylenebilmektedir [78].

6. DENEYSEL ÇALIŞMALAR VE ÖNERİLEN YAKLAŞIM

Bu bölümde ilk olarak çalışmada kullanılan yazılım ve donanım ortamı, veri seti ve veri ön işleme aşamaları anlatılmıştır. Deneysel sonuçlar bölümünde ise 3 farklı veri seti kullanılarak, kelime temsil yöntemleri, kümeleme algoritmaları ve Temel Bileşenler Analizi boyut indirgeme yöntemleri ile yapılan deneysel çalışmalar ayrıntılı bir şekilde anlatılmıştır. Oluşturulan modellerin değerlendirilmesi aşamasında Davies Bouldin, Silhouette ve Calinski Harabasz indekslerinden yararlanılmıştır. Deneylerde kullanılan tüm yöntemler, parametre değerleri, değerlendirme metrikleri ve tüm bunlarla birlikte elde edilen sonuçlar karşılaştırmalı olarak verilmiştir.

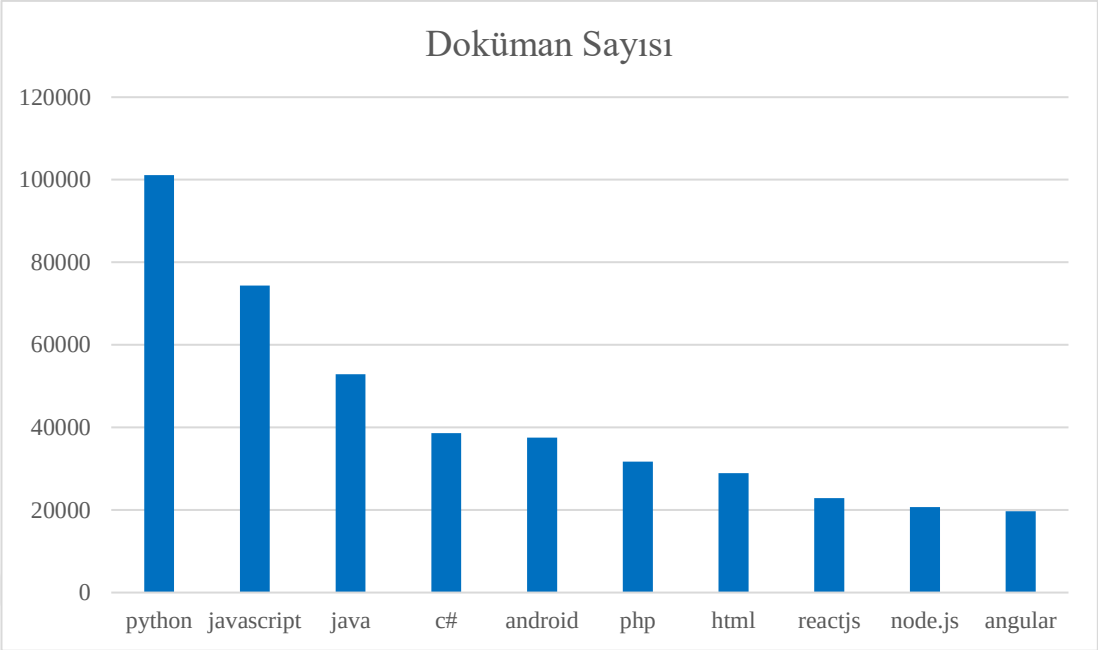
6.1. Yazılım ve Donanım Ortamı

Tez kapsamında tüm deneyler MacOS Catalina işletim sistemi üzerinde, 2.9 GHz 6 çekirdek Intel core i9 işlemcili ve 32 GB belleğe sahip bir bilgisayarda gerçekleştirilmiştir. Programlama dili olarak Python 3, IDE olarak Visual Studio Code kullanılmıştır. Veri toplama aşamasında stackapi kütüphanesi kullanılarak stackoverflow üzerinden veriler toplanmıştır. Veri ön işleme aşamasında durak kelimelerinin temizlenmesi için nltk.corpus kütüphanesi, kelime düzeltme işleminde SymSpell kütüphanesi ve kelimeleri köklerine ayırma işleminde SnowballStemmer kütüphanesi kullanılmıştır. Word2vec, fastText ve doc2vec yöntemleri için gensim kütüphanesi kullanılmıştır. Temel bileşenler analizi ve kümeleme algoritmalarında ise sklearn kütüphanesi kullanılmıştır.

6.2. Veri Kümesinin Oluşturulması

Veri toplama aşamasında, StackOverflow web sitesi aracılığıyla stackexchange API kullanılarak Ocak ve Ekim 2019 arasında toplam 617,347 örnek İngilizce veri toplanmıştır.

Toplanan verilerden en fazla soru sorulan 10 anahtar kelimeye belirlenmiştir. Bu 10 anahtar kelimenin hepsi programlama dilidir. Programlama dillerine göre doküman sayısının dağılımı Şekil 6.1'de gösterilmektedir.



Şekil 6.1. Doküman sayısı

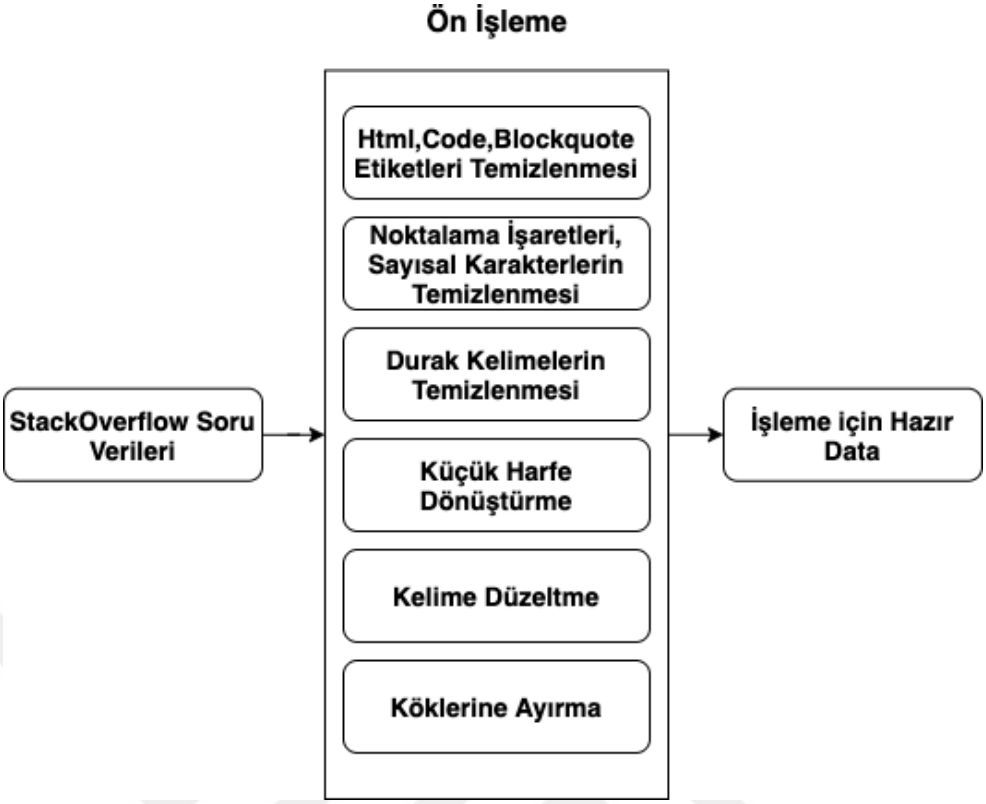
En çok sorulan ilk 3 programlama dili ile ilgili belgeler veri seti olarak hazırlanmıştır. Analizde kullanılan verilerin ayrıntılı bilgileri Tablo 6.1'de gösterilmiştir.

Tablo 6.1. Veri seti

Anahtar Kelime	**Doküman Sayısı**
Python	101,111
Javascript	74,357
Java	52,885

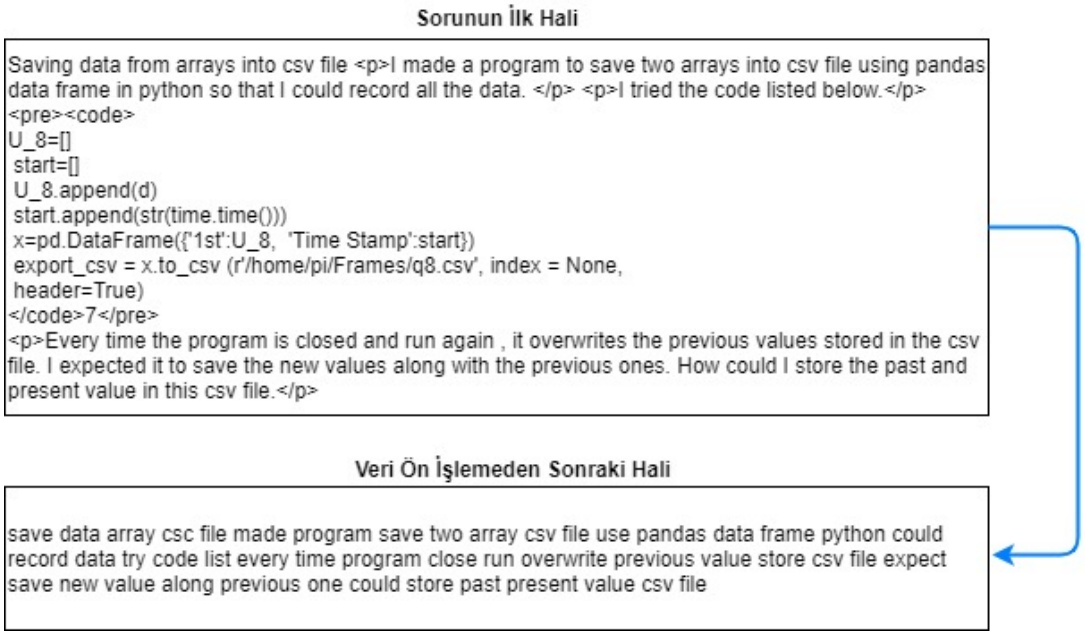
6.3. Veri Ön İşleme

Veri ön işleme aşamasında dokümanlardaki bağlantılar, sayısal karakterler, noktalama işaretleri, durdurma sözcükleri, blok alıntı ve kod etiketleri kaldırılmış olup, tüm kelimeler de küçük harfe dönüştürülmüştür. Sonraki aşamada ise, Python Symspell kütüphanesi kullanılarak kelime düzeltme işlemi gerçekleştirilmiştir. Son olarak, modelin başarı oranını arttırmak için Python snowballstemmer kütüphanesi kullanılarak İngilizce kelimeler köklerine ayrılmıştır. Köklerine ayrıştırma işleminden sonra ise bazı kelimelerde hata olduğundan dolayı Symspell kütüphanesi ile kelime düzeltme işlemi tekrar uygulanmıştır. Tüm veri ön işleme adımları Şekil 6.2'de gösterilmiştir.



Şekil 6.2. Veri ön işleme aşamaları

Veri ön işleme adımlarının sonucunda oluşan soru dokümanı örneği Şekil 6.3'te gösterilmiştir.



Şekil 6.3. Örnek bir soru üzerinde veri ön işleme adımlarının uygulanması

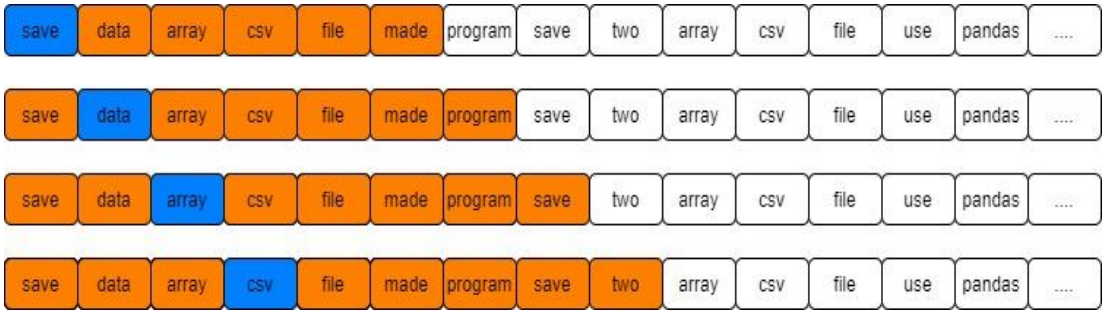
6.4. Deneysel Sonuçlar

Doküman kümeleme çalışması kapsamında 3 farklı kelime temsil yöntemi 4 farklı kümeleme algoritması kullanılarak deneysel çalışmalar yapılmıştır. Deneylerin sonucu ise 3 farklı değerlendirme metriği baz alınarak değerlendirilmiştir.

Model eğitilirken her bir eğitim adımında en uygun ağırlık değerleri hesaplanmaya çalışılır. Bu eğitim adımlarının her biri epoch değeri olarak adlandırılır. Modelin eğitilmesi işlemi uzun zaman alabilmektedir. Bu nedenle sinir ağı modellerini eğitmek için önemli parametrelerden birisi epoch sayısıdır. Epoch sayısını çok fazla veya çok az olması model verilerine uymayabilir ve düşük performanla karşılaşılabilir. Bu nedenle optimal değeri yakalamak önemlidir [79]. Çalışma kapsamında ilk önce Doc2Vec, Word2Vec ve fastText temsil yöntemlerinin epoch sayısına bağlı olarak performans değişimi ölçülmüştür. Doc2Vec ve fastText modellerinde düşük epoch sayısında başarı daha yüksek olurken, Word2Vec modelinde epoch sayısı daha yüksek değerlerde daha yüksek sonuç verdiği gözlemlenmiştir. Bir diğer önemli parametre ise vektör boyutudur. Vektör, belgeyi n boyutlu bir uzayda bir noktaya eşler. Boyutlar büyüdükçe dokümanlar arasında daha fazla ayrım yapılabilir. Yapılan deneysel çalışmalarda Doc2Vec, Word2Vec ve fastText modelleri için en uygun vektör boyutu araştırılmıştır. 25 ve 250 arasındaki her bir epoch değeri için, 25'lik artışlarla, modeller 10 kez eğitilip ve sonuçları değerlendirilmiştir. Bu işlem 3 farklı veri seti için de yapılmış olup, 3 farklı veri setinde 3 model için en uygun epoch sayısı, pencere ve vektör boyutu parametreleri Tablo 6.2'de gösterilmiştir.

Tablo 6.2. Model eğitim parametreleri

Veri Seti	Model	Epoch	Pencere Boyutu	Vektör Boyutu
Java Soruları	Doc2Vec	25	8	200
	Word2Vec	100	5	200
	fastText	50	5	100
Python Soruları	Doc2Vec	25	8	200
	Word2Vec	100	5	200
	fastText	50	5	100
Javascript Soruları	Doc2Vec	25	8	200
	Word2Vec	100	5	200
	fastText	50	5	100



Şekil 6.4. Word2Vec CBOW modeli için soru dokümanı pencere boyutu örneği

Tez çalışmasında kullanılan Word2Vec CBOW modeli için en uygun pencere boyutu 5 olarak tespit edilmiştir. Şekil 6.4.'te pencere boyutu 5 olan CBOW modelinde ilk olarak "save" kelimesi merkez kelime olarak seçilmiştir. Daha sonra sağındaki ve solundaki 5'er kelimeyi ayrı ayrı girdi olarak alıp "save" kelimesi tahmin edilmeye çalışılmıştır.

Word2vec matrisinde kullanılan word2vec modelleri, sürekli kelime torbası (CBOW) yöntemi [22], 200 vektör boyutu, 100 epoch, 5 pencere boyutu ve minimum kelime sayısı 5 ile eğitilmiştir. Hiper parametrelerin varyasyonları, 2 ila 10 arasında değişen penceresi boyutu, daha yüksek vektör boyutu ve minimum kelime sayısı ile test edilmiştir. Vektör boyutu ve epoch arttıkça silhouette ve CH skorunun arttığı, DB skorunun azaldığı tespit edilmiştir. Diğer yandan, deneyde epoch değerini 25'ten 250'ye 25'erlik artışlarla yükselttiğimizde optimal epoch değeri 100 olarak belirlenmiştir. Python verileri üzerinde word2vec, temel bileşenler analizi yöntemi ve bulanık c-means algoritmalarının beraber kullanılması sonucunda epoch değerine göre kümeleme skor değişimleri Tablo 6.3'de gösterilmiştir.

Tablo 6.3. Word2vec ve bulanık c-means algoritmasının epoch değerlerine göre kümeleme skor değişimleri

Epoch	Pencere Boyutu	Vektör Boyutu	Silhouette Skoru	DB Skoru	CH Skoru
25	5	200	0,11901096	2,2304030983	1072,92837077
50	5	200	0,11944699	2,2166256158	1202,68450351
75	5	200	0,12314897	2,2077236199	1266,17087147
100	5	200	**0,12540242**	**2,1944785881**	**1295,65702974**
125	5	200	0,12514367	2,1984343966	1287,93660419
150	5	200	0,12026462	2,2020771419	1234,79361732
175	5	200	0,11630782	2,2270379558	1151,63012083

Tablo 6.3. (Devam) Word2vec ve bulanık c-means algoritmasının epoch değerlerine göre kümeleme skor değişimleri

Epoch	Pencere Boyutu	Vektör Boyutu	Silhouette Skoru	DB Skoru	CH Skoru
200	5	200	0,11787761	2,2230183088	1169,16409937
225	5	200	0,11519113	2,2323630971	1087,28009376
250	5	200	0,11332997	2,2391541773	1036,65013966

FastText, dağıtılmış kelime temsilini ve kelime sırasını dikkate almak için bir yöntem kombinasyonu kullanan kelime temsil modelidir. [80]. Algoritma denetimli ve denetimsiz olarak çalışır. Denetimli modda, dokümanlar temsillerin ortalaması alınarak vektörlere dönüştürülür. Ayrıca, fastText, denetimsiz modda genel amaçlar için kelime temsilleri oluşturur ve sınıfları dikkate almaz [81].

FastText ile yapılan deneyde, 100 vektör boyutu, 50 epoch, pencere boyutu olarak 5 ve minimum kelime sayısı 5 ile hiper parametrelerin varyasyonlarını test edilmiştir. Bu deneyde epoch'un azalması durumunda silhouette ve CH skorunun arttığı, DB skorunun azaldığı tespit edilmiştir. Diğer yandan, deneyde epoch değerini 25'ten 250'ye 25'erlik artışlarla yükselttiğimizde optimal epoch değeri 50 olarak belirlenmiştir. Python verileri üzerinde fastText, temel bileşenler analizi yöntemi ve bulanık c-means algoritmalarının beraber kullanılması sonucunda epoch değerine göre kümeleme skor değişimleri Tablo 6.4'de gösterilmiştir.

Tablo 6.4. FastText ve bulanık c-means algoritmasının epoch değerlerine göre kümeleme skor değişimleri

Epoch	Pencere Boyutu	Vektör Boyutu	Silhouette Skoru	DB Skoru	CH Skoru
25	5	100	0,14470012	1,9181242302	1585,43389839
50	5	100	**0,15731099**	**1,7134240222**	**1631,59846075**
75	5	100	0,15660173	1,7896058031	1612,67643104
100	5	100	0,15379301	1,8293905547	1603,48466003
125	5	100	0,15090335	1,8717454841	1596,95130874
150	5	100	0,14637071	1,9004810804	1589,88301159
175	5	100	0,14169004	1,9269632771	1571,18671467
200	5	100	0,13733098	2,0296394109	1563,04605465
225	5	100	0,13288159	2,0964833655	1540,54056639
250	5	100	0,12574109	2,1253857105	1526,99032844

Doc2vec, bir corpustaki her bir doküman için sabit boyutta yoğun bir vektör oluşturmak üzere eğitilmiş bir sinir ağı temelli temsil yöntemidir. Doc2vec ile yapılan deneyde dağıtılmış bellek yöntemiyle her doküman için 200 vektör boyutu, 25 epoch ve pencere boyutu olarak 8 kullanılmıştır. Deneyde epoch değeri 25'ten 250'ye 25'erlik artışlarla yükseltilmiştir. Sonuç itibariyle epochun azalması durumunda silhouette ve CH skorunun arttığı, DB skorunun ise azaldığı tespit edilmiştir. Optimal epoch değeri 25 olarak belirlenmiştir. Python verileri üzerinde doc2vec, temel bileşenler analizi yöntemi ve bulanık c-means algoritmalarının beraber kullanılması sonucunda epoch değerine göre kümeleme skor değişimleri Tablo 6.5'de gösterilmiştir.

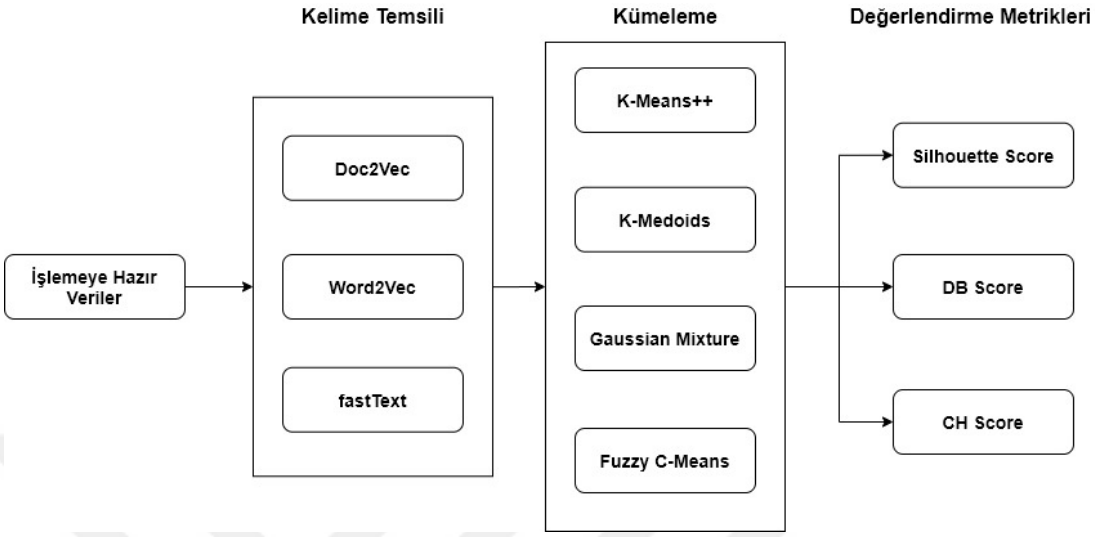
Tablo 6.5. Doc2vec ve bulanık c-means algoritmasının epoch değerlerine göre kümeleme skor değişimleri

Epoch	Pencere Boyutu	Vektör Boyutu	Silhouette Skoru	DB Skoru	CH Skoru
25	8	200	**0,2289602**	**1,5533075461**	**30650,4775533**
50	8	200	0,1914204	2,1674904048	23030,4021821
75	8	200	0,1739097	2,4793723767	19738,5598986
100	8	200	0,1353102	3,1748579341	17193,5774809
125	8	200	0,1241993	3.4378760394	16391,3075018
150	8	200	0,1183006	3.7778760301	15296,8357661
175	8	200	0,0961138	4.8237876038	11399,6260814
200	8	200	0,0869203	4.4378760344	8672,14676607
225	8	200	0,0842465	4.0787609393	7339,13006874
250	8	200	0,0796374	5.8630787603	6092,76314075

Modelin başarısı için önemli parametrelerden biri vektör boyutudur. Vektör, belgeyi n boyutlu bir alanda bir noktaya eşler ve boyutlar büyüdükçe belgeler arasında daha fazla ayrım yapılabilir. Doc2Vec ve Word2Vec için vektör boyutu arttıkça, modelin başarısı artarken, FastText'teki modelin başarısı düşük vektör boyutunda daha yüksektir. Doc2Vec ve Word2Vec için optimum vektör boyutu değeri 200 olup, bu değer FastText için 100'dür.

Deneylerde ilk önce, word2vec, doc2vec ve fasttext modellerinin epoch değişimleri ile performans değişimi araştırılmıştır. Sonrasında, temsil yöntemleri ile birlikte en iyi sonucu veren modeli bulmak için k-means++, k-medoids, gaussian mixture ve bulanık c-means kümeleme algoritmaları ile deneyler yapılmıştır. Bu deneyler Python, Java ve

Javascript veri setleri için ayrı ayrı uygulanmıştır. Sistem mimarisi Şekil 6.5'te gösterilmiştir.



Şekil 6.5. Önerilen sistem mimarisi

Kelime temsil yöntemleri ile vektör haline dönüştürülen Python, Java ve Javascript verileri k-means++, k-medoids, gaussian mixture ve bulanık c-means algoritmaları kullanılarak kümelenmiştir. Oluşan kümeler silhouette, DB, ve CH skorlarına göre değerlendirilmiştir. Kümeleme algoritmalarının DB skorunun daha düşük, CH ve silhouette skorunun daha yüksek olması kümelemenin daha başarılı olduğunu gösterir. DB skoru küme içi mesafenin kümeler arası mesafelere oranıdır. CH skoru ise kümeler arası uzaklıkların kareleri toplamının küme içi uzaklıkların kareleri toplamına oranıdır. Silhouette skoru ise bir noktanın diğer kümelere kıyasla kendi kümesine ne kadar benzediğini göstermektedir.

Java veri setinde Word2vec yöntemi ve K-means++ algortimasının birlikte uygulanmasıyla DB skoru 7,639046527 elde edilmiştir. Elde edilen bu değer diğer algoritmalara göre daha düşük yani başarılıyken CH skoru 108,68963147 olup, diğer algoritmalara göre en yüksek skor elde edilmiştir. Word2Vec yöntemiyle bulanık c-means algoritmasının birlikte kullanılmasıyla silhouette skoru 0,047850814 olarak ölçülmüş olup bu skor diğer algoritmalardan daha yüksektir.

Javascript veri setinde Word2Vec yönteminin K-means++ algoritması ile birlikte kullanılmasıyla DB skoru 8,027184053 olarak ölçülmüştür. Bu skor aynı veri setinde diğer algoritmalara göre daha düşük yani başarılıyken CH skoru 83,923222372 olarak

ölçülmüş ve bu skor diğer algoritmalara göre daha yüksektir. Word2Vec yönteminin Gaussian mixture algoritmasıyla birlikte kullanılmasıyla silhouette skoru 0,03829827 olarak hesaplanmıştır. Bu değer diğer algoritmalardan daha yüksektir.

Python veri setinde Word2Vec yönteminin K-means++ algortiması ile birlikte kullanılmasıyla DB skoru 7,3327440298 olarak hesaplanmıştır. Bu skor diğer algoritmalara göre daha düşük yani başarılıdır. Silhouette skoru 0,040494468 olarak bulunmuş olup, bu skor diğer algoritmalara göre daha yüksek yani başarılıdır. Word2Vec yönteminin bulanık c-means algoritmasıyla birlikte kullanımıyla CH skoru 109,71888332 olarak hesaplanmış ve bu değer diğer algoritmalardan daha yüksektir.

Deneylerde kullanılan 3 farklı veri setinde doküman sayıları da farklıdır. Word2Vec ve 4 kümeleme algoritmasıyla 3 farklı veri seti üzerinde yapılan deneylerde, doküman sayısının artmasıyla Word2Vec ve K-means++ algoritmasının kullanımıyla küme içindeki noktaların birbirleriyle benzerliklerinin ve diğer kümelerle olan uzaklığının arttığı görülmüştür. Word2vec modelinin bulanık c-means ve k-medoids algoritmaları ile kullanılmasıyla yapılan deneyde ise en yüksek başarı Python veri setinde olmuştur. Gaussian mixture algoritmasında ise sonuçlar 3 veri setinde de birbirinde farklıdır. Sonuç olarak word2vec yönteminde k-means++ kümeleme algoritmasının kullanılmasıyla daha başarılı kümeler elde edilmiştir. Bunun sebebi küme içindeki noktaların birbirine olan benzerliklerinin diğer kümelerdeki noktalara göre daha yüksek olmasıyla ve kümelerin birbirleriyle olan uzaklarının diğer algorirmalara göre daha fazla olmasıyla açıklanabilir.

Java veri setinde FastText yöntemi ve bulanık c-means algoritmasının birlikte kullanılmasıyla silhouette ve CH skoru diğer algoritmalara göre daha yüksek iken DB skoru diğer algoritmalara göre daha düşüktür yani daha başarılıdır. Bulanık c-means algoritmasının silhouette skoru 0,046082705, DB skoru 4,48052626970 ve CH skoru 284,965650337 olarak ölçülmüştür.

Javascript veri setinde fastText yöntemi ve k-means++ algortimasının birlikte kullanılmasıyla silhouette ve CH skoru diğer algoritmalara göre daha yüksek iken DB skoru diğer algoritmalara göre daha düşüktür yani daha başarılıdır. K-means++ algoritmasının silhouette skoru 0,03905312, DB skoru 6,02983390344 ve CH skoru 159,24636644 olarak ölçülmüştür.

Python veri setinde fastText yöntemi ve k-means++ algortimasının birlikte kullanılmasıyla silhouette ve CH skoru diğer algoritmalara göre daha yüksek iken DB skoru diğer algoritmalara göre daha düşüktür yani daha başarılıdır. K-means++ algoritmasının silhouette skoru 0,021525037 DB skoru 5,37800917587 ve CH skoru 185,93377172 olarak ölçülmüştür.

FastText ve 4 kümeleme algoritmasıyla 3 farklı veri seti üzerinde yapılan deneylerde, doküman sayısının artmasıyla k-means++ algoritmasında küme içindeki noktaların birbirleriyle benzerliklerinin azaldığı görülmüştür. FastText modelinin bulanık c-means ve k-medoids algoritmaları ile birlikte kullanılmasıyla yapılan deneyde ise en yüksek başarı Java veri setinde olmuştur. Gaussian mixture algoritmasında ise küme içi noktaların benzerlikleri ve kümeler arası uzaklıklar düşüktür. Sonuç olarak fastText yönteminde k-means++ ve bulanık c-means kümeleme algoritmalarının skorları incelendiğinde 2 algoritmanın skorları birbirlerine hemen hemen yakın olup daha başarılı kümeler elde edilmiştir. Bunun sebebi küme içindeki noktaların birbirine olan benzerliklerinin diğer kümelerdeki noktalara göre daha yüksek olmasıyla ve kümelerin birbirleriyle olan uzaklarının diğer algorirmalara göre daha fazla olmasıyla açıklanabilir.

Java veri setinde Doc2Vec yöntemi ve k-means++ algoritmasının birlikte kullanılmasıyla DB skoru 4,37059696784 olarak ölçülmüştür. Bu skor diğer algoritmalara göre daha düşük yani daha başarılıdır. Yine K-means++ algoritmasının CH skoru 2139,14811148 olarak ölçülmüş olup diğer algoritmalara göre daha yüksektir. Doc2Vec yönteminin Gaussian mixture algoritmasıyla birlikte kullanılmasıyla silhouette skoru 0,10659909 ölçülmüş olup, bu skor diğer algoritmalardan elde edilen skorlardan daha yüksektir.

Javascript veri setinde Doc2Vec yönteminin k-means++ algortimasıyla birlikte kullanılmasıyla DB skoru 4,76598785848 olarak ölçülmüş olup, bu skor diğer algoritmalara göre daha düşük yani daha başarılıdır. Yine K-means++ algoritmasının CH skoru 2714,00157830 olarak ölçülmüş olup, bu skor diğer algoritmalara göre daha yüksektir. Gaussian mixture algoritmasının Doc2Vec ile kullanılmasıyla silhouette skoru 0,09895941 ölçülmüş olup, bu skor diğer algoritmalardan daha yüksektir.

Python veri setinde Doc2Vec algoritmasının k-means++ algoritmasıyla birlikte kullanılmasıyla DB skoru 4,8058380393 olarak ölçülmüş olup, bu skor diğer algoritmalara göre daha düşük yani daha başarılıdır. Yine K-means++ algoritmasının CH skoru 3548,42058300 ölçülmüş olup, diğer algoritmalara göre daha yüksek bir değer elde edilmiştir. Gaussian mixture algoritmasının Doc2Vec yöntemiyle birlikte kullanılmasıyla silhouette skoru 0,114848055 ölçülmüş olup, bu skor diğer algoritmalardan daha yüksektir.

Doc2Vec ve 4 kümeleme algoritmasıyla 3 farklı veri seti üzerinde yapılan deneylerde, doküman sayısının artmasıyla k-means++ algoritmasında küme içindeki noktaların birbirleriyle benzerliklerinin azaldığı görülmüştür. Doc2vec modelinin bulanık c-means ve k-medoids algoritmaları ile kullanılmasıyla yapılan deneyde ise en yüksek başarı Java veri setinde olmuştur. Gaussian mixture algoritmasında ise küme içi noktaların benzerlikleri diğer algoritmalara göre daha yüksektir. Sonuç olarak doc2vec yönteminde gaussian mixture algoritmasında küme içi noktaların benzerlikleri yüksek olup k-means++ algoritmasında kümelerin birbirleriyle olan uzaklıkları daha yüksektir. Bulanık c-means ve k-means++ algoritmalarının kümeleme skorları birbirlerine yakındır. Tüm veri setlerine ait kümeleme skorları Tablo 6.6, Tablo 6.7 ve Tablo 6.8'de detaylı olarak gösterilmiştir.

Tablo 6.6. Python veri setine ait kümeleme skorları

Özellik Gösterimi	Kümeleme Algoritması	Silhouette Skoru	DB Skoru	CH Skoru
Doc2Vec	K-Means++	0,051210795	**4,8058380393**	**3548,42058300**
	K-Medoids	0,032164106	5,0148237613	3298,4308891
	Gaussian Mixture	**0,114848055**	16,4701857471	327,526274165
	Bulanık C-Means	0,04469543	4,80878036857	3528,02328740
Word2Vec	K-Means++	**0,040494468**	**7,3327440298**	98,636932026
	K-Medoids	0,008342793	13,0225250665	28,736541345
	Gaussian Mixture	0,014674502	12,037839418	52,64006280
	Bulanık C-Means	0,0119269155	8,15893911838	**109,71888332**
fastText	K-Means++	**0,021525037**	**5,37800917587**	**185,93377172**
	K-Medoids	0,0056469887	9,84681786948	57,7360064885
	Gaussian Mixture	-0,021828536	8,2948405292	95,8266475889
	Bulanık C-Means	0,012207914	5,43927366255	177,370572034

Tablo 6.7. Java veri setine ait kümeleme skorları

Özellik Gösterimi	Kümeleme Algoritması	Silhouette Skoru	DB Skoru	CH Skoru
Doc2Vec	K-Means++	0,06838688	**4,37059696784**	**2139,14811148**
	K-Medoids	0,044066083	4,45102375108	2031,74117646
	Gaussian Mixture	**0,10659909**	11,9195187115	791,511729325
	Bulanık C-Means	0,0536236	4,39120242864	2118,55084670
Word2Vec	K-Means++	0,03131144	**7,639046527**	**108,68963147**
	K-Medoids	0,0046180845	13,9083265470	34,7995411703
	Gaussian Mixture	-0,029389992	9,25921454144	91,947370647
	Bulanık C-Means	**0,047850814**	7,94505054681	103,794171248
fastText	K-Means++	0,037033044	4,77583210009	223,403066312
	K-Medoids	0,009161554	9,53811005829	64,0642978006
	Gaussian Mixture	-0,037554566	16,2336919251	75,9356227267
	Bulanık C-Means	**0,046082705**	**4,48052626970**	**284,965650337**

Tablo 6.8. Javascript veri setine ait kümeleme skorları

Özellik Gösterimi	Kümeleme Algoritması	Silhouette Skoru	DB Skoru	CH Skoru
Doc2Vec	K-Means++	0,052172665	**4,76598785848**	**2714,00157830**
	K-Medoids	0,034015682	4,92001783639	2553,03896722
	Gaussian Mixture	**0,09895941**	17,1666303168	223,078107033
	Bulanık C-Means	0,045855563	4,77497205021	2695,58038652
Word2Vec	K-Means++	0,030102726	**8,027184053**	**83,923222372**
	K-Medoids	0,0032026912	15,2630728763	24,4869098910
	Gaussian Mixture	**0,03829827**	22,35105576	3,22494998
	Bulanık C-Means	0,0130492635	8,5900460812	77,446854515
fastText	K-Means++	**0,03905312**	**6,02983390344**	**159,24636644**
	K-Medoids	0,012441033	10,277095287	45,6851897722
	Gaussian Mixture	-0,022544546	8,690906774	60,743575117
	Bulanık C-Means	0,031592343	6,26527214047	155,0768793

Yüksek boyutlu verilerde maksimum varyansı bulmak bilgiyi korurken, daha küçük boyutlara indirgemek amacıyla boyut indirgeme teknikleri kullanılır. Tez kapsamında temsil yöntemlerinden elde edilen vektörler Temel Bileşenler Analizi yöntemi kullanılarak boyut indirgemesi yapılmış olup, 4 farklı kümeleme algoritmasıyla deneysel çalışmalar yapılmıştır.

Java veri setinde Word2vec yöntemi, temel bileşenler analizi ve bulanık c-means algortimasının birlikte kullanılmasıyla silhouette skoru 0,2116015, DB skoru 2,03369259478 ve CH skoru 1485,2497058 olarak ölçülmüştür. Java veri setinde en başarılı yaklaşım bu olmuştur.

Javascript veri setinde Word2Vec yöntemi, temel bileşenler analizi ve bulanık c-means algortimasının birlikte kullanılmasıyla silhouette skoru 0,12970085, DB skoru 2,3080778986 ve CH skoru 891,173435677 olarak ölçülmüştür. Javascript veri setinde en başarılı yaklaşım bu olmuştur.

Python veri setinde Word2vec yöntemi, temek bileşenler analizi ve k-means++ algortimasının birlikte kullanılmasıyla silhouette skoru 0,18451995 ve DB skoru 2,1706320593 olarak ölçülmüş olup kümeleme işlemi diğer algoritmalara göre daha başarılıdır. Yine Word2Vec yöntemi, temel bileşenler analizi ve bulanık c-means algoritmasının birlikte kullanılmasıyla CH skoru 1295,65702974 olarak ölçülmüş olup, bu skor diğer algoritmalara göre daha yüksektir.

Word2Vec yöntemi, temel bileşenler analizi ve 4 kümeleme algoritmasıyla 3 farklı veri seti üzerinde yapılan deneylerde, doküman sayısının artmasıyla k-means++ algoritmasında küme içindeki noktaların birbirleriyle benzerliklerinin arttığı ve bulanık c-means algoritmasında ise azaldığı görülmüştür. Deneyler sonucunda temel bileşenler analizi yönteminin k-means++ ve bulanık c-means kümeleme algoritmalarının kümeleme başarısını büyük oranda artırdığı gözlemlenmiştir. Fakat k-medoids ve gaussian mixture algoritmalarının kümeleme başarısını küçük bir oranda artırmıştır. Sonuç olarak word2vec yönteminde temel bileşenler analizi ve bulanık c-means kümeleme algoritmasının kullanılmasıyla daha başarılı kümeler elde edilmiştir.

Java veri setinde FastText yöntemi, temel bileşenler analizi ve bulanık c-means algortimasının birlikte kullanılmasıyla silhouette skoru 0,1941493, DB skoru 1,77026133979 ve CH skoru 1751,90188139 olarak ölçülmüştür. Java veri setinde en başarılı yaklaşım bu olmuştur.

Javascript veri setinde FastText yöntemi, temel bileşenler analizi yöntemi ve bulanık c-means algortimasının birlikte kullanılmasıyla silhouette skoru 0,17278989, DB

skoru 2,0952332643 ve CH skoru 1278,405028 olarak ölçülmüştür. Javascript veri setinde en başarılı yaklaşım bu olmuştur.

Python veri setinde FastText yöntemi, temel bileşenler analizi yöntemi ve bulanık c-means algortimasının birlikte kullanılmasıyla silhouette skoru 0,15731099, DB skoru 1,7134240222 ve CH skoru 1631,59846075 olarak ölçülmüştür. Python veri setinde en başarılı yaklaşım bu olmuştur.

FastText yöntemi, temel bileşenler analizi ve 4 kümeleme algoritmasıyla 3 farklı veri seti üzerinde yapılan deneylerde, doküman sayısının artmasıyla bulanık c-means algoritmasında küme içindeki noktaların birbirleriyle benzerliklerinin azaldığı görülmüştür. Deneyler sonucunda temel bileşenler analizi yönteminin k-means++ ve bulanık c-means kümeleme algoritmalarının kümeleme başarısını büyük oranda artırdığı gözlemlenmiş fakat k-medoids ve gaussian mixture algoritmalarının kümeleme başarısını küçük bir oranda artırmıştır. Sonuç olarak fastText yönteminde temel bileşenler analizi ve bulanık c-means kümeleme algoritmasının kullanılmasıyla daha başarılı kümeler elde edilmiştir.

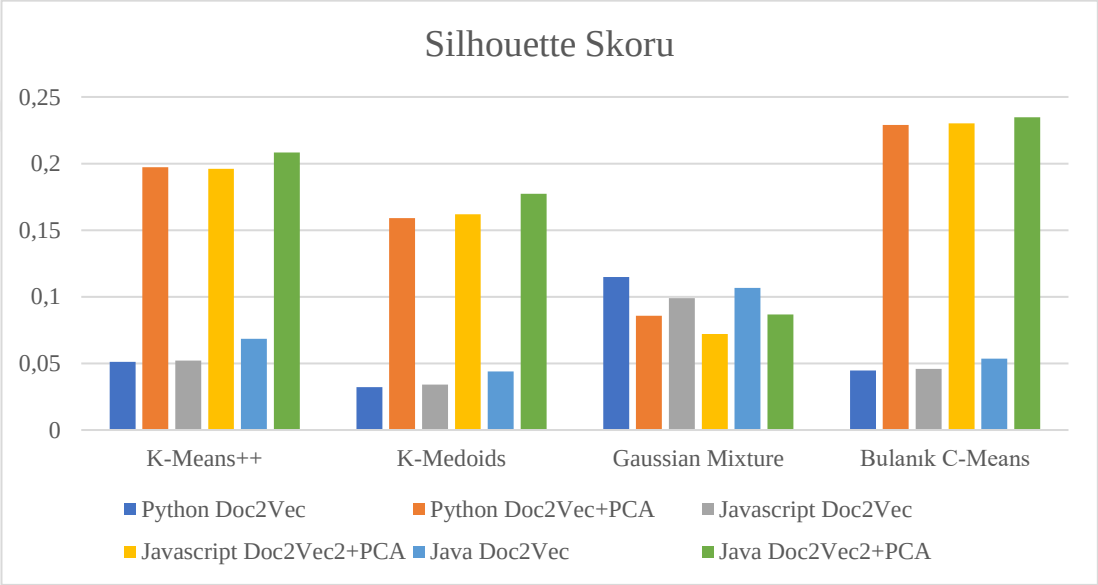
Java veri setinde Doc2vec yöntemi, temel bileşenler analizi ve bulanık c-means algoritmasının birlikte kullanılmasıyla silhouette skoru 0,23483634, DB skoru 1,51135170141 ve CH skoru 14319,8437998 olarak ölçülmüştür. Java veri setinde en başarılı yaklaşım bu olmuştur.

Javascript veri setinde Doc2vec yöntemi, temel bileşenler analizi ve bulanık c-means algoritmasının birlikte kullanılmasıyla silhouette skoru 0,23009749, DB skoru 1,5597675802 ve CH skoru 23362,7150831 olarak ölçülmüştür. Javascript veri setinde en başarılı yaklaşım bu olmuştur.

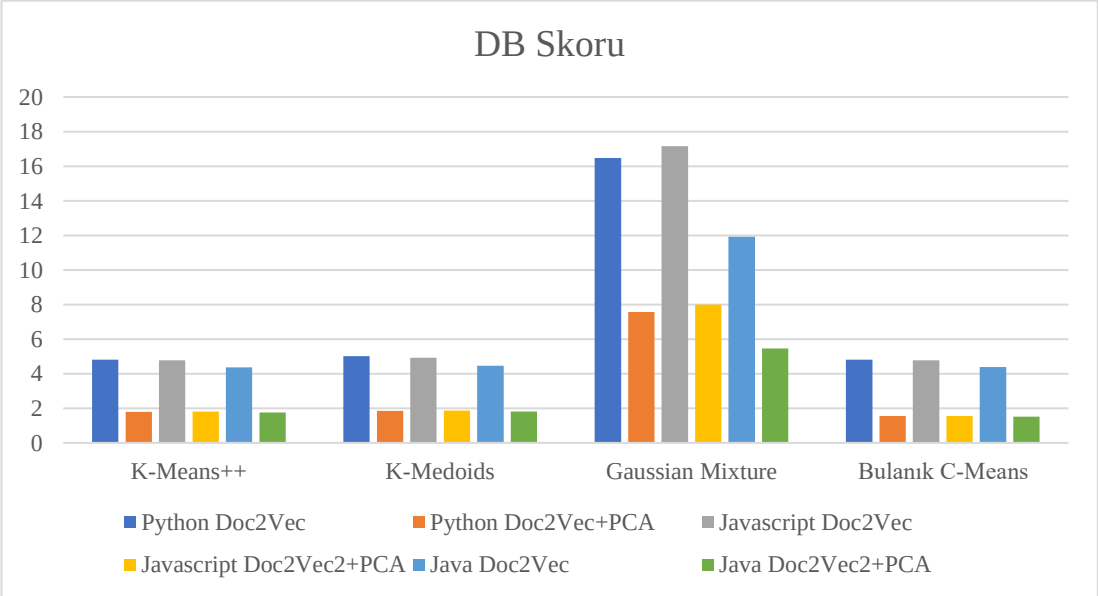
Python veri setinde Doc2vec yöntemi, temel bileşenler analizi ve bulanık c-means algoritmasının birlikte kullanılmasıyla silhouette skoru 0,2289602, DB skoru 1,5533075461 ve CH skoru 30650,4775533 olarak ölçülmüştür. Python veri setinde en başarılı yaklaşım bu olmuştur.

Doc2Vec yöntemi, temel bileşenler analizi ve 4 kümeleme algoritmasıyla 3 farklı veri seti üzerinde yapılan deneylerde, doküman sayısının artmasıyla bulanık c-means

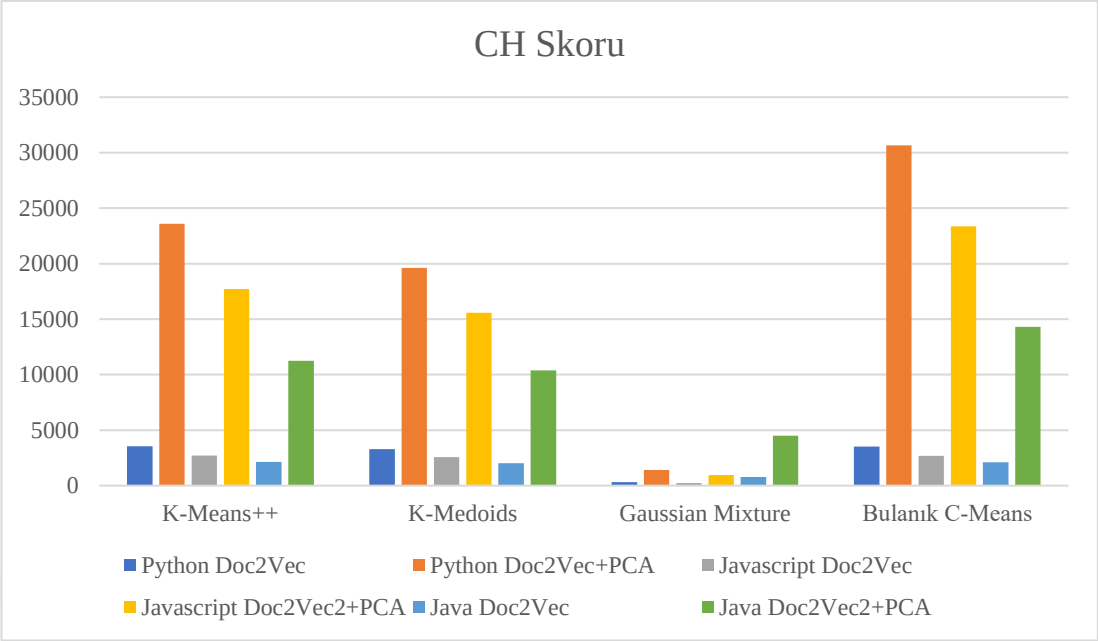
algoritmasında küme içindeki noktaların birbirleriyle benzerliklerinin azaldığı ve kümelerin birbirleriyle olan uzaklıklarının arttığı görülmüştür. Deneyler sonucunda temel bileşenler analizi yönteminin k-means++, bulanık c-means ve k-medoids kümeleme algoritmalarının kümeleme başarısını büyük oranda artırdığı gözlemlenmiş fakat gaussian mixture algoritmalarının kümeleme başarısını düşürdüğü gözlemlenmiştir. Temel bileşenler analizi yönteminin Doc2Vec temsil yöntemiyle birlikte kullanılmasıyla benzer dokümanları kümelemeye olan etkisi farklı veri setleri üzerinde de denenerek Şekil 6.6, Şekil 6.7 ve Şekil 6.8'de detaylı olarak gösterilmiştir.



Şekil 6.6. Doc2Vec ve PCA kullanılması sonucu Silhouette skor değişimleri



Şekil 6.7. Doc2Vec ve PCA kullanılması sonucu DB skor değişimleri



Şekil 6.8. Doc2Vec ve PCA kullanılması sonucu CH skor değişimleri

Sonuç olarak doc2vec yönteminde temel bileşenler analizi ve bulanık c-means kümeleme algoritmasının kullanılmasıyla daha başarılı kümeler elde edilmiştir. Tüm veri setlerine ait kümeleme skorları Tablo 6.9, Tablo 6.10 ve Tablo 6.11'de detaylı olarak gösterilmiştir.

Tablo 6.9. Pyton Veri setine ait PCA uygulanmış kümeleme skorları

Özellik Gösterimi	Kümeleme Algortiması	Silhouette Skoru	DB Skoru	CH Skoru
Doc2Vec + PCA	K-Means++	0,19736119	1,7894635810	23580,0780914
	K-Medoids	0,15910028	1,8505662819	19602,0528931
	Gaussian Mixture	0,08571174	7,5636469330	1410,91300213
	Bulanık C-Means	**0,2289602**	**1,5533075461**	**30650,4775533**
Word2Vec + PCA	K-Means++	**0,18451995**	**2,1706320593**	933,01874421
	K-Medoids	0,0048171054	7,1440914565	89,6482014279
	Gaussian Mixture	0,069923595	3,2886506597	585,265354102
	Bulanık C-Means	0,12540242	2,1944785881	**1295,65702974**
fastText + PCA	K-Means++	0,1336568	2,0376030367	1236,10095035
	K-Medoids	0,0190268	7,1231139336	140,87493744
	Gaussian Mixture	0,0246711	3,3929005773	653,872157682
	Bulanık C-Means	**0,15731099**	**1,7134240222**	**1631,59846075**

Tablo 6.10. Java veri setine ait PCA uygulanmış kümeleme skorları

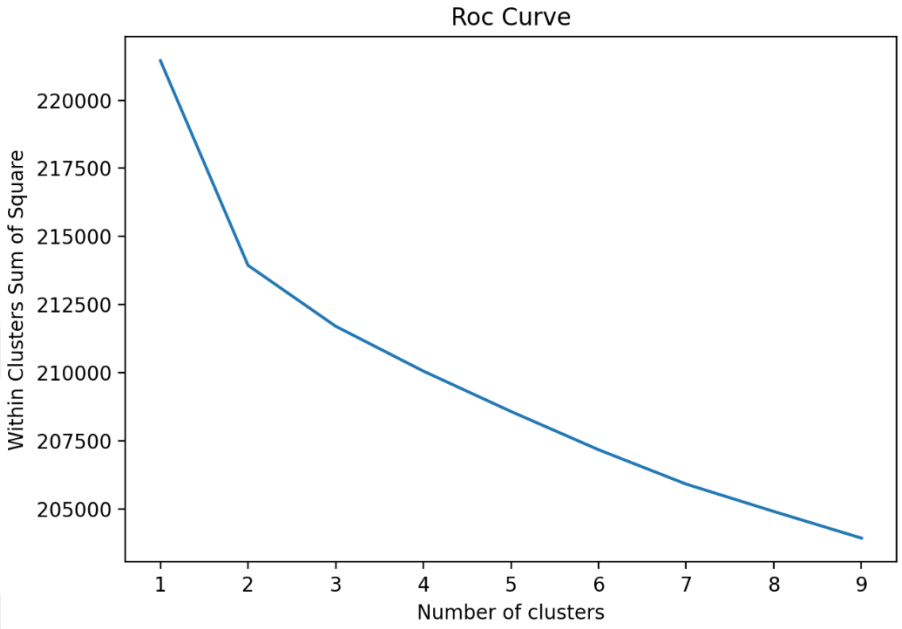
Özellik Gösterimi	Kümeleme Algortiması	Silhouette Skoru	DB Skoru	CH Skoru
Doc2Vec + PCA	K-Means++	0,20826976	1,7632678767	11241,838940
	K-Medoids	0,17724766	1,81158085065	10383,2209420
	Gaussian Mixture	0,086755656	5,4655767150	4496,7888117
	Bulanık C-Means	**0,23483634**	**1,51135170141**	**14319,8437998**
Word2Vec + PCA	K-Means++	0,15336287	1,99709464521	1465,84705122
	K-Medoids	0,014416903	6,72022705715	133,78200406
	Gaussian Mixture	-0,03423659	2,9743953653	170,505561746
	Bulanık C-Means	**0,2116015**	**2,03369259478**	**1485,2497058**
fastText + PCA	K-Means++	0,13262811	2,06975007887	1138,79134550
	K-Medoids	0,013456795	7,02655414565	114,241904793
	Gaussian Mixture	0,0895558	8,46556365838	384,595955000
	Bulanık C-Means	**0,1941493**	**1,77026133979**	**1751,90188139**

Tablo 6.11. Javascript veri setine ait PCA uygulanmış kümeleme skorları

Özellik Gösterimi	Kümeleme Algortiması	Silhouette Skoru	DB Skoru	CH Skoru
Doc2Vec + PCA	K-Means++	0,19613242	1,81040251508	17717,4432159
	K-Medoids	0,16203382	1,8628830018	15582,9004673
	Gaussian Mixture	0,072113246	7,9980463499	950,306844350
	Bulanık C-Means	**0,23009749**	**1,5597675802**	**23362,7150831**
Word2Vec + PCA	K-Means++	0,12745672	2,56728601075	736,68123422
	K-Medoids	0,005006832	9,838695144	50,3464930895
	Gaussian Mixture	0,0751768	3,45015027706	405,41177867
	Bulanık C-Means	**0,12970085**	**2,3080778986**	**891,173435677**
fastText + PCA	K-Means++	0,15297924	2,3638393551	983,714610650
	K-Medoids	0,009198457	7,4396370869	81,269060978
	Gaussian Mixture	0,022325028	3,29696973190	452,42178805
	Bulanık C-Means	**0,17278989**	**2,0952332643**	**1278,405028**

Tüm veri setlerinde ve kümeleme algoritmalarında 2 farklı küme oluşmuştur. Küme sayısının belirlenmesi için roc eğrisinden faydalanılmıştır. Şekil 6.9'da da görülebileceği gibi roc eğrisinde en fazla keskin geçişin olduğu kısım küme sayısını göstermektedir. Oluşan kümeler incelendiğinde, 1. kümede kullanıcının karşılaştığı hata ve problem tipindeki sorular varken, 2. kümede teorik olarak veya bir konu

hakkında bilgi almak için sorulmuş sorular yer almaktadır. 2 farklı örnek doküman üzerinde, dokümanlara en benzer olan dokümanların Kosinüs ve Pearson Korelasyon benzerlik skorları Tablo 6.12 ve Tablo 6.14'de gösterilmiştir. Benzerlik skorları incelendiğinde Kosinüs benzerliğinin Pearson Korelasyon benzerliğine göre birbirine benzer dokümanları bulmada daha başarılı olduğu görülmüştür.



Şekil 6.9. Roc eğrisi

Tablo 6.12. Python veri setindeki 1. örnek dokümana benzer dokümanların benzerlik skorları

Doküman	Kosinüs Benzerliği	Pearson Korelasyon Benzerliği
1	0,7599612474441528	0,7593756914138794
2	0,7233498692512512	0,7225711345672607
3	0,7099725008010864	0,7090925574302673
4	0.7077314853668213	0.7068307995796204
5	0.7076513767242432	0.7070661783218384
6	0.7075223922729492	0.7070504426956177
7	0.7075188159942627	0.7066984176635742
8	0.7075053453445435	0.7065587639808655
9	0.707503616809845	0.7064993977546692
10	0.7073231935501099	0.7064351439476013

Birinci örnek doküman ve bu dokümana benzer olan dokümanların soru metinleri Tablo 6.13'de gösterilmiştir.

Tablo 6.13. Python veri setindeki 1. örnek dokümana benzer dokümanlar

Örnek Doküman	Python reading csv file by read_csv Hello data science community, how can i read a csv file without using separators? pd.read_csv(filename) but without using separator I tried but it gives me error unicode
Dokümanın Vektördeki Temsili	python read csc file read hello data science community read csc file without use spear pm read csc filename without use spear try give error unicode

Benzer Doküman 1	How to refer to parent class' class variable without an instance
	<pre><code>
	class A(): a = [1, 2] class B(A): a = super().a + [3]</code></pre>
	 Gives the error:
	<pre><code> RuntimeError: super(): no arguments</code></pre>
	 I want the result to be
	<pre><code> class B(A): = [1, 2, 3]</code></pre>

Dokümanın Vektördeki Temsili	refer parent class class variable without instant give error want result

Benzer Doküman 2	Title: AttributeError: type object 'DatasetV2' has no attribute 'from_tensor' I am learning Tensorflow 2.0 and I am getting an error as I run the following code:
	<pre><code>data = np.random.randint(0,10, (3,4)) dataset1 = tf.data.Dataset.from_tensor(data)
	---------- AttributeError Traceback (most recent call last) <ipython-input-38-9873d7e2b8bc> in <module>() ----> 1 dataset1 = tf.data.Dataset.from_tensor(data)
	AttributeError: type object 'DatasetV2' has no attribute 'from_tensor'
	</code></pre>
	 The same here:
	<pre><code>e= tf.data.Dataset.from_element(10)
	AttributeError: type object 'DatasetV2' has no attribute 'from_element'
	</code></pre>
	 But I can run the code:
	<pre><code>dataset = tf.data.Dataset.range(10)</code></pre>
	 without a problem.

Dokümanın Vektördeki Temsili	attribute error type object attribute tensor learn tensor low get error run follow code run code without problem

Tablo 6.13. (Devam) Python veri setindeki 1. örnek dokümana benzer dokümanlar

Benzer Doküman 3	Python: Writing into file fails without error - getting an empty file This code seems to be working fine and the console is giving me exactly what I need without any errors. However, when I try to open it up with excel it gives me an empty file. <pre><code>import csv import urllib.request from bs4 import BeautifulSoup f = open('aapl_analyst_estimates', 'w', newline= '') writer = csv.writer(f) soup = BeautifulSoup(urllib.request.urlopen ('https://www.marketwatch.com/investing/stock/aapl/ analystestimates').read(), 'lxml') tbody = soup('table',{'class':'snapshot'})[0].find_all('tr') for row in tbody: cols = row.findChildren(recursive=False) cols = [ele.text.strip() for ele in cols] writer.writerow(cols) print(cols) f.close() </code></pre>
Dokümanın Vektördeki Temsili	python write file fail without error get empty file code seem work fine consol give exact need without error howe try open excel give empty file

Tablo 6.14. Python veri setindeki 2. örnek dokümana benzer dokümanların benzerlik skorları

Doküman	Kosinüs Benzerliği	Pearson Korelasyon Benzerliği
1	0.7252475023269653	0.7199762463569641
2	0.6597511172294617	0.6551288962364197
3	0.6575804948806763	0.6511399745941162
4	0.6549136638641357	0.651769757270813
5	0.6527427434921265	0.6465413570404053
6	0.6506528258323669	0.6497166156768799
7	0.6481313109397888	0.6414132118225098
8	0.6416511535644531	0.6331203579902649
9	0.6351412534713745	0.6281378865242004
10	0.6340853571891785	0.6237061023712158

İkinci örnek doküman ve bu dokümana benzer olan dokümanların soru metinleri Tablo 6.15'de gösterilmiştir.

Tablo 6.15. Python veri setindeki 2. örnek dokümana benzer dokümanlar

Örnek Doküman	How to Parallelize a GridSearch Scan with Talos While <code>talos</code> supports GPU parallelization, how do you extend the Scan object to support CPU + GPU parallelization?
Dokümanın Vektördeki Temsili	parallel grid search scan talk support gnu parallel nation extend scan object support cup gnu parallel
Benzer Doküman 1	Parallelism in pairgrid plots For example, if I had a pairgrid plot like so: <pre class="lang-py prettyprint-override"><code>import matplotlib as plt import seaborn as sns p = sns.PairGrid(<some data>) p.map_upper(plt.scatter) p.map_lower(sns.kdeplot) p.map_diag(plt.hist) </code></pre> Each of these <code>map_*</code> ops can run a while depending on the size and dimensionality of data, is there some way these can be parallelized?
Dokümanın Vektördeki Temsili	parallel pair grid plot example pair grid plot like run depend size dimension data way parallel

Benzer Doküman 2	drawing grid lines beween points in python
	 I have 6 points in the <code>(x,y)</code> plane: <code> x=[x1,x2,x3,x4,x5,x6]</code> and
	<code>
	y=[y1,y2,y3,y4,y5,y6] </code>
	<pre><code>import matplotlib.pyplot as plt
	x = [0, 2, 4, 0, 2, 4, 0, 2, 4]
	y = [0, 0, 0, 3, 3, 3, 7, 7, 7]
	plt.scatter(x, y)
	plt.show()
	</code></pre>
	 I want to between the points, draw entirely parallel lines on each axis <code>x,y</code>(like photo). and how to hide x and y axis on diagram. I want to draw a 2D view of the beams and columns of 3 story building; does <code>matplotlib</code> bring me to my goal or should I go to other libraries?

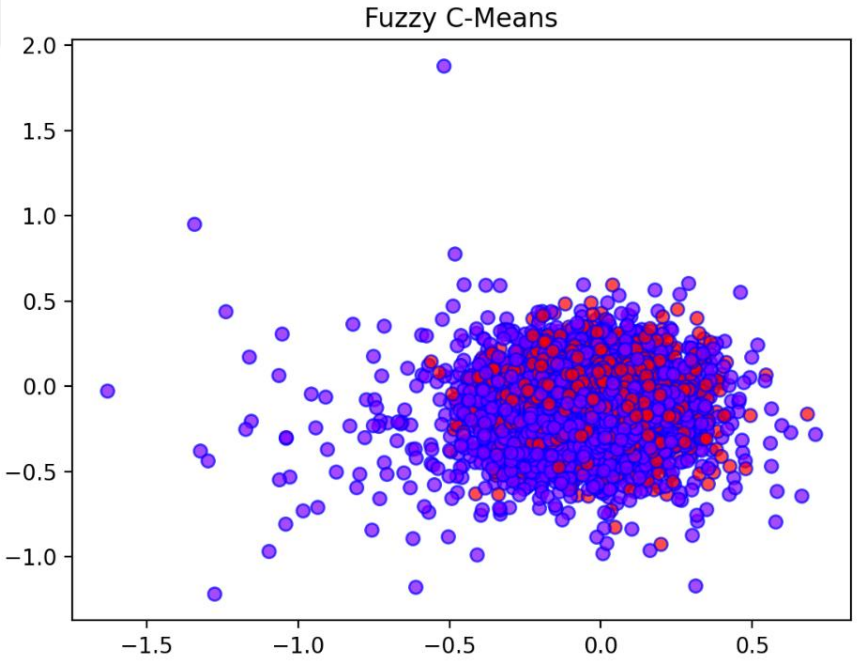
Dokümanın Vektördeki Temsili	draw grid line point python point want point draw enter parallel line axis hide axis diagram want draw view beam column store build bring goal go library

Tablo 6.15. (Devam) Python veri setindeki 2. örnek dokümana benzer dokümanlar

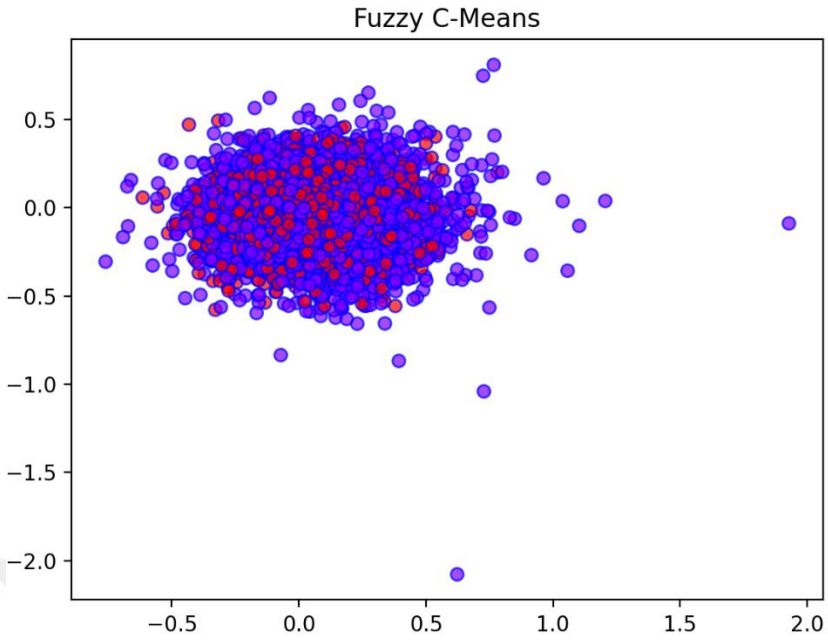
Benzer Doküman 3	Networkx- finding the parallel edges of multiDigraph
	 I have a multidigraph like this-
	<pre>
	<code>
	Import Networkx as nx
	G=nx.MultiDiGraph()
	G.add_edge(0,1)
	G.add_edge(1,0)
	G.add_edge(1,3)
	</code>
	</pre>
	 Any networkx way to find edge 0-1 and 1-0 are parallel?

Dokümanın Vektördeki Temsili	network find parallel edge multi digraph multi digraph like network way find edge parallel

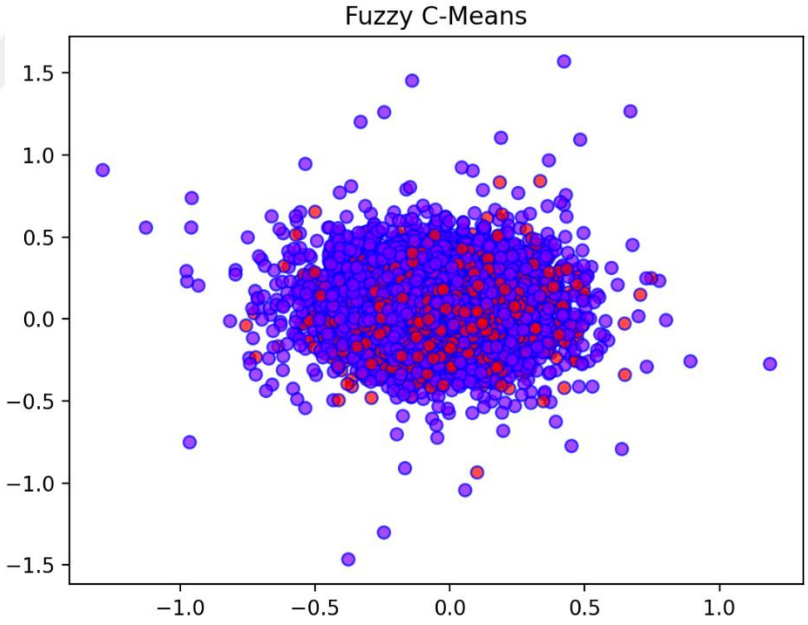
Java, Javascript ve Python veri setlerine ait benzer dokümanların Doc2Vec ve bulanık c-means algoritmasıyla kümelenmesi sonucu oluşan küme gösterimleri Şekil 6.10 Şekil 6.11, Şekil 6.12'de gösterilmiştir.



Şekil 6.10. Java veri setindeki Doc2Vec ve Bulanık C-Means küme gösterimi

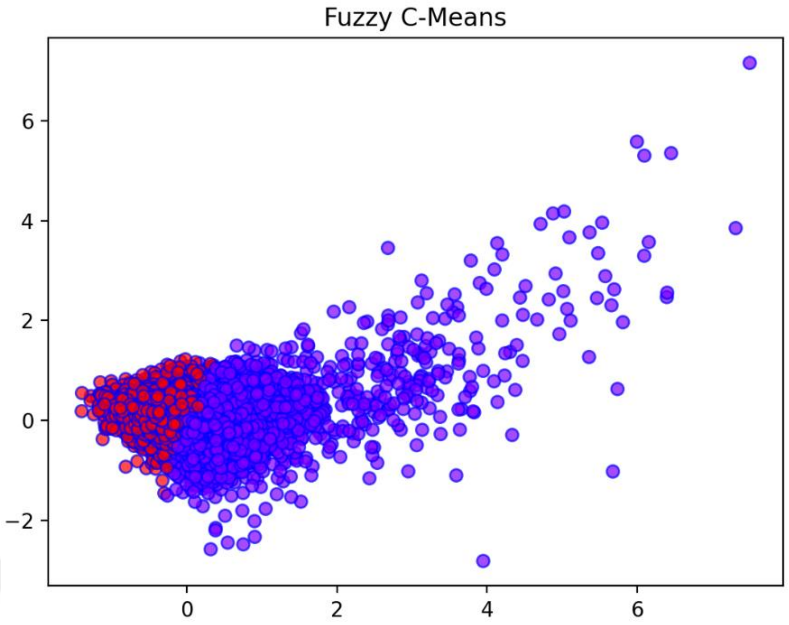


Şekil 6.11. Javascript veri setindeki Doc2Vec ve Bulanık C-Means küme gösterimi

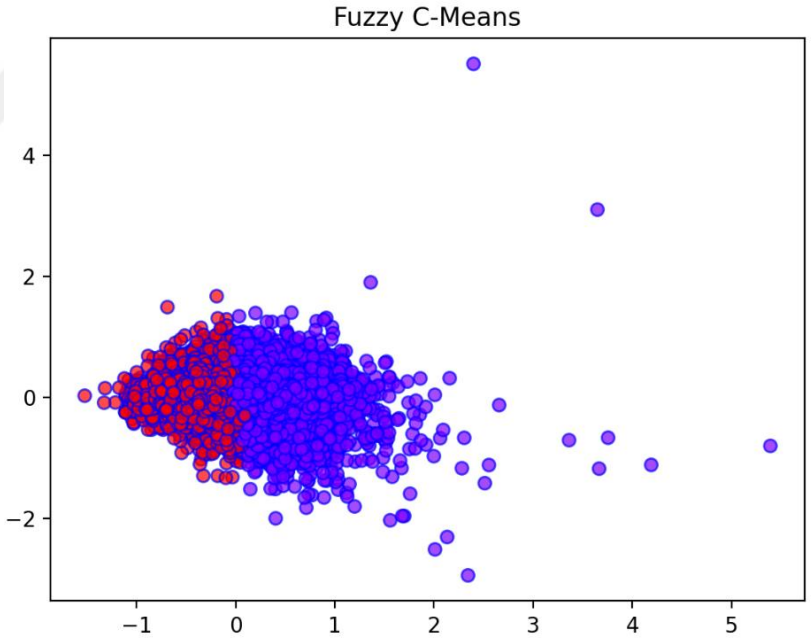


Şekil 6.12. Python veri setindeki Doc2Vec ve Bulanık C-Means küme gösterimi

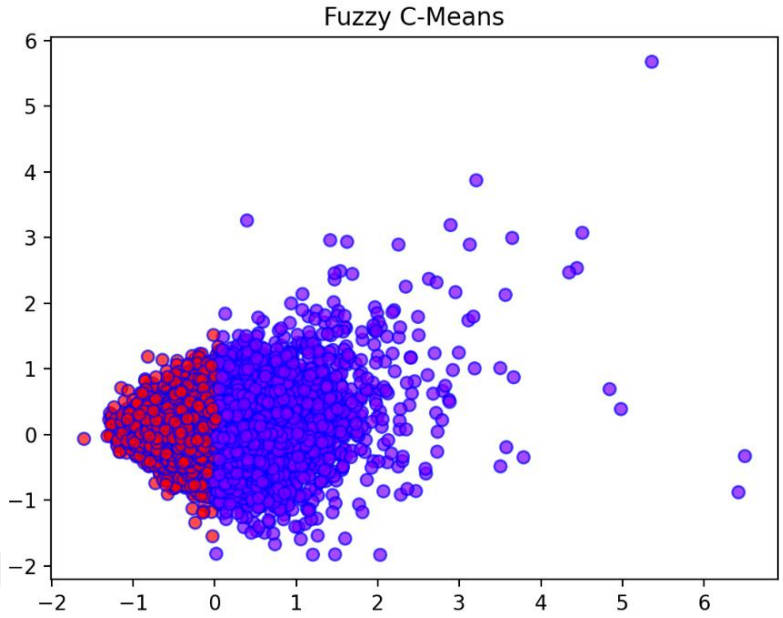
Java, Javascript ve Python veri setlerine ait benzer dokümanların Doc2Vec ve PCA kullanılarak en iyi sonuç veren bulanık c-means algoritmasıyla kümelenmesi sonucu oluşan küme gösterimleri Şekil 6.13 Şekil 6.14, Şekil 6.15'de gösterilmiştir.



Şekil 6.13. Java veri setindeki Doc2Vec ve PCA yöntemi ile Bulanık C-Means küme gösterimi



Şekil 6.14. Javascript veri setindeki Doc2Vec ve PCA yöntemi ile Bulanık C-Means küme gösterimi



Şekil 6.15. Python veri setindeki Doc2Vec ve PCA yöntemi ile Bulanık C-Means küme gösterimi

Tez kapsamında yapılan tüm deneylerden elde edilen sonucu özetleyecek olursak, Doc2Vec, temel bileşenler analizi ve bulanık c-means algoritması birlikte kullanıldığında benzer dokümanların kümelenmesinde daha başarılı bir sonuç elde edilmiştir. Bunun nedeni Doc2Vec modelinin bağlamsal benzerliğe dayandırılmış olmasıdır. Diğer modellerde, benzerlik sadece kelimelere dayanmaktadır. Yani küme içindeki noktaların birbirine olan benzerlikleri diğer kümelerdeki noktalara göre daha yüksek ve kümelerin birbirleriyle olan uzaklıkları daha fazladır.

7. SONUÇLAR VE ÖNERİLER

Günümüzde sosyal medya, forumlar, haber siteleri veya çevrimiçi ortamlarda veri miktarında yaşanan büyük artışla birlikte bilgiye erişim zorlaşmıştır. Bu veriler üzerinden istediğimiz bilgiye ulaşmak için otomatik yöntemlere ihtiyaç duyulmaktadır.

Çalışma kapsamında bilgisayar programcılığında popüler olarak kullanılan ve bir soru cevap sitesi olan stackoverflow üzerinden gönderilen sorular kullanılmıştır. Bu veri setinden yola çıkarak otomatik kümeleme yöntemi yaklaşımı önerilmiştir. Veri seti üzerinde veri ön işleme adımları uygulanarak veri içerisinde yer alan tutarsız veriler daha anlamlı hale getirilerek modelin başarısı yükseltilmiştir. Aynı anahtar kelimeye sahip veriler gruplanmış ve en fazla soru sorulan 3 farklı veri seti üzerinde çalışma yapılmıştır.

Soru dokümanları üzerinde doc2vec, word2vec ve fastText kelime yerleştirme yöntemleri kullanılarak dokümanların anlamsal benzerliklerine dayalı olarak vektör haline getirilmiştir. Vektörler üzerinde bir öznitelik boyut indirgeme yöntemi olan Temel Bileşenler Analizi yöntemi uygulandığında başarı artmıştır. Gaussian Mixture kümeleme algoritmasında doc2vec ve word2vec yöntemleriyle beraber PCA uygulandığında modelin başarısı düşmüş olup fastText algoritmasında modelin başarısı yükselmiştir. Diğer kümeleme algoritmalarında ise modelin başarısı yükselmiştir. Öznitelik boyut indirgeme arama alanını daraltacağından, modelin başarısını artırmak ve bu işlemi doğru şekilde yaparak daha anlamlı bilgiler elde etmek amaçlanmıştır. Yine kelime temsili yöntemlerinde kullanılan epoch sayısının başarıyı etkilediği tespit edilmiştir. Doc2vec ve fastText temsil yöntemlerinde epoch sayısı daha düşük değerlerde daha yüksek başarı gösterirken word2vec yönteminde epoch sayısı daha yüksek değerlerde yüksek başarı göstermiştir.

Dokümanlar k-means++, k-medoids, gaussian mixture ve bulanık c-means kümeleme algoritmalarıyla kümelenmiştir ve 3 farklı değerlendirme metriğiyle sonuçlar kıyaslanmıştır. Doc2vec ve bulanık c-means algoritmaları birlikte kullanıldığında en yüksek skor elde edilmiştir. Doc2vec algoritmasının diğer algoritmalara göre daha yüksek sonuç vermesinin nedeni tüm kelime vektörlerinin toplanması ve ortalamasının alınmasıyla dokümanlar anlamsal olarak korunmakta ve dokümanın tamamına semantik olarak bakılmasıdır. Word2vec kelime bazlı olarak tahminleme yaptığı için ve doc2vec dokümanın tamamına bakarak bir tahminleme yaptığı için doc2vec algoritmasıyla belgeler arasındaki ilişki daha düzgün olarak ölçülebilmiştir.

Önerilen model 3 farklı veri setleri üzerinde uygulanmış olup elde edilen sonuçlar tüm veri setleri için hemen hemen aynı başarıyı göstermiştir. Veriler üzerinde PCA uygulanmadığında 3 farklı değerlendirme metriğine göre en yüksek model başarısı doc2vec ve k-means++ algoritmalarının beraber kullanılmasıyla elde edilmiş olup, PCA uygulandığında doc2vec ve bulanık c-means algoritmalarının kullanılmasıyla elde edilmiştir.

Kümeleme işlemi sonucunda 3 veri setinde de 2 farklı küme oluşmuştur. Oluşan kümelerden ilkinde hata ve problem tipindeki sorular varken, ikinci kümede teorik olarak veya bir konu hakkında bilgi almak için sorulmuş sorular yer almaktadır. Dokümanların 2 farklı gruba ayrılması arama alanını daraltacağı için benzer dokümanları bulmada kolaylık sağlamıştır. Küme içindeki birbirlerine benzer dokümanları bulmak için kosinüs ve pearson korelasyon benzerlik ölçüleri kullanılmış olup, bu ölçüler birbirleriyle karşılaştırılmıştır. Kosinüs benzerlik ölçüsü, Pearson Korelasyon benzerlik ölçüsüne benzer dokümanları bulmada daha yüksek sonuç vermiştir.

Dokümanların kümelenmesinde Doc2Vec, temel bileşenler analizi ve bulanık c-means yöntemlerinin birlikte kullanılmasıyla başarılı bir sonuç elde edildiği ve yapısal olmayan metin dokümanlarının otomatik olarak kümelendirilmesinde çözüm olacağı bu çalışma ile beraber belirlenmiştir. Ayrıca küme içindeki benzer dokümanları bulmada Kosinüs benzerliğinin daha iyi olduğu belirlenmiştir.

Bu çalışmanın devamı olarak ileride autoencoder, t-SNE gibi farklı öz nitelik boyut indirgeme yöntemlerine odaklanıp bu yöntemleri bağımsız veya harmanlayarak kullanmak suretiyle yeni yaklaşımların oluşturulması planlanmaktadır.

KAYNAKLAR

[1] Tunalı V., Metin Madenciliği için İyileştirilmiş Bir Kümeleme Yapısının Tasarımı ve Uygulaması, Doktora Tezi, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2011, 304604.

[2] Sherkat E., Nourashrafeddin S., Milios E.E., Minghim, R., Interactive document clustering revisited: A visual analytics approach, *23rd International Conference on Intelligent User Interfaces*, Tokyo, Japan, 7-11 Mart 2018.

[3] Klinczak M., Kaestner C., Comparison of clustering algorithms for the identification of topics on twitter, *Latin American Journal of Computing Faculty of Systems Engineering Escuela Politécnica Nacional Quito-Ecuador*, 2016, **3**(1), 19–26.

[4] Recupero D. R., A New Unsupervised Method for Document Clustering by Using Wordnet Lexical and Conceptual Relations, *Information Retrieval*, 2007, **10**(6), 563-579.

[5] Özdoğan A.G., Metin Madenciliği Kullanarak İngilizce Doküman Sınıflama, Yüksek Lisans Tezi, İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2019, 600821.

[6] Han J., Pei J., Kamber M., *Data Mining: Concepts and Techniques*, Elsevier, 2011.

[7] Işık M., Çamurcu A.Y., K-means ve Aşırı Küresel C-means Algoritmaları ile Belge Madenciliği, *Marmara Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 2010, **22**(2010), 1-18.

[8] Huang G.Y., Liang D.P., Hu C.Z., Ren, J.D., An Algorithm for Clustering Heterogeneous Data Streams with Uncertainty, *2010 International Conference on Machine Learning and Cybernetics*, Qingdao, China, 11-14 Temmuz 2010.

[9] Lee S.J., Zeng X.J., A three-part input-output Clustering-Based Approach to Fuzzy System Identification, *2010 10th International Conference on Intelligent Systems Design and Applications*, Cairo, Egypt, 29 Kasım - 1 Aralık 2010.

[10] Meng L., Tan A.H., Xu D., Semi-Supervised Heterogeneous Fusion for Multimedia Data Co-Clustering, *IEEE Transactions on Knowledge and Data Engineering*, 2013, **26**(9), 2293–2306.

[11] Meng L., Tan A.H.,Wunsch D.C., Adaptive Scaling of Cluster Boundaries for Large-Scale Social Media Data Clustering, *IEEE Transactions on Neural Networks and Learning Systems*, 2015, **27**(12), 2656–2669.

[12] Kalogeratos A., Likas A., Document Clustering Using Synthetic Cluster Prototypes, *Data & Knowledge Engineering*, 2011, **70**(3), 284–306.

[13] Mendes M. E. S., Sacks, L., Evaluating Fuzzy Clustering for Relevance-Based Information Access, *12th IEEE International Conference on Fuzzy Systems, St Louis*, USA, 25-28 Mayıs 2003.

[14] Dursun B., Sönmez A.C., Türkçe Metin Benzerlik Hesaplaması için Yeni Bir Yöntem, *2008 IEEE 16th Signal Processing, Communication and Applications Conference*, Aydın, Türkiye, 20-22 Nisan 2008.

[15] Afzali M., Kumar S., Text Document Clustering: Issues and Challenges, *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Faridabad, India, 14-16 Şubat 2019.

[16] Li Y., Cai J., Wang, J., A Text Document Clustering Method Based on Weighted BERT Model, *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chongqing, China, 12-14 Haziran 2020.

[17] Mijangos V., Sierra G., Montes A., Sentence Level Matrix Representation for Document Spectral Clustering, *Pattern Recognition Letters*, 2017, **85**, 29-34.

[18] Krizhevsky A., Sutskever I., Hinton G.E., ImageNet Classification with Deep Convolutional Neural Networks, *Advances in Neural Information Processing Systems*, 2012, 1097-1105.

[19] Yang Z., Yang D., Dyer C., He X., Smola A., Hovy E., Hierarchical Attention Networks for Document Classification, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, California, USA, 12-17 Haziran 2016.

[20] Young T., Hazarika D., Poria D., Cambria E., Recent Trends in Deep Learning Based Natural Language Processing, *IEEE Computational IntelligenCe Magazine*, 2018, **13**(3), 55-75.

[21] Karasoy O., Ballı S., Classification Turkish SMS with Deep Learning Tool Word2Vec, *2nd International Conference on Computer Science and Engineering (UBMK)*, Antalya, Türkiye, 5-8 Ekim 2017.

[22] Mikolov T., Chen K., Corrado G., Dean. J., Efficient Estimation of Word Representations in Vector Space, *arXiv preprint arXiv*, 2013, 1301.3781.

[23] Le Q., Mikolov T., Distributed Representations of Sentences and Documents, *Proceedings of the 31 st International Conference on Machine Learning*, Beijing, China, 21-26 Haziran 2014.

[24] Eddamiri S., Zemmouri E., Benghabrit A., Graph embeddings for linked data clustering, *Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services*, Yogyakarta, Indonesia, 19-21 Kasım 2018.

[25] Steinbach M., Ertöz L., Kumar V., The challenges of clustering high dimensional data, *New directions in statistical physics*, 2004, 273–309.

[26] Hofmann T., Schölkopf B., Smola A.J., Kernel methods in machine learning, *The Annals of Statistics*, 2008, **36**(3), 1171–1220.

[27] De la Torre F., Kanade, T., Discriminative cluster analysis, *Proceedings of the 23rd international conference on machine learning*, Pittsburgh, USA, 2006.

[28] Wikipedia contributors, Stack Overflow, Wikipedia, https://en.wikipedia.org/wiki/Stack_Overflow#cite_note-20 (Ziyaret Tarihi : 30 Haziran 2020)

[29] Sewak, M., Finding a Growth Business Model at Stack Overflow, Inc, *Stanford CasePublisher*, Stanford University School of Engineering, 2010.

[30] Kadhim A. I., Cheah Y., Ahamed N. H., Text Document Preprocessing and Dimension Reduction Techniques for Text Document Clustering, *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*, Kota Kinabalu, Malaysia, 3-5 Aralık 2014.

[31] Vijayarani S., Ilamathi M. J., Nithya M., Preprocessing techniques for text mining-an overview, *International Journal of Computer Science & Communication Networks*, 2015, **5**(1), 7-16.

[32] Tunalı V., Bilgin, T. T., PRETO: A high-performance Text Mining Tool for Preprocessing Turkish Texts, *Proceedings of the 13th International Conference on Computer Systems and Technologies*, Ruse, Bulgaria, 22-23 Haziran 2012.

[33] Sandhya M., Sarika, S., Anukriti, S., Sushila, A., Automatic Text Categorization on News Articles, *International Journal of Engineering and Techniques*, Pune, India, Mayıs-Haziran 2016.

[34] Yılmaz T., Improving Educational Search and Question Answering, Yüksek Lisans Tezi, İhsan Doğramacı Bilkent Üniversitesi, Mühendislik ve Fen Bilimleri Enstitüsü, Ankara, 2016, 434215.

[35] Schnabel T., Labutov I., Mimno D., Joachims T., Evaluation methods for unsupervised word embeddings, *Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 17-21 Eylül 2015.

[36] Sen M.U., Erdogan H., Learning word representations for Turkish, *22nd Signal Processing and Communications Applications Conference (SIU)*, Trabzon, Türkiye, 23-15 Nisan 2014.

[37] Gözükara F., Özel S. A., Türkçe ve İngilizce Yorumların Duygu Analizinde Doküman Vektörü Hesaplama Yöntemleri için Bir Deneysel İnceleme, *Çukurova Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*, 2016, **31**(2), 464-482.

[38] Şahin G., Turkish document classification based on Word2Vec and SVM classifier, *2017 25th Signal Processing and Communications Applications Conference (SIU)*, Antalya, Türkiye, 15-18 Mayıs 2017.

[39] Mikolov T., Sutskever I., Chen K., Corrado G., Dean J., Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems*, 2013, 3111–3119.

[40] Wikipedia contributors, fastText, Wikipedia, https://en.wikipedia.org/wiki/FastText, (Ziyaret Tarihi : 30 Haziran 2020)

[41] Gülaç M.E., Identifying Image Related Sentences in News Articles, Yüksek Lisans Tezi, Boğaziçi Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2019, 602669.

[42] Zhang L., Li, J., Wang C., Automatic Synonym Extraction Using Word2Vec and Spectral Clustering, *2017 36th Chinese Control Conference (CCC)*, Dalian, China, 26-18 Temmuz 2017.

[43] Bilgin, M., Şentürk, İ. F., Sentiment Analysis on Twitter Data with Semi-Supervised Doc2Vec, *2017 International Conference on Computer Science and Engineering (UBMK)*, Antalya, Türkiye, 5-8 Ekim 2017.

[44] Van Dijk B., Towards text analytical information enrichment in the analysis of crime, Master Thesis, Eindhoven University of Technology, Utrecht, 2017.

[45] Bojanowski P., Grave E., Joulin A., Mikolov T., Enriching Word Vectors with Subword Information, *Transactions of the Association for Computational Linguistics*, 2017, **5**, 135-146.

[46] Feldman R., Sanger J., *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*, Cambridge University Press, 2007.

[47] Yıldız K., Çamurcu Y., Doğan B., Veri madenciliğinde temel bileşenler analizi ve Negatifsiz matris çarpanlarına ayırma tekniklerinin karşılaştırmalı analizi, *Akademik Bilişim*, 2010, 10-12.

[48] Berkhin P., Survey of Clustering Data Mining Techniques, In Grouping multidimensional data, Springer, Berlin, 2006.

[49] Gorur K., Bozkurt M.R., Bascil M.S., Temurtas F., Glossokinetic Potential Based Tongue–Machine Interface for 1-D Extraction Using Neural Networks, *Biocybernetics and Biomedical Engineering*, 2018, **38**(3), 745-759.

[50] Cılasun M.H., Yalçın H., A Deep Learning Approach to EEG based Epilepsy Seizure Determination, *24th Signal Processing and Communication Application Conference (SIU)*, Zonguldak, Türkiye, 16-19 Mayıs 2016.

[51] Basçıl M.S., Beyinde Üretilen Yöne Bağlı EEG Sinyallerinin Öznitelik Çıkarımı Yardımıyla Sınıflandırılması, Doktora Tezi, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Sakarya, 2015, 397355.

[52] Rokach L., Maimon O., Clustering Methods, *Boston, MA: Springer*, 2005.

[53] Kongwudhikunakorn S., Waiyamai K., Combining Distributed Word Representation and Document Distance for Short Text Document Clustering, *Journal of Information Processing Systems*, 2020, **16**(2), 277-300.

[54] Sariman G., Veri madenciliğinde kümeleme teknikleri üzerine bir çalışma: k-means ve k-medoids kümeleme algoritmalarının karşılaştırılması, *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 2011, **15**(3), 192-202.

[55] Day W.H., Edelsbrunner H., Efficient algorithms for agglomerative Gaussian Mixture Model clustering methods, *Journal of classification*, 1984, **1**(1), 7–24.

[56] Virupakshappa K., Oruklu E., Unsupervised Machine Learning for Ultrasonic Flaw Detection using Gaussian Mixture Modeling, K-Means Clustering and Mean Shift Clustering, *2019 IEEE International Ultrasonics Symposium (IUS)*, Glasgow, United Kingdom, 6-9 Ekim 2019.

[57] McLachlan G.J., Peel D., *Finite Mixture Models*, John Wiley & Sons, 2004.

[58] Dempster A., Laird N.M., Rubin D.B., Maximum Likelihood from Incomplete Data Via the EM Algorithm, *Journal of the Royal Statistical Society:Series B*, 1977, **39**(1), 1–38.

[59] Höppner F., Klawonn F., Kruse R., Runkler T., *Fuzzy cluster analysis: methods for classification, data analysis and image recognition*, John Wiley & Sons 1999.

[60] Moertini V. S., Introduction to Five Clustering Algorithms, *Integral*, 2002, **7**(2), 87-96.

[61] Duda R. O., Hart P. E., Stork D. G., *Pattern Classification*, John Wiley & Sons, 2012.

[62] Wikipedia contributors, Similarity measure, Wikipedia, https://en.wikipedia.org/wiki/Similarity_measure, (Ziyaret Tarihi: 7 Temmuz 2020).

[63] Amasyalı M. F., Ersoy O., The Performance Factors of Clustering Ensembles, *2018 IEEE 16th Signal Processing, Communication and Applications Conference*, Aydın, Türkiye, 20-22 Nisan 2008.

[64] Arthur D., Vassilvitskii S., K-means++: The advantages of careful seeding, *Stanford*, 2006, 1027-1035.

[65] Zhang Y., Chen B., Xu Y., Wang L., Xu H., Ma H., Mechanical Fault Diagnosis Method for OLTC Based on MPE and K-means++ Algorithm, *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, China, 12-14 Ekim 2018.

[66] Akgün A.S., Arama Sorguları Üzerinde Görev Tabanlı Kümeleme, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2018, 507150.

[67] Qiao S., Geng X., Wu, M., An improved method for K_medoids algorithm, *In 2011 International Conference on Business Computing and Global Informatization*, Shanghai, China, 29-31 Temmuz 2011.

[68] Gordon, A.D., Cluster validation, Hayashi C., Yajima K., Bock H.H., Ohsumi N., Tanaka Y., Baba Y., *Studies in Classification, Data Analysis, and Knowledge Organization: Data Science, Classification, and Related Methods*, Springer, Tokyo, 22–39, 1998.

[69] Hämäläinen J., Jauhiainen S., Kärkkäinen T., Comparison of Internal Clustering Validation Indices for Prototype-Based Clustering, *Algorithms*, 2017, **10**(3), 105.

[70] Bolshakova N., Azuaje F., Cluster Validation Techniques for Genome Expression Data, *Signal Processing*, 2003, **83**(4), 825-833.

[71] Dimitriadou E., Dolnicar S., Weingessel A., An examination of indexes for determining the Number of Cluster in binary data sets, *Psychometrika*, 2002, **67**(1), 137-160.

[72] Kaufman L., Rousseeuw P.J., *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 2009.

[73] Dudoit S., Fridlyand J., A prediction-based resampling method for estimating the number of clusters in a dataset, *Genome Biology*, 2002, **3**(7), 0036.1- 21.

[74] Wikipedia contributors, Davies-Bouldin Index, Wikipedia, https://en.wikipedia.org/wiki/Davies- Bouldin_index, (Ziyaret Tarihi: 25 Mayıs 2020).

[75] Liu Y., Li Z., Xiong H., Gao X., Wu J., Understanding of internal clustering validation measures. *2010 IEEE International Conference on Data Mining*, Sydney, Australia, 13-17 Aralık 2010.

[76] Wikipedia contributors, Silhouette (clustering), Wikipedia, https://en.wikipedia.org/wiki/Silhouette_(clustering), (Ziyaret Tarihi: 25 Mayıs 2020).

[77] Calinski T., Harabasz J., A Dendrite Method for Cluster Analysis, *Communications in Statistics – Theory and Methods*, 1974, **3**(1), 1–27.

[78] Ergüt Ö., Uzaklık ve Benzerlik Ölçülerinin Kümeleme Sonuçlarına Etkisi, Yüksek Lisans Tezi, Marmara Üniversitesi, Sosyal Bilimleri Enstitüsü, İstanbul, 2011, 291467.

[79] Curiskis S. A., Drake B., Osborn T.R., Kennedy P. J., An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit, *Information Processing & Management*, 2020, **57**(2), 1-21.

[80] Joulin A., Grave E., Bojanowski P., Mikolov T., Bag of Tricks for Efficient Text Classification, *arXiv preprint arXiv*, 2016, 1607.01759.

[81] Stein R. A., Jaques P. A., Valiati, J. F., An analysis of hierarchical text classification using word embeddings, *Information Sciences*, 2019, **471**, 216-232

[82] Ünal Z., Likert Tipi Verilerde Bulanık Mantık ve Derin Öğrenme Entegrasyonu, Doktora Tezi, Akdeniz Üniversitesi, Sosyal Bilimleri Enstitüsü, Antalya, 2019, 621670.

KİŞİSEL YAYIN VE ESERLER

Yelmen E., Duru N., Doc2vec Approach for Text Document Clustering, *International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES 2020),* Karabük, Türkiye, 03-05 Haziran 2020.

ÖZGEÇMİŞ

Eray YELMEN, 1994 yılında Çankırı'da doğdu. İlk, orta ve lise eğitimini Çankırı'da tamamladı. 2012 yılında girdiği Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2016 yılında mezun oldu. 2017 yılında başladığı, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine devam etmektedir.