

**HAVA H ARP OKULU
HAVACILIK VE UZAY TEKNOLOJİLERİ
ENSTİTÜSÜ**

**BÜYÜK VERİ ANALİZİ YÖNTEMLERİ VE YAZILIM TEKNOLOJİLERİYLE
METİN MADENCİLİĞİ**

YÜKSEK LİSANS TEZİ

Evren PALA

Bilgisayar Mühendisliği Ana Bilim Dalı Başkanlığı

Siber Güvenlik Program

Ağustos 2016

HAVA HARP OKULU
HAVACILIK VE UZAY TEKNOLOJİLERİ ENSTİTÜSÜ

**BÜYÜK VERİ ANALİZİ YÖNTEMLERİ VE YAZILIM TEKNOLOJİLERİYLE
METİN MADENCİLİĞİ**

YÜKSEK LİSANS TEZİ

Evren PALA
113107

Bilgisayar Mühendisliği Ana Bilim Dalı Başkanlığı

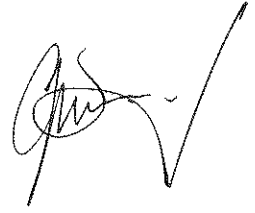
Siber Güvenlik Programı

Tez Danışmanı: Doç. Dr. Güray YILMAZ

Ağustos 2016

Hava Harp Okulu Havacılık ve Uzay Teknolojileri Enstitüsünün 113107 numaralı Yüksek Lisans Öğrencisi **Evren PALA**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra **Hv. Müh. Alb. Doç. Dr. Güray YILMAZ** danışmanlığında hazırladığı “**BÜYÜK VERİ ANALİZİ YÖNTEMLERİ VE YAZILIM TEKNOLOJİLERİ İLE METİN MADENCİLİĞİ**” başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Doç. Dr. Güray YILMAZ**
Hava Harp Okulu



Jüri Üyeleri : **Doç. Dr. İlker BEKMEZCİ**
Hava Harp Okulu



Yrd. Doç. Dr. Akhan AKBULUT
İstanbul Kültür Üniversitesi



Teslim Tarihi : 12 Ağustos 2016
Savunma Tarihi : 19 Ağustos 2016

Bu tez çalışmasında belirtilen görüş ve yorumlar yazara aittir. Türk Silahlı Kuvvetleri'nin ya da diğer kamu kuruluşlarının görüşlerini yansıtmaz. Ayrıca bu tez çalışması bilimsel ahlak ve etik değerlere uygun olarak yazılmış olup, yararlanılan tüm eserler kaynaklarda gösterilmiştir.

Ağustos 2016

Evren PALA

Aileme ve nişanlıma,

ÖNSÖZ

Bu tez çalışmasını yapmama olan katkılarından dolayı değerli hocam Doç. Dr. Güray Yılmaz'a, değerli meslektaşım Furkan Kamacı'ya, anne-babama ve nişanlıma teşekkür ederim.

Ağustos 2016

Evren PALA

İÇİNDEKİLER

Sayfa

KISALTMALAR	xiii
SEMBOL LİSTESİ	xv
ÇİZELGE LİSTESİ	xvii
ŞEKİL LİSTESİ	xix
ÖZET	xxi
1. GİRİŞ	1
1.1 Tezin Amacı	4
2. LİTERATÜR ARAŞTIRMASI	5
3. MALZEME VE YÖNTEM	9
3.1 Uygulamalarda Kullanılan Veri Kümeleri	9
3.1.1 Vikipedi Dokümanları	9
3.1.2 Twitter iletileri	10
3.1.3 20-Haber grubu veri seti (20-Newsgroups)	10
3.2 Uygulamalarda Kullanılan Dağıtık Veri İşleme ve Saklama Araçları	11
3.2.1 Apache Solr	11
3.2.2 Apache Hadoop	12
3.2.3 Apache Mahout	13
4. VİKİPEDİ DOKÜMANLARI REFERANS ALINARAK TWITTER İLETİLERİNİN ANALİZİ	15
4.1 Twitter Verisine Erişim	15
4.2 Vikipedi Dokümanlarının Solr ile İndekslenmesi	17
4.3 Vikipedi Dokümanları Baz Alınarak Tweet'lerin Yeninden Sıralanması	19
4.4 Elde Edilen Sonuçlar	22
5. 20-NEWSGROUPS DOKÜMANLARININ MERKEZİ ELEMANLAR REFERANS ALINARAK SPEKTRAL ALGORİTMA İLE ÖBEKLENMESİ	23
5.1 Spektral Öbekleme	23
5.2 Merkezi Elemanların Referans Olarak Kullanılması	25
5.3 Uygulamalar	27
5.4 Elde Edilen Sonuçlar	28
6. SONUÇ VE ÖNERİLER	31
KAYNAKLAR	33
ÖZGEÇMİŞ	37
TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR	37

KISALTMALAR

API	: Application Programming Interface
EM	: Expectation Maximization
ESA	: Explicit Semantic Analysis
HDFS	: Hadoop Distributed File System
LDA	: Linear Discriminant Analysis
LPI	: Locality Preserving Indexing
NMF	: Non-negative Matrix Factorization
PCA	: Principal Component Analysis
P2P	: peer-to-peer
TF-IDF	: Term Frequency – Inverse Document Frequency
VSM	: Vector Space Model

SEMBOL LİSTESİ

t	: <i>tweet</i> vektörü
u	: Twitter kullanıcısı vektörü
$sim(x,y)$: x ile y arasındaki benzerlik/yakınlık
$dist(x,y)$: x ile y arasındaki uzaklık
X	: TF-IDF vektörlerinden oluşan veri kümesi
x^i	: TF-IDF doküman vektörü
C	: Veri kümesi içindeki merkezi elemanlar (<i>medoidler</i>) kümesi
c^i	: TF-IDF medoid doküman
P	: Medoidlere indirgenmiş veri kümesi
p^i	: Medoidlere indirgenmiş doküman vektörü
A	: Yakınlık matrisi
k	: Veri kümesindeki öbek (cluster) sayısı
r	: TF-IDF vektörlerin boyut sayısı
m	: Veri kümesinde bulunan merkezi eleman (medoid) sayısı

ÇİZELGE LİSTESİ

Sayfa

Çizelge 3.1 : 20-NewsGroups veri setindeki kategoriler.....	11
Çizelge 4.1 : Twitter kullanıcılarının yeniden sıralamaya dair değerlendirmesi.....	22

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : Arama kayıtlarından otomatik tespit edilen kategoriler	2
Şekil 3.1 : Örnek bir Vikipedi sayfası.	9
Şekil 3.2 : Ters indeks mantığı.	11
Şekil 3.3 : Solr web arayüzünden "sosyal ağ" arama sonuçları	12
Şekil 3.4 : MapReduce çalışma mantığı.	13
Şekil 4.1 : Twitter'dan bir kullanıcıya ait tweetlerin çekilmesi (java).	16
Şekil 4.2 : Veritabanına kaydedilmiş <i>tweetler</i>	17
Şekil 4.3 : İndekslenecek dokümanlara uygulanan filtreler.....	18
Şekil 4.4 : Solr arama sunucusuna sorgu gönderilmesi	20
Şekil 5.1 : Spektral Öbekleme Algoritması	24
Şekil 5.2 : İki farklı veri dağılımı	25
Şekil 5.3 : Farklı m değerleri için öbekleme sonuçları.....	30
Şekil 5.4 : Algoritmaların 1 ve 3 düğümde çalışma zamanları	30

BÜYÜK VERİ ANALİZİ YÖNTEMLERİ VE YAZILIM TEKNOLOJİLERİYLE METİN MADENCİLİĞİ

ÖZET

Üretilen veri miktarının çok yüksek boyutlarda olduğu günümüzde, ham verilerin işlenerek bunlardan anlam çıkarılması oldukça önemli ve giderek yaygınlaşan bir işittir. Bu verilerin önemli bir kısmı, serbest halde yazılmış metinlerdir. Çeşitli ağlarda ve internet üzerinde üretilen metin içeriklerinin öğrenme algoritmaları kullanılarak otomatize bir şekilde tasnif edilmesinin önemli faydalarından biri de siber güvenlik kapsamında farkındalık kazandırmasıdır. Metin madenciliği olarak adlandırılan bu alanda yapılan geliştirme çalışmaları devam etmektedir.

Veri madenciliğinde, üzerinde çalışılan veri türünün kendine has özellikleri, veri setlerinin nasıl temizleneceği, matematiksel olarak nasıl ifade edileceği, ne tür algoritmalarla işleneceği, bu algoritmalarda kullanılacak yakınlık / uzaklık ölçülerinin nasıl belirleneceği gibi konular önemli rol oynamaktadır. Metin dokümanları, serbest halde üretilmiş, yazıldıkları dilin kuralları dışında bir şekle veya şablona tabi olmayan veri tipleridir. Dolayısıyla, veri madenciliği yöntemleri ile işlenmesi konusunda bir takım meseleleri öne çıkarmaktadır. Öncelikle, öğrenme algoritmalarının çalışması için veri örneklerinin yapısal bir halde, belirli sayıda bir özellik kümesi (feature set) ile boyutları sınırlandırılmış bir formda bulunması gerekmektedir. Metin dokümanlarının nümerik vektörlere dönüştürülmesinde her bir kelime ayrı birer özellik olduğu için yüksek boyut sayısı hem bellek kullanımı açısından, hem de veri işleme maliyeti açısından önemli bir problemdir. Dokümanların boyut sayısı belirli vektörel bir forma dönüştürülürken nasıl bir yöntem kullanılacağı önemli bir sorundur. İkinci olarak, metin veri kümelerinde, birbiriyle aynı kategorideki veri örneklerinden oluşan öbekler, genellikle diğerlerinden kolay ayırt edilebilir kompakt yapılar sergilememekte, iç içe geçmiş gruplar gözlemlenmektedir. Basit öbekleme (clustering) algoritmalarının, iyileştirici süreçler olmaksızın metin dokümanlarını yüksek isabette öbeklere ayırması oldukça zordur. Veri setlerinde iç içe geçmiş grupları birbirinden ayırmak için alternatif yaklaşımlar gerekmektedir.

Bu çalışmada, bahsedilen sorunlara bazı öneriler getirilerek metin verileri üzerinde kategorizasyon / öbekleme uygulamalarının isabet oranı iyileştirilmeye çalışılmıştır. Öncelikle, metin dokümanlarını vektörel forma getirme sürecinde metin içinde geçen kelimelerin doğrudan kullanılmasına alternatif bir yaklaşım olarak, bazı referans dokümanların kullanılması salık verilmiş ve uygulanmıştır. Bu yöntemle göre, dokümanlar vektöre dönüştürülürken vektörün her bir nümerik girdisi, söz konusu dokümanın, referans olarak kullanılan her bir dokümana olan yakınlık değeri ile ifade edilmektedir. İkinci olarak, iç içe geçmiş grupları birbirinden ayırmak

konusunda spektral öbikleme yöntemleri ele alınmıştır. Bu yöntem, veri üzerinde herhangi bir istatistiksel model varsayımı yapıp modele ait parametreleri optimize etmek yerine, veri örneklerinin birbiri ile nasıl bağlar kurduğunu temel alır. Spektral öbikleme algoritması, aynı kategorideki verilerden oluşan öbeklerin kompakt şekiller oluşturmadığı fakat bağlı bileşenler (connected components) formunda bulunduğu durumlarda oldukça başarılı bir algoritmadır. Bir diğer mesele olan uzaklık / yakınlık ölçüsü hususunda ise, metin verilerinde oldukça yaygın kullanılan kosinüs uzaklığı kullanılmıştır.

Bu tez kapsamında, metin madenciliğinde dokümanların vektörizasyonunda referans dokümanların kullanımı ve spektral öbikleme algoritmasının etkileri incelenmiş, öbikleme sonuçlarının isabet oranına olumlu katkıları olduğu gözlemlenmiştir. Çalışma, iki ayrı uygulamadan oluşmaktadır. İlk olarak, Twitter kullanıcıları tarafından yazılan türkçe tweetler, Vikipedi'deki Türkçe parçalar referans olarak kullanılarak ifade edilmiş ve bu yöntemle, kullanıcıların kendi Twitter hesaplarında gördükleri tweetler, kendileri ile daha alakalı olanlar üstte olacak şekilde yeniden sıralanmıştır. İkinci uygulamada ise İngilizce haber gruplarından oluşan 20-newsgroups veri seti, verinin içindeki merkezi elemanlar referans olarak kullanılarak vektörel forma dönüştürülmüş ve spektral öbikleme algoritması kullanılarak kategorize edilmiştir. Dokümanlar ayrıca referans dokümanlar kullanılmaksızın spektral öbikleme ve k-means algoritmaları ile öbeklenmiş ve sonuçlar karşılaştırılmıştır.

Uygulamalar, büyük ölçekli veri setleri üzerinde de hizmet verebilecek şekilde dağıtık veri işleme ve saklama araçları kullanılarak geliştirilmiştir. Vikipedi makaleleri Apache Solr üzerinde saklanarak tweetler ile arama sorguları gönderildiğinde skorlu bir şekilde sonuç seti dönmesi sağlanmıştır. Öte yandan, 20-newsgroups veri seti üzerinde yapılan çalışmalarda ise Apache Hadoop tarafından sağlanan dağıtık dosya sistemi kullanılmıştır. Öbikleme algoritmaları ise Apache Mahout kullanılarak dağıtık mantıkla çalıştırılmıştır.

TEXT MINING USING BIG DATA ANALYSIS METHODS AND TOOLS

SUMMARY

In today's world, making meaningful inferences from raw data is an important task and getting more widespread gradually as huge amount of data is being produced continuously. Text documents written in free format constitutes an important part of data being produced. One of the benefits of categorizing text documents using learning algorithms is creation of awareness. This is an important acquisition in terms of cyber security. There are ongoing studies to improve text mining.

In data mining, there are some issues to consider such as characteristics of data domain, how to clean dataset, how to convert data instances into vector form, which algorithm to use, which distance/similarity measure to use and so on. Text documents are data types written in free format and they are regulated only by grammar rules of the language they written. Because of this fact, processing text data with data mining methods has some difficulties. First of all, data instances must be in a structural form and have finite number of features in order to be processed by learning algorithms. When converting text documents into vectors, each distinct term in text collection takes part as a feature. Therefore, document vectors have large feature size, which causes complexity in terms of both space and time. The technique of converting text documents into vectors is an important issue. Secondly, document clusters does not always exist in form of easily separable compact structures. Different clusters may seem nested structures in text data collections. In such cases, ordinary clustering algorithms can hardly find clusters with high accuracy. They usually need additional processes to handle such datasets. Alternative approaches are needed to achieve high accuracy when clustering text datasets with nested groups.

In this study, it is aimed to improve text document clustering results by introducing some solutions to problems addressed. To begin with, as an alternative method to term based vectorization, using some reference documents is suggested for converting text documents into vectors. According to this method, each feature of a document vector is determined by its similarity to each reference document. Another suggestion is about algorithm. Spectral clustering algorithm is proposed to use when clustering datasets with nested clusters. This algorithm considers similarities/distances between instances of dataset instead of assuming a specific statistical model and optimizing parameters of that model. Algorithm performs well when clusters exist in form of connected components but not necessarily linearly separable compact structures. Another issue in text mining is to decide which distance measure to use. Cosine distance is picked as its usefulness in text data.

In scope of this thesis study, effects of reference documents and spectral clustering algorithm on text mining is examined. It is observed that these methods enhance

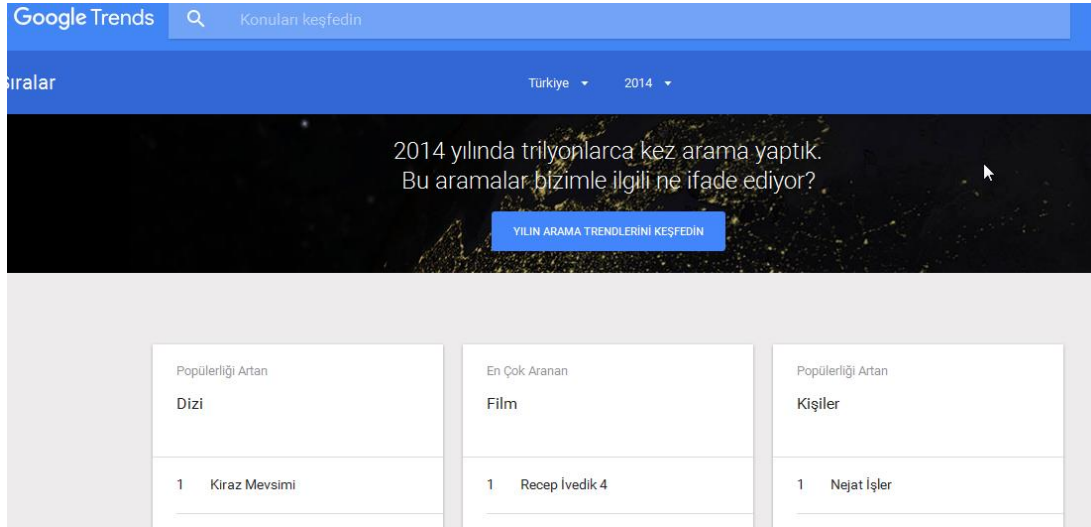
clustering accuracy. Two applications are developed in order to demonstrate effects. First one includes analysis of Turkish tweets using articles in Wikipedia as reference document set. In application, tweets are represented with respect to their similarities to Wikipedia articles and user timeline is reorganized according to relevancy of tweets to the user. In second application, 20-newsgroups data set is clustered using spectral clustering. Reference documents in this application are picked from central elements in dataset. Dataset is also clustered using k-means and spectral clustering algorithm with and without using reference documents. Then, results are compared.

Distributed storage and processing tools are used in applications in order to respond big data sets. Wikipedia articles are indexed on Apache Solr. By this way, scored document set result is provided for search queries consisting of tweets. On the other hand, Apache Hadoop's distributed file system is used in application running on 20-newsgroups dataset. Apache mahout is used for machine learning library. By this way, clustering algorithms are executed as MapReduce jobs on Hadoop.

1. GİRİŞ

İnternet ve bilişim teknolojilerinin erişilebilirliğinin giderek artmasının sonuçlarından biri de hem kullanıcılar, hem de içerik sağlayıcılar tarafından sürekli olarak oluşturulan verinin çok yüksek boyutlara erişmesidir. Bu verinin oldukça önemli bir kısmı serbest metinlerdir. Her gün sayısı takip edilemeyecek kadar fazla mail ya da anlık mesaj gönderilmekte, sosyal medyada paylaşımlar yapılmakta, haberler yayınlanmakta, bu haberlere yorumlar yazılmakta, birçok başka şekilde veri üretilmektedir. Günbegün artan, yalnızca oluşturulan verinin hacmi değildir. Aynı zamanda, veri üretenlerin bu veriler üzerinde arama yapma ve birtakım muayyen bilgileri elde etme isteği de artmaktadır. Büyük veri kümeleri içinde isabetli ve hızlı arama, bugünün hemen hemen bütün büyük ölçekli içerik sağlayıcıların sunmaya çalıştığı bir hizmettir. Bu konudaki güncel çalışmalar oldukça yaygındır. Öyle ki, büyük veri ortamlarında sunulan arama hizmetlerinde enerji verimliliği dahi ele alınan bir konu olmuştur (Abdelrahman ve Gelenbe, 2014).

Veri kümeleri içinde arama yapan kullanıcıların son zamanlardaki eğilimi, aradıkları bilgiye birkaç anahtar kelime ile ulaşma yönündedir. Aranılan veri tipi belli özelliklere göre kolayca filtrelenebilir olsa dahi kullanıcılar kelime bazlı arama yapmayı tercih edebilmekte ve alakalı sonuçlar görmeyi beklemektedirler. Bu nedenle metin verilerinden otomatize bir şekilde anlamlı sonuçlar çıkarabilmek önemli bir çalışma alanıdır. Bir metin dokümanına etiket/kategori tayin edebilmek ya da bir doküman yığını içinde benzer dokümanları gruplayabilmek, kelime bazlı arama uygulamalarında veri yığını küçültüp sorgu sonuçlarını daha alakalı hale getirmeyi sağlayacak çalışmalardır. Bu tür çalışmalar, arama motorlarının giderek daha da akıllı hale geldiği, soru&cevap motoru olarak hizmet verebilmeyi hedeflediği günümüzde bu anlamda önemli çabalardır. Şekil 1.1’de arama motoru olan Google’ın kullanıcıların arama ve tıklama kayıtları neticesinde yaptığı analizlerle, aranan kavram veya konseptin kategorisini (yemek, TV dizisi, kişi vs.) otomatik olarak tespit ederek sunduğu rapor gösterilmektedir.



Şekil 1.1 : Arama kayıtlarından otomatik tespit edilen kategoriler.

Metin dokümanlarının öbeklenmesi, güdüksüz öğrenme uygulamalarının ana dallarından biridir. Öğrenme uygulamalarında veri örneklerinin vektör formuna dönüştürülmesi öncelikli bir aşamadır. Metin dokümanları da genellikle metinde geçen kelimelerle oluşturulan vektörlerle ifade edilmektedir. Bu vektörlerin boyut sayısı, söz konusu doküman setinde geçen kelime sayısı ile belirlenmektedir. Doküman seti taranırken görülen her bir farklı kelime, vektör boyutunu artırmaktadır. Dolayısı ile doküman öbeklemedeki problemlerden biri, veri setine göre değişebilen ve genellikle yüksek sayılarda olan boyut sayısıdır. Bu durum veri işleme maliyetini ve bellek gereksinimini artırmaktadır. Öte yandan toplam kelime sayısının işlenecek veri setine göre farklılık göstermesi ve metinlerin yazıldığı dil içinde kısıtlı bir kapsama sahip olması da algoritmalar için saptırıcı olabilmektedir. Dokümanların vektörizasyonu konusunda metinlerde geçen kelimelerin kullanılmasına alternatif yaklaşımlardan biri, bir grup referans doküman kullanılmasıdır. Bu yönüme göre, veri setindeki dokümanlar, bu referans dokümanlara olan yakınlık / alaka oranları ile vektörize edilirler. Referans dokümanlar belirlenirken, veri setini ve metinlerin yazıldığı dili iyi düzeyde temsil edebilecek olmaları önemli bir ölçüdür.

Metin verilerinde birbirinden farklı kategoriler ayrık halde bulunmayabilmekte, A kategorisine ait dokümanlar ile B kategorisine ait dokümanlar iç içe geçmiş gibi görünebilmektedir. Farklı kategoriye/sınıfa ait veri öbeklerinin kompakt şekiller sergilemediği durumlarda, alışılagedik öbekleme algoritmaları yüksek isabetli sonuçlar üretememektedir. Spektral öbekleme algoritması, veri grupları kompakt

şekillerde olmadığı fakat bağlı bileşenler (connected components) halinde bulunduğu durumlarda etkili olabilen bir yöntemdir (Ng. ve diğ., 2001). Algoritma, veri örneklerinin ikili olarak birbirine yakınlığından oluşturulan yakınlık matrisini baz alarak bu matrisin özdeğerleri ve özvektörleri üzerinden çalışır. Algoritmanın görüntü segmentasyonu ve ses ayrımı konularında başarılı uygulamaları mevcuttur (Bach ve Jordan, 2006). Algoritmanın detaylı anlatımı için Alpaydın'ın kitabına (2014) müracaat edilebilir.

Bu çalışma, yukarıda bahsedilen problemlere odaklanan iki ayrı uygulamadan oluşmaktadır. İlk olarak, Twitter uygulamasında bir kullanıcıların takip ettiği kişiler tarafından yazılmış iletiler, kullanıcı ile alakalılık seviyesine göre bir sıralamaya tabi tutulmuştur. Bunun için hem kullanıcının kendi attığı tweetler, hem de takip ettiği kişiler tarafından atılan tweetler analiz edilmiştir. Bu uygulamada, Türkçe Vikipedi girdileri referans dokümanlar olarak kullanılmıştır. İkinci uygulamada ise, metin madenciliği çalışmalarında sıklıkla kullanılan 20-newsgroups veri seti üzerinde yine öbikleme çalışmaları yapılmıştır. Bu çalışmalarda, farklı öbikleme algoritmaları mukayese edilmiş ve yine referans dokümanlar kullanılmıştır. Bu uygulamadaki referans dokümanlar, veri serindeki merkezi elemanlar arasından seçilmiştir. İkinci çalışmada ayrıca spektral öbikleme algoritmasının etkileri ortaya konulmuş ve iyileştirici rolü olduğu düşünülen referans dokümanlarla vektörizasyon yöntemi bu algoritma ile birlikte sunulmuştur.

Sürekli kaydedilen ve işlenebilecek verinin çok yüksek miktarlarda olması, bu verilerin hem saklanması, hem de analiz edilmesi adına dağıtık mimaride çalışan büyük veri araçlarını kullanmayı gerekli hale getirmiştir. Bu bağlamda, tez kapsamında sunulan çözümler, dağıtık mimaride çalışan araçlar kullanılarak geliştirilmiş ve büyük veri problemleri üzerine de uygulanabilir haldedir. Birinci uygulamada, Vikipedi makalelerinden ters indeks (inverted index) oluşturulması ve saklanması, Apache Solr tarafından sağlanmaktadır. İkinci uygulamada ise, veriler Apache Hadoop tarafından sağlanan dağıtık dosya sisteminde saklanmış, öbikleme algoritmaları Apache Mahout tarafından sağlanan kütüphane kullanılarak yine Hadoop üzerinde MapReduce işleri şeklinde çalıştırılmıştır.

Çalışmanın bir sonraki bölümünde metin madenciliği, dağıtık veri öbikleme ve spektral öbikleme algoritması ile ilgili literatür araştırması sunulacaktır. 3. bölümde kullanılan veri setleri, araçlar ve yöntemler anlatılacak, 4. ve 5. bölümlerde ise

geliştirilen uygulamalar detaylı bir biçimde izah edilerek elde edilen deneysel sonuçlar değerlendirilecektir. 6. bölümde ise genel değerlendirmeler ve bu çalışmada sunulan çözümlerin daha da geliştirilmesi hakkında ileriye dönük öneriler ifade edilecektir.

1.1 Tezin Amacı

Bu tezin amacı, metin dokümanlarının öbeklenmesinde ölçeklenebilirliği ve isabet oranı yüksek yöntemler araştırmaktır. Bunun için dokümanların referans veri örnekleri ile belirlenmiş bir vektör uzayına aktarılarak modellenmesi ve spektral öbeleme yöntemlerinin, metin verileri üzerindeki başarısı incelenmiştir. Metotlar, doğal yollarla üretilmiş veri setleri (Twitter, Wikipedi, 20-newsgroups) üzerinde denenmiştir. Sunulan çözümlerin isabet oranına olan etkisinin yanı sıra ölçeklenebilir ve büyük veri setlerine de uygulanabilir olması da bu çalışmanın ana amaçlarından biridir.

2. LİTERATÜR ARAŞTIRMASI

Metin madenciliği ile ilgili çalışmalar, bilhassa bilgi erişimi (information retrieval) bakımından incelendiğinde 1990'lı yıllardan beri süregelen bir süreçtir (Grimes, 2007). Bu alandaki en kapsamlı literatür tarama çalışmalarından biri Berry (2003) tarafından yapılmıştır. Bu çalışmada ilk olarak veri setindeki dağılım ve gruplaşma/ayırışma korunarak boyut azaltma / özellik seçimi yöntemleri ortaya konulmuştur. Daha sonra, metin dokümanlarının vektör uzay modeli ile ifade edilmesi konusu ele alınarak, metin verilerinin vektörizasyona dair farklı öneri ve yaklaşımlar ele alınmıştır. Bu tarama çalışmasının odaklandığı bir diğer nokta ise istatistiksel yöntemler kullanılarak metin analizi, içerik ve eğilim tespitidir. Sathiyakumari ve diğerleri (2011) tarafından yapılan incelemede ise, k-means, hiyerarşik öbikleme gibi klasik algoritmaların doküman öbiklemedeki avantaj ve dezavantajları ortaya konulmaktadır. Thangamani ve diğerleri (2010) tarafından ortaya konulan literatür incelemesinde ise yine metin madenciliğindeki farklı yaklaşımlar ele alınarak gelecek çalışmalar için bir çerçeve sunulmuştur. Svadas ve diğerleri (2014) tarafından yapılan çalışmada, metin öbikleminin semantik web yaklaşımı ile geliştirilmesine dair bir tarama yapılmıştır. Aggarwal ve Zhai (2012), doküman öbikleme yöntemleri ve aşılması gereken problemler hususunda oldukça kapsamlı bir tarama ortaya koymuşlardır. Bu çalışmada doğru özellik seçimi / boyut azaltma metodu ve doğru algoritma seçimlerinin metin öbiklemedeki önemi vurgulanmıştır. Steinbach ve diğerleri (2000) tarafından yapılan karşılaştırma çalışmasında ise en bilindik güdümsüz öğrenme yöntemlerinden olan k-means ve hiyerarşik öbikleme algoritmalarının metin verileri üzerindeki performansları karşılaştırılmıştır.

Dokümanların vektörel forma çevrilmesinde, klasik VSM ve TF-IDF yaklaşımlarına alternatif birtakım öneriler de mevcuttur. Reed ve diğerleri (2006) tarafından ortaya konulan TF-ICF (Term Frequency – Inverse Corpus Frequency) adlı doküman-vektör dönüşümü modeline göre, bir dokümanın vektöründe bir terimin / kelimenin ağırlık değerinin bulunması için diğer dokümanların taranmasına gerek yoktur. Buna göre, doküman seti vektörel forma lineer bir zamanda dönüştürülebilir. Çalışmada

belirtildiğine göre önerilen yöntem öbekleme başarısını düşürmeden, diğer vektörizasyon yöntemlerine göre daha hızlı bir şekilde dokümanların matematiksel forma dönüştürülmesini sağlamaktadır. Diğer bir çalışmada ise Vikipedi dokümanları bir bilgi kaynağı olarak kullanılmıştır (Lu ve diğ., 2012) . Bu bilgi kaynağı kullanılarak Twitter'daki kullanıcının kendi Tweet'leri ve takip ettiği kişilerin yazdığı Tweet'ler analiz edilerek, açılış sayfasında görünen Tweet'ler bu analize göre yeniden sıralanmıştır. Önerilen yöntem, bu çalışmada sunulan uygulamada kullanılan yöntemle oldukça benzer olmakla beraber birtakım farklılıklar mevcuttur. Öncelikle çalışmada ortaya konulan yöntemlerde Vikipedi dokümanlarının ESA (Explicit Semantic Analysis) algoritması ile Tweet'lerden Vikipedi konuları belirlenmekte ve random walk algoritması ile bir genişletme yapılmaktadır. Bizim çalışmamızda ise Apache Lucene tabanlı Solr ile Vikipedi dokümanlarının ters indeksi oluşturulmuş ve Tweet'ler, arama sorgularımız gibi Solr'dan skorlu arama sonuçları alınmıştır. Diğer bir fark ise, bizim çalışmamızda modellemede Vikipedi konsept ya da kategorileri yerine doğrudan dokümanların kullanılmasıdır. Son olarak, bahsedilen çalışmada Tweet ve dokümanlar İngilizce iken bizim uygulamamız Türkçe veriler üzerinde gerçekleştirilmiştir.

Metin dokümanlarının sınıflandırılması ya da öbeklenmesi problemlerinde önemli bir mesele yakınlık/uzaklık ölçüsüdür. Bu konuda da Huang (2008) tarafından yapılmış bir mukayese çalışmasında standard k-means algoritması öklid, kosinüs ve izafi entropi ölçüleri kullanılarak çalıştırılmış ve bu uzaklık ölçülerinin isabet oranına olan etkileri karşılaştırılmıştır.

Metin madenciliğindeki çalışmaların yoğunlaştığı bir diğer konu ise özellik seçimidir (feature selection). Metin madenciliğinde özellik seçimi konusunda yapılan çalışmaların önemli bir kısmı semantik analiz ve doğal dil işleme yöntemlerini de içermektedir fakat bu konular çalışmamızın kapsamı dışındadır. Bu çalışmaya yalnızca basit veri temizleme yöntemleri ve veri madenciliği teknikleri dahil edilmiştir. Riberio ve diğerleri (2008) tarafından yapılan çalışma hiyerarşik öbeklemede lokal özellik seçimini önermektedir. Buna göre veri örneklerinin ifade edileceği özellik seti hiyerarşik öbeklemeden sonraki ait olunan grup içinde belirlenecektir. Yine Ienco ve Meo (2008) tarafından UCI veri setlerinin önemli bir kısmı kullanılarak yapılan araştırmada, hiyerarşik yöntemle tüm özelliklerin gruplara ayrılması önerilmiştir. Gruplanmış alt kümelerde bir araya gelen özellikler

kullanılarak seçim yapılmış ve başarılı sonuçlar elde edilmiştir. Diğer bir çalışmada ise, EM algoritması ile öbekleme işleminin iterasyonları sırasında meydana gelen ara çıktılar kullanılarak özellik seçimi yapılmış ve bu yöntemin başarılı sonuçlar ortaya koyduğu ifade edilmiştir (Xu ve diğ., 2007). Cai ve diğerleri (2005) tarafından yapılan bir çalışmada ise boyut azaltma / özellik seçme konusunda LPI yöntemi önerilmiş ve Reuters-21578 veri seti üzerinde yapılan testlerde yöntemin iyi sonuçlar verdiği ifade edilmiştir.

Spektral öbekleme algoritmasının metin vereri üzerindeki pratiklerini konu alan birtakım çalışmalar da mevcuttur. Vempala ve Wang (2005), bu çalışmalardan birini gerçekleştirmişlerdir. Doküman öbeklemede spektral yöntemlerin klasik yöntemlere göre daha iyi sonuç verdiğini öne sürmüş ve spektral projeksiyon neticesinde grupların daha belirgin hale geldiğini savunmuşlardır. Diğer bir araştırmada, dokümanlar ve kelimelerin iki ayrı kümede toplanarak aralarında oluşturulacak iki parçalı graf üzerinde spektral öbekleme algoritmasının uygulanması önerilmiş ve ampirik sonuçlarla bu öneri desteklenmiştir (Dhillon, 2001) . Bao ve diğerleri (2008) tarafından yapılan çalışmada ise veri örnekleri arasındaki benzerlik matrisinin ortogonal pozitif çarpanlarına ayrılması yöntemi ile spektral öbekleme algoritmasındaki klasik özvektör yaklaşımına alternatif bir çözüm sunulmaktadır. Başka bir çalışmada ise, spektral öbekleme algoritmasının merkezi elemanlar üzerinde çalıştırılması öne sürülmüştür. (Yan ve diğ, 2009). Bu uygulama, mevcut tez kapsamında yapılan uygulamalardan biriyle benzerlik göstermektedir fakat önerilen yöntemde, sadece merkezi elemanlar öbekenerek isabet oranında makul bir düşüş göze alınıp algoritmanın çalışma hızında bir artış hedeflenmektedir. Buna karşın, tarafımızdan sunulan uygulamada makul bir zaman yükü ile spektral öbekenmenin isabetini arttırmak hedeflenmektedir. Merkezi elemanlar ise doğrudan öbeklemede kullanılmak yerine, diğer elemanların özellik seçimi konusunda bir referans kümesi olarak ele alınmaktadır.

“Big Data” kavramının yaygınlaşması, veri madenciliğinde bu kapsama girecek boyutlardaki veri kümelerinin işlenmesi ile ilgili dağıtık yöntemlerin tartışılmasının da önünü açmıştır. Judith ve Jayakumari (2015), dağıtık öbekleme algoritmalarına dair güncel çalışmaları taramışlardır. Çalışmada, mevcut dağıtık çözümler hız, öbek kalitesi, ölçeklenebilirlik ve isabet oranı gibi kıstaslara göre mukayese edilerek sunulmuştur. Başka bir çalışmada P2P ağlarda dağıtık veri madenciliği ele alınarak,

dağıtık ortamlarda daha etkin lokal analiz yapılması ve düğümler arası mesajlaşmanın minimize edilmesi konuları tartışılmıştır (Datta ve diğ, 2006). Li ve diğerleri (2014) ise yaptıkları çalışmada k-means algoritmasının MapReduce modeli ile dağıtık mimaride çalışmasını daha verimli hale getirmeye odaklanmışlardır. Verinin etkin bir hash algoritması ile parçalanması, başlangıç merkezlerinin etkin bir yöntemle belirlenmesi ve gereksiz uzaklık hesaplamalarından tasarruf edilmesi gibi yaklaşımlarla k-means algoritmasını dağıtık mimaride daha verimli hale getirmeyi amaçlamışlardır.

3. MALZEME VE YÖNTEM

3.1 Uygulamalarda Kullanılan Veri Kümeleri

3.1.1 Vikipedi Dokümanları

Vikipedi, internet tabanlı ansiklopedik bir bilgi kaynağıdır. Aranan her bir kavram ayrı bir web sayfasında sunulmaktadır. Bu sitede sunulan içerik kullanıcılar tarafından oluşturulmaktadır. Dolayısıyla bu sitedeki sayfalara sürekli düzeltmeler ve eklemeler yapılmaktadır. İçeriğin kullanıcılar tarafından oluşturulmasının olumsuz bir sonucu, zaman zaman doğruluğu tartışılır tarifler ortaya çıkmasıdır. Buna rağmen oldukça yaygın kullanılması ve geniş bir kullanıcı kitlesine hitap etmesi sebebiyle sayfalar hızlı bir şekilde kullanıcılar aracılığıyla tashih edilmektedir. Şekil 3.1’de Vikipedi’den alınan bir ekran görüntüsü mevcuttur.



Şekil 3.1 : Örnek bir Vikipedi sayfası.

Vikipedi’de yer alan sayfalar ansiklopedik bilgi içerdiği için genellikle muntazam bir üslup ve objektif bir dille yazılmışlardır. Bu nedenle, referans doküman olarak kullanılmaya oldukça elverişli metinler olarak değerlendirilmektedir. Vikipedi veritabanı, zaman zaman indirilebilir paketler halinde belli bir dil için o esnadaki tüm sayfaları kapsayacak şekilde hazırlanmaktadır. Bu tez kapsamındaki uygulamalarda kullanılan Vikipedi paketi, 226460 adet Türkçe Vikipedi dokümanını kapsayan veri kümesidir.

3.1.2 Twitter iletileri

Twitter, kullanıcılarına belli sınırlı karakterde iletiler (tweet) yazarak bunları paylaşmayı sağlayan bir sosyal ağdır. Bu anlamda bir mikroblog olarak da adlandırılabilen bir paylaşım sitesi olan Twitter, sosyal medya uygulamaları arasında en yaygın olanlarından biridir. Gerçek kişilerin yanı sıra, tüzel kişilerin de kurumsal iletişim amacıyla kullandığı bir ağdır. Twitter hesabı bulunan, kullanıcılar iletilerini görmek istedikleri kişileri takip edebilmekte ve diğer kullanıcılarla bu iletiler üzerinden etkileşim kurabilmektedir. Kullanıcılar Twitter'a giriş yaptıklarında ana sayfa olarak takip ettikleri kişilerin yazdıkları tweetler zamana göre sıralı (en yeni olan en üstte) olarak gösterilir.

Twitter'da oluşturulan tweetler, paylaşan kişinin üslubuna ve görüşlerine bağlı olduğu için biçimsel açıdan düzgün bir dil kullanıldığı ve objektif bir bakış açısı ile yazıldığı çoğu zaman söylenemez. İstihza ve tariz içerebilir. Yalnızca paylaşan kişiyi bağladığı için iletilerin kollektif bir şekilde düzeltilmesi de söz konusu değildir. Karakter sınırlamasından ötürü, kelimeler bazen sesli harfler çıkarılarak kısaltılabilmekte yahut belli bütünlükte bir paylaşım, birkaç ileti ile ancak ifade edilebilmektedir. Kısacası Twitter kullanıcıları tarafından yazılan iletiler, analiz edilmek için bir normalizasyon sürecine girmeye muhtaç metin verileridir.

3.1.3 20-Haber grubu veri seti (20-Newsgroups)

20-Newsgroups dokümanları, University of Carolina – Irvine (UCI) bünyesindeki Yapay Öğrenme ve Akıllı Sistemler Merkezi tarafından, bilimsel çalışmalarda kullanılması adına sağlanan veri kümelerinden biridir. Bir haber grubunda paylaşılmış 20 kategoriye ait toplam 18846 adet İngilizce metin dokümanından oluşmaktadır. Bu dokümanlarda toplam 93560 farklı kelime geçmektedir. Bu rakam, herhangi bir doğal dil işleme yöntemi kullanılmadan (kelimeleri eklerden arındırma, yaygın kelimeleri eleme vb.) elde edilen rakamdır. Bu haber grubu kategorilerinin bir kısmı oldukça yakın kategoriler, bir kısmı ise alakasız, birbirine uzak kelimelerdir. Veri setindeki kategoriler, Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1 : 20-Newsgroups veri setindeki kategoriler.

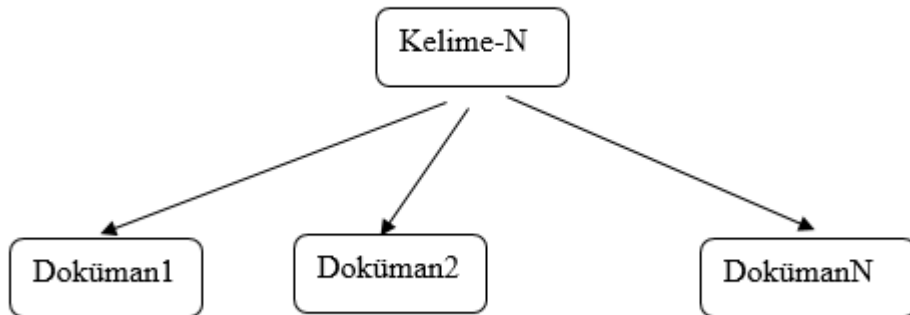
comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey
sci.crypt sci.electronics sci.med sci.space	talk.politics.misc talk.politics.guns talk.politics.mideast
talk.religion.misc alt.atheism soc.religion.christian	misc.forsale

3.2 Uygulamalarda Kullanılan Dağıtık Veri İşleme ve Saklama Araçları

3.2.1 Apache Solr

Apache Solr, metin dokümanlarının indekslenerek arama motorlarının temel işlevi olan anahtar kelime bazlı arama hizmetini sunan, açık kaynak kodlu özgür bir yazılımdır. Lucene tabanlı çalışan Solr’da, dokümanların ters indekleri (inverted index) oluşturulur. Buna göre, alışlageldik veri dosyası indeksleme işlemindeki “dokümanda geçen kelimelerin indekslenmesi” yerine “kelimenin geçtiği dokümanların indekslenmesi” yöntemi uygulanır. Bu mantık şekil 3.2’de gösterilmiştir. Kelime bazlı arama uygulamalarında, aranılan kelimelere karşılık gelen sonuç listesinin büyük bir veri seti içinde az bir zamanda bulunarak kullanıcıya dönülmesinin bu mekanizma sağlamaktadır. Apache Solr’ın 4.5.1 numaralı versiyonu, 4. Bölümde bahsedilen uygulamada kullanılmıştır.

✓



Şekil 3.2 : Ters indeks mantığı.

Solr indeksleme motoru olarak Apache Lucene kullanan ve kendine has bir sunucu üzerinde çalışan bir yazılımdır. Yazılım geliştiriciler için birçok dilde erişim kütüphanesi bulunmakla birlikte, bir Web arayüzü de mevcuttur. Şekil 3.3’te, Solr üzerinde indekslenmiş dokümanlar arasından “sosyal ağ” kelimeleri ile arama sorgusu gönderildiğinde sonuç olarak döndürülen dokümanlar gösterilmektedir. Solr, kendisine bir arama geldiğinde, aratılan kelimelerle ilgili dokümanlara birer skor verir ve döndüğü sonuç kümesini bu skora göre sıralayarak döner. Buna göre, Solr’a gönderilen bir arama sonucunda skoru daha yüksek olan doküman daha üstte olacak şekilde bir sonuç kümesi elde edilir.

The screenshot shows the Apache Solr web interface. On the left is a navigation menu with options like Dashboard, Logging, Core Admin, Java Properties, Thread Dump, social (selected), Overview, Analysis, Config, Dataimport, Documents, Ping, Plugins / Stats, Query (selected), Replication, Schema, and Schema Browser. The main area is titled 'Request-Handler (qt)' and shows a search query 'q=sosyal ağ'. Below the query are fields for fq, sort, start, rows, fl, df, Raw Query Parameters, wt (set to json), and checkboxes for indent, debugQuery, and dismax. On the right, the JSON response is displayed, showing search results for 'sosyal ağ' and 'sosyal ağ, film'.

```

{
  "responseHeader": {
    "status": 0,
    "QTime": 339,
    "params": {
      "indent": "true",
      "q": "sosyal ağ",
      "_: "1470637083068",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 9125,
    "start": 0,
    "maxScore": 8.150859,
    "docs": [
      {
        "id": "170349",
        "title": "sosyal, ağ",
        "content": "diger, anlami, sosyal, ađlar, bireyleri, internet, üzerinde, toplum, yařamı, içinde, kendilerini, tanımlayar",
        "_version_": 1470349968913989600,
        "score": 8.150859
      },
      {
        "id": "168887",
        "title": "sosyal, ağ, film",
        "content": "diger, anlami, film, sosyal, ağ, özgün, social, network, yapımı, david, fincher, imzalı, başrollerini, jesse, e",
        "_version_": 1470349963472928800,
        "score": 6.520687
      }
    ]
  }
}

```

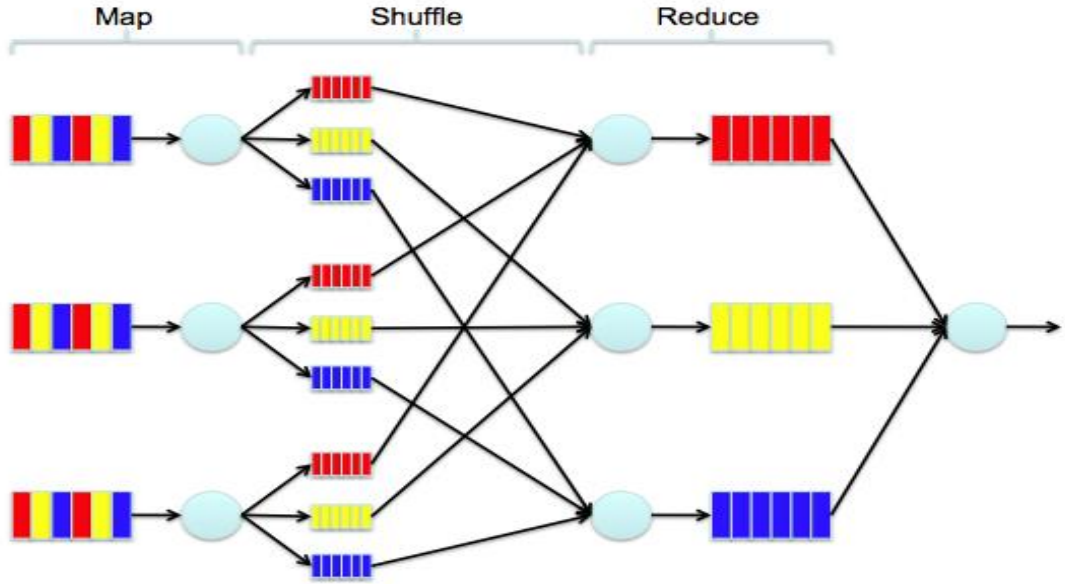
Şekil 3.3 : Solr web arayüzünden “sosyal ağ” arama sonuçları.

3.2.2 Apache Hadoop

Apache Hadoop, büyük veriler üzerinde dağıtık hesaplama yapmak ve bu verileri dağıtık bir ortamda saklamak için elverişli bir yazılımdır. Hadoop iki ana bileşene ayrılabilir. Bunlardan ilki, daha önce Google tarafından ortaya atılan (2004) ve bir çok problemin çözümünde model olarak kullanılacak MapReduce modelidir. Bu kısım, Hadoop’un işlem yapma / veri işleme ile ilgili bileşenidir. İkinci bileşen ise dağıtık dosya sistemidir (HDFS).

MapReduce dağıtık problem çözme modeli, Map ve Reduce olarak iki fazdan oluşmaktadır. Map ve Reduce adımları, fonksiyonel programlama dillerinin birçoğundaki aynı isimli fonksiyonlarla oldukça benzer mantıkta işlemektedir. Map fazında, dağıtık sisteme dahil olan her bir düğüm kendi içinde (diğer düğümlerden bağımsız) bir işlem yaparak lokal ara sonuçlar üretir. Daha sonra ise ara sonuçlar

toplanıp birleştirilerek Reduce fazındaki işlem sonucunda veriler nihai global sonucu teşkil edecek forma getirilir. Şekil 3.4'te, MapReduce mantığı basit bir çizimle anlatılmıştır. Bu tez kapsamında yapılan çalışmalarda, Hadoop tam dağıtık modda çalıştırılmamış, psödo-dağıtık ayar ile koşulmuştur zira bu çalışmanın birincil odak noktası öbikleme algoritmalarının isabet oranına etkisinin incelenmesidir. Bu çalışmada, Hadoop'un 1.2.1 versiyonlu dağıtımını kullanılmıştır.



Şekil 3.4 : MapReduce çalışma mantığı.

3.2.3 Apache Mahout

Apache Mahout, büyük veriler üzerinde öğrenme algoritmaları çalıştırmak için geliştirilmiş bir yapay öğrenme kütüphanesidir. Mahout, Hadoop ile entegre çalışarak, öbikleme ya da sınıflandırma algoritmalarını MapReduce işleri şeklinde gerçekleştirir. Mahout, yapay öğrenme algoritmalarına ek olarak, bazı kullanışlı veri işleme süreçlerini de sunmaktadır. Örneğin büyük bir metin dokümanı kümesinden, TF- IDF vektörleri oluşturmak için bir komut sunmaktadır. Çalıştırılan Mahout uygulamasının sürüm numarası 0.9.0'dır ve kullanılan Hadoop sürümü ile entegre çalışmaktadır.

4. VİKİPEDİ DOKÜMANLARI REFERANS ALINARAK TWITTER İLETİLERİNİN ANALİZİ

Bu tezde vurgulanan, metin dokümanların vektörel forma çevrilmesinde yine metin dokümanlarının referans alınması uygulamasının ilk örneği bu bölümde yer almaktadır. Bu bölümde sunulacak olan uygulamanın amacı, Twitter kullanıcılarını analiz ederek giriş sayfalarında görünen tweetlerin, kendileri ile daha alakalı olanlar daha üst sıralarda görülecek şekilde yeniden sıralanmasıdır. Hem kullanıcıların analizinde, hem de kullanıcıların takip ettiği kişiler tarafından yazılan iletilerin analizinde Wikipe di makaleleri kullanılmıştır. Uygulama, Apache Solr ve MySQL veritabanı kullanılarak Java ile geliştirilmiştir.

4.1 Twitter Verisine Erişim

Twitter, kullanıcıları tarafından üretilen içeriğe erişilmesi adına çeşitli uygulama programlama arayüzleri (API) paylaşmaktadır. Kullanıcıların gizlemek istediği tweetler bu erişimin kapsamı dışında tutulmaktadır. Java ile geliştirilen uygulamalar için Twitter4J adlı kütüphane kullanılmaktadır. Twitter'dan veri çekebilmek için herşeyden önce bu siteden veri çeken bir uygulama geliştirildiği beyan edilerek bir API anahtarı elde edilmelidir. Twitter'dan veri çeken API'ler, ancak geçerli bir API anahtarı ile çalışabilmektedir. Twitter'dan veri çekmeyi gösteren kod parçası Şekil 4.1'de verilmiştir. Bu kod parçasında işaretlenerek vurgulanmış değişkenler Twitter'dan alınmış geçerli API anahtarları olmalıdır. Örnek kod parçasında Twitter'dan bir liste halinde veri getirilmekte ve bu veriler veritabanına yazılmaktadır.

```

import twitter4j.Paging;
import twitter4j.ResponseList;
import twitter4j.Status;
import twitter4j.Twitter;
import twitter4j.TwitterException;
import twitter4j.TwitterFactory;
import twitter4j.conf.ConfigurationBuilder;

import com.pala.social.service.StatusService;
import com.pala.social.util.ApiKeys;
import com.pala.social.util.Constants;

public class Twitter4jExample {

    public static void main(String[] args) {

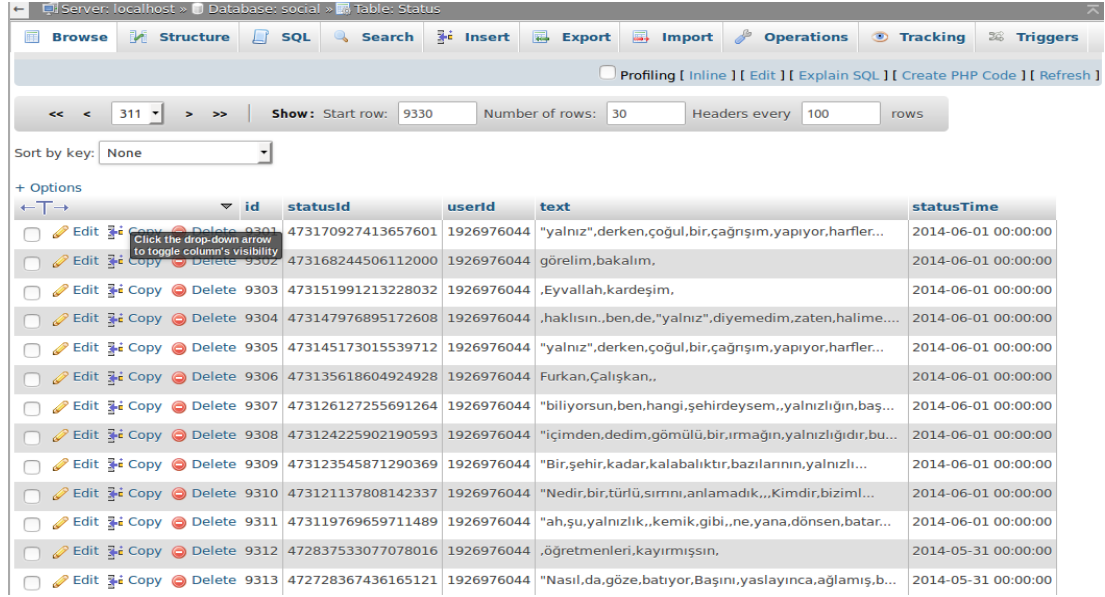
        ConfigurationBuilder cb = new ConfigurationBuilder();
        cb.setDebugEnabled(true);
        cb.setOAuthConsumerKey(ApiKeys.CONSUMER_KEY);
        cb.setOAuthAccessToken(ApiKeys.ACCESS_TOKEN);
        cb.setOAuthAccessTokenSecret(ApiKeys
            .ACCESS_TOKEN_SECRET);
        TwitterFactory tf = new TwitterFactory(cb.build());
        Twitter twitter = tf.getInstance();
        Paging paging = new Paging(Constants.SELF_TWEETS_PAGES,
            Constants.PAGE_SIZE);
        for (int i = 0; i < Constants.SELF_TWEETS_PAGES; i++) {
            paging.setPage(i + 1);
            ResponseList<Status> statusList = null;
            try {
                statusList =
                    twitter.getUserTimeline("palaevren", paging);
            } catch (TwitterException e) {
                e.printStackTrace();
            }
            for (Status status : statusList) {
                StatusService.insert(status);
            }
        }
    }
}

```

Şekil 4.1 : Twitter'dan bir kullanıcıya ait tweetlerin çekilmesi (java).

Çekilen tweetler, MySQL veri tabanına yazılmıştır. Bu tabloda bulunan verilerin örnek bir görüntüsü şekil 4.2'de gösterilmiştir. Şekildeki tabloda tweetler, yazar kişinin numarası, tweet numarası (bu numaralar eşsiz belirteç olarak kullanılmaktadır), metin içeriği ve oluşturulma tarihi kolonları ile saklanmıştır.

Bunlara ek olarak, bu kayıtlara tarafımızca verilen bir numara alanı da söz konusudur.



id	statusid	userid	text	statusTime
9301	473170927413657601	1926976044	"yalnız",derken,çoğul,bir,çağrışım,yapıyor.harfler...	2014-06-01 00:00:00
9302	473168244506112000	1926976044	görelim,bakalım,	2014-06-01 00:00:00
9303	473151991213228032	1926976044	,Eyvallah,kardeşim,	2014-06-01 00:00:00
9304	473147976895172608	1926976044	,haklısın.,ben,de,"yalnız",diyemedim,zaten,halime....	2014-06-01 00:00:00
9305	473145173015539712	1926976044	"yalnız",derken,çoğul,bir,çağrışım,yapıyor.harfler...	2014-06-01 00:00:00
9306	473135618604924928	1926976044	Furkan,Çalışkan.,	2014-06-01 00:00:00
9307	473126127255691264	1926976044	"biliyorsun,ben,hangi,şehirdesem.,yalnızlığın,baş...	2014-06-01 00:00:00
9308	473124225902190593	1926976044	"içimden,dedim,gömülü,bir,ırmağın,yalnızlığıdır,bu...	2014-06-01 00:00:00
9309	473123545871290369	1926976044	"Bir,şehir,kadar,kalabalıktır,bazılarının,yalnızlı...	2014-06-01 00:00:00
9310	473121137808142337	1926976044	"Nedir,bir,türü,sırmını,anlamadık,..Kimdir,biziml...	2014-06-01 00:00:00
9311	473119769659711489	1926976044	"ah,şu,yalnızlık.,kemik,gibi.,ne,yana,dönsen,batar...	2014-06-01 00:00:00
9312	472837533077078016	1926976044	,öğretmenleri,kayırmışsın,	2014-05-31 00:00:00
9313	472728367436165121	1926976044	"Nasıl,da,göze,batıyor,Başını,yaslayınca,ağlamış,b...	2014-05-31 00:00:00

Şekil 4.2 : Veritabanına kaydedilmiş tweetler.

4.2 Vikipedi Dokümanlarının Solr ile İndekslenmesi

Apache Solr, dışarıdan veri içe aktarma konusunda farklı özelliklere sahiptir. Bunlardan bir kısmı, bilindik dosya sistemlerinin içindeki dosyalardan veri indekslenmesidir. Düz metin belgeleri, ya da farklı alanlar da Solr indeksi üzerinde saklanmak isteniyorsa XML dosyaları Solr'a yüklenebilir. Verilerin bir diğer içe aktarma yöntemi ise veritabanı üzerinden aktarımdır. Vikipedi dokümanları MySQL veritabanında bir tabloya yazıldıktan sonra Solr tarafından sağlanan "DataImportHandler" modülü aracılığı ile indekslenmiştir. İndekse alınan Vikipedi dokümanları üç alandan oluşmaktadır:

- id: Dokümana has eşsiz bir numara. Bu alanın aramalarda bir işlevi yoktur.
- title: Vikipedi'deki sayfa başlığı
- content: metin içeriği

Dokümanlar indekslenmeden önce birtakım filtrele ve analizlerden geçmektedir. Normalizasyon sağlamayı amaçlayan bu filtrelerin Solr üzerinde nasıl ayarlandığı Şekil 4.3'te gösterilmiştir. Bu konfigürasyon dosyası, Solr dağıtımının içinde var olan "solrconfig.xml" adlı dosyadır. Bahsedilen filtrelerden birisi, bütün kelimelerin küçük harflere çevrilerek bu şekilde indekslenmesidir. Böylelikle, büyük-küçük harf

duyarlılığı ortadan kaldırılarak normalizasyon sağlanmıştır. Örneğin, “Bahçe” ve “bahçe” terimlerinin birbiri ile aynı olduğu, indeksleyici için anlaşılır olmuştur. Şekilde gösterilen ayarlardan “solr.TurkishLowerCaseFilterFactory” adlı ayar bu işlevi görmektedir. Diğer bir filtre ise edatlar, bağlaçlar gibi yaygın kelimelerin elenmesidir. “Ama”, “de”, “gibi”, “yine”, vs. kelimeler, metin dokümanının kapsamı ve vurgusu konusunda bir anlam ifade etmeyen, yalnızca anlatıma yardımcı olma noktasında rolü olan kelimelerdir. Bu kelimelerin bir dokümanda fazla yahut az geçmesinin bahsedilen kavramlar üzerinde bir etkisi yoktur. Dolayısıyla bu kelimeler kirlilik ve karışıklığı ortadan kaldırmak, basitlik sağlamak adına indeksleme işlemi sırasında elenerek dikkate alınmaz. Bu ayar, aşağıdaki şekilde “solr.StopFilterFactory” ismiyle verilmiş ve Türkçe için elenecek kelimeleri barındıran “stopwords_tr.txt” dosyasının adı parametre olarak geçilmiştir. Şekil 4.3’te görülen bir diğer filtre ise, “solr.SnowBallPorterFilterFactory” adıyla ayarlanmış olan köklere ayırma filtresidir. Bu filtre kelimelerin eklerini çıkararak köklerine iner ve bu şekilde indekslenmelerini sağlar. Örneğin “çarşı”, “çarşıya” ve “çarşıda” kelimeleri, aynı kelime (çarşı) olarak indekslenir. Aynı kelime herhangi bir ek ile arandığında da kelime kökünden dolayı isabet edildiği, indekslemede kullanılan bu filtre ile tespit edilebilir. Aksi halde, “çarşıya” diye aratıldığında, içinde “çarşı” geçen dokümanlar gözardı edilecektir.

```
<!-- Turkish -->
<fieldType name="text_tr" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.TurkishLowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="false" words="lang/stopwords_tr.txt" />
    <filter class="solr.SnowballPorterFilterFactory" language="Turkish"/>
  </analyzer>
</fieldType>
```

Şekil 4.3 : İndekslenecek dokümanlara uygulanan filtreler.

Vikipedi dokümanları yukarıda belirtilen filtrelerden geçerek Solr indeksine aktarılmıştır. Filtrelerin tanımlanmasında dikkat edilecek önemli bir husus, doküman metinleri ile sorgu metinlerinin aynı filtre ve analizlerden geçmesidir. Bu ön işleme süreçlerinin sebebi standart bir form belirlemek olduğu için, bu standart form hem doküman kümesindeki metinler, hem de sorgu olarak gelen metinler için aynı olmalıdır. Dolayısıyla yapılan çalışmalarda, bu ayarlar sorgular ve indekslenmiş dokümanlar için aynı şekilde tanımlanmıştır.

4.3 Vikipedi Dokümanları Baz Alınarak Tweet'lerin Yeniden Sıralanması

Metin dokümanlarının vektörel forma dönüştürülmesinde kendileri yine de metin belgesi olan birtakım referans dokümanlarının kullanılması, daha önce de bahsedildiği gibi bu çalışmada etkisi vurgulanmak istenen asıl yöntemdir. Bu bağlamda, Twitter kullanıcıları tarafından yazılan metin verileri, Vikipedi dokümanları referans olarak kullanılarak ifade edilecektir. Zira mikroblog olarak da nitelendirilebilecek Twitter platformunda tweetler 140 karakterle sınırlı, genellikle sübjektif, serbest bir üslup ve imla ile yazılan metinlerdir. Öte yandan Vikipedi dokümanları, muntazam bir dille yazılan, karakter sınırlaması olmayan, ansiklopedik bilgi içeren, sübjektif ya da yanlış bilgilerin sıklıkla düzeltildiği metin belgeleridir. Dolayısı ile Vikipedi belgelerinin Twitter dokümanlarını hem şeklen normalize etmek, hem de içerik olarak zenginleştirmek adına referans olarak kullanılabilirliği değerlendirilmektedir.

Bu uygulamanın hedefi, Twitter kullanıcılarını yazdığı tweetlere göre analiz ederek, onların Twitter'a giridklerinde sayfalarında görünen (takip ettiği kişiler tarafından atılmış) tweetleri yeniden sıralamaktır. Bu yeniden sıralama, o anda zaman tüneline (Timeline) görülen tweetlerin, kullanıcı ile benzerliğine göre yapılacaktır. Bu tarife göre kullanıcıları ve tweetleri vektörel bir forma dönüştürmek ve bir benzerlik ölçüsüne sahip olarak benzerliğin nümerik değerini elde etmek gereklidir.

Öncelikle uygulama kapsamında Apache Solr indeksine alınan vikipedi dokümanları, tweetler ile sorgulanmıştır. Diğer bir deyişle, yazılan tweetler sanki arama motorunun arama kutucuğuna yazılan kelimelermiş gibi aratılmış ve bu arama Vikipedi dokümanları üzerinde yapılarak alakalılık skorları elde edilmiştir. Bu uygulama Java ile yazılmış bir uygulama olduğu için Solr arama sunucusuna erişim de Java için oluşturulmuş SolrJ kütüphanesi üzerinden sağlanmıştır. Bu şekilde nasıl arama sorgusu yapıldığı ve sonuç olarak doküman ve skorların nasıl elde edildiğini gösteren Java kodu Şekil 4.4'te gösterilmiştir. Buradaki örnek kod parçacığında, java kodunun çalıştığı makine ile aynı ortamda (localhost) ve 8983 numaralı port üzerinde çalışan Solr sunucusunda "bilgisayar bilimi" kelimeleri aratılarak, sonuç olarak döndürülen dokümanların "id" alanı (kendine has sıra numarası) ve arama kelimelerine göre Solr tarafından hesaplanan alakalılık skorları ekrana bastırılmaktadır. "id" alanının Solr tarafından tanınması için, Vikipedi verileri içe

aktarılmadan önce, bölüm 4.2’de üç madde halinde verilen alanların ayarlanarak Solr’a tanıtılması gerekmektedir.

```
import org.apache.solr.client.solrj.SolrQuery;
import org.apache.solr.client.solrj.SolrServer;
import org.apache.solr.client.solrj.SolrServerException;
import org.apache.solr.client.solrj.impl.HttpSolrServer;
import org.apache.solr.common.SolrDocument;
import org.apache.solr.common.SolrDocumentList;

public class SolrExample {
    public static void main(String[] args) {
        SolrServer solrServer =
            new
HttpSolrServer("http://localhost:8983/solr/social");
        SolrDocumentList results = null;
        SolrQuery query = new SolrQuery();
        query.setQuery("bilgisayar bilimi");
        query.setFields("id", "title", "score");
        query.setStart(0);
        query.setRows(226460);
        try {
            results = solrServer.query(query).getResults();
        } catch (SolrServerException e) {
            e.printStackTrace();
        }
        for(SolrDocument doc : results){
            System.out.println((String) doc.getFieldValue("id")
                + ":" + (Float)
doc.getFieldValue("score"));
        }
    }
}
```

Şekil 4.4 : Solr arama sunucusuna sorgu gönderilmesi.

Bir tweet, aşağıdaki formda vektöre çevrilmektedir:

$$t = \{d_1:s_1, \quad d_2:s_2, \quad \dots \quad d_{226460}:s_{226460}\} \quad (3.1)$$

Burada t , *tweete* karşılık gelen vektörü, d_i Vikipedi dokümanlarına verilen numarayı, s_i ise Solr’da yapılan arama sonucunda dokümana verilen *alakalılık* skorunu ifade etmektedir. Her bir vektör, indekslenmiş Vikipedi dokümanı sayısı büyüklüğünde boyuta sahiptir. Bu çalışmada 226460 doküman indekslenmiştir.

Bir kullanıcının analiz edilerek vektörel forma dönüşmesi, attığı *tweetler* arasından seçilen 1000 tanesinin vektörel ortalamasının alınması şeklinde sağlanmaktadır:

$$u = \frac{t_1 + t_2 + \dots + t_{1000}}{1000} \quad (3.2)$$

Bu denklemde u kullanıcıyı ifade edecek olan vektör, t_i 'ler ise kullanıcının attığı her bir *tweet*ten 3.1 numaralı denklemdeki gibi elde edilen vektörlerdir. 1000 sayısı, tarafımızca keyfi olarak seçilmiş bir sayıdır. Twitter kullanıcısı analiz edilirken ele alınacak olan *tweet*ler, bu kullanıcı tarafından en son atılanlardan seçilmiştir. Kullanıcı tarafından atılan bütün *tweet*ler arasından rasgele seçilmesi de mümkündür.

Kullanıcılar ve *tweet*ler vektörlere dönüştürüldükten sonra, aralarında bir benzerlik hesaplamak mümkündür. u ve t vektörleri aynı boyuttadır. Bu çalışmada uzaklık ölçüsü olarak, metin dokümanlarının uzaklık/benzerlik hesaplamalarında sıkça kullanılan kosinüs uzaklığı yöntemi tercih edilmiştir. Bu yöntemle göre uzaklık hesabı aşağıdaki gibi vektörlerin iç çarpımının her iki vektörün normları çarpımına oranı ile hesaplanır. Uzaklık fonksiyonu aşağıda verilmiştir:

$$dist(u, t) = \frac{\vec{u} \cdot \vec{t}}{|\vec{u}| \cdot |\vec{t}|} \quad (3.3)$$

Uzaklık değerinden benzerlik/yakınlık hesaplamak bir diğer meseledir. Kosinüs uzaklığı 0 ile 1 arasında değişen bir değer olduğu için, benzerlik hesabı, 1'den uzaklık değerinin çıkarılması ile oldukça basit bir şekile yapılabilir:

$$sim(u, t) = 1 - dist(u, t) \quad (3.4)$$

Kullanıcılar ve *tweet*leri temsil eden vektörler arasındaki benzerliğin 3.4 numaralı denklemde verilen fonksiyona göre hesaplanmasıyla, bir *tweet*in bir kullanıcı ile ne kadar *alakalı* olduğu sorusuna rakamsal bir cevap verilmiş olur. Buna göre, kullanıcının Twitter ana sayfasında, zaman yakınlığına göre gösterilen n adet *tweet*, özetlediğimiz yöntemle yeniden sıralanarak daha alakalı *tweet*ler daha üst sırada çıkacak şekilde gösterilebilir. Örneğin, $n = 5$ için, normalde $[t_1, t_2, t_3, t_4, t_5]$ sırasıyla gösterilen *tweet*ler, bu uygulama sonucunda $[t_2, t_5, t_3, t_1, t_4]$ sırasıyla gösterilebilir. Bu çalışmanın, kullanıcıların daha çok ilgi duyacağı bir zaman tüneli gösterimi ortaya çıkaracağı değerlendirilmektedir

4.4 Elde Edilen Sonular

Twitter kullanıcılarının bir kısmı yazdığı iletileri açık bir şekilde paylaşırken diğler bir kısmı ise bu iletileri koruma altına alarak serbest erişime izin vermemektedir. Kullanıcıların gizlilik tercihi gereğı gizlenen *tweet*ler, onları takip etmeyen kimselerce görülemeyeceğı gibi, Twitter'dan veri çekilen kütüphaneler üzerinden de erişilememektedir. Bu sebeplerden ötürü, bu uygulamanın pratikteki tam sonuçlarının değerlendirilmesi mümkün olmamakla birlikte, erişilebilen *tweet*ler üzerinden bir değerlendirme yapılmıştır. Seçilen 20 kullanıcı için, kendi zaman tünellerindeki en yakın 30 *tweet*in, farklı bir şekilde sıralandığı ifade edilerek bu işlem sonucu daha iyi bir sıralama elde edilip edilmediğı sorulmuştur. Bu çalışmanın gizlenmiş *tweet*leri kapsamadığı katılımcılara belirtilmiştir. Katılımcıların bu yeniden sıralama işlemine dair olumlu / olumsuz / nötr şeklindeki kanaatleri alınmıştır. Bu 20 kullanıcı arasında verilen reylerin dağılımı Çizelge 4.1'de gösterilmiştir.

Çizelge 4.1 : Twitter kullanıcılarının yeniden sıralamaya dair değerlendirmesi

Olumlu	Nötr	Olumsuz
14	5	1

5. 20-NEWSGROUPS DOKÜMANLARININ MERKEZİ ELEMANLAR REFERANS ALINARAK SPEKTRAL ALGORİTMA İLE ÖBEKLENMESİ

Çalışmanın bu bölümünde, spektral öbeleme algoritmasının metin verileri üzerindeki etkileri incelenmiştir. Dokümanların vektörlere dönüştürülmesinde ise yine referans dokümanların kullanılması araştırılarak kullanılmıştır. 4. bölümde anlatılan uygulamada, veri kümesi Twitter iletileri, referans vektör olarak kullanılan dokümanlar ise Vikipedi sayfalarıydı. Bu çalışmada veri kümesi olarak bölüm 3.1.3'te anlatılan 20 farklı başlıktan İngilizce haber grubu metinleri kullanılmış, referans dokümanlar ise bu veri kümesinin içindeki merkezi elemanlardan seçilmiştir. Uygulamada yapılan deneysel çalışmalar kapsamında öncelikle k-means ve spektral öbeleme algoritmaları dokümanların TF-IDF vektörlerine çevrilmiş haline uygulanmış, sonrasında ise veri kümesi, merkezi dokümanlar referans alınarak yeni bir vektörel uzaya dönüştürülmüş ve bu dönüşümden sonra öbeleme algoritmaları çalıştırılmıştır. Sonuçlar, uyarlanmış rand indeks (Adjusted Rand Index) ve uyarlanmış karşılıklı bilgi (Adjusted Mutual Information) ölçütleri kullanılarak değerlendirilmiştir.

5.1 Spektral Öbeleme

Spektral öbeleme algoritması, karmaşık veri kümelerinin öbelemesinde kullanılan etkili bir algoritmadır. Algoritma sadece veri örnekleri arasındaki ikili yakınlık / uzaklık ilişkilerini dikkate almaktadır. Diğer birçok öbeleme algoritması bir istatistiksel modeli temel alarak veri kümesinin vektörel forma dönüştürüldüğünde bu modele uyum sağlayacağını varsayar. Zaman zaman veri setleri bu modellerin dışına çıkabilmekte ve bu tür varsayımlar algoritmaları saptırarak öbeleme sonuçlarına olumsuz katkıda bulunabilmektedir. Sadece veri örneklerinin göz önünde bulundurulması öbeleme yapılması, bu riski engelleyici bir yöntemdir.

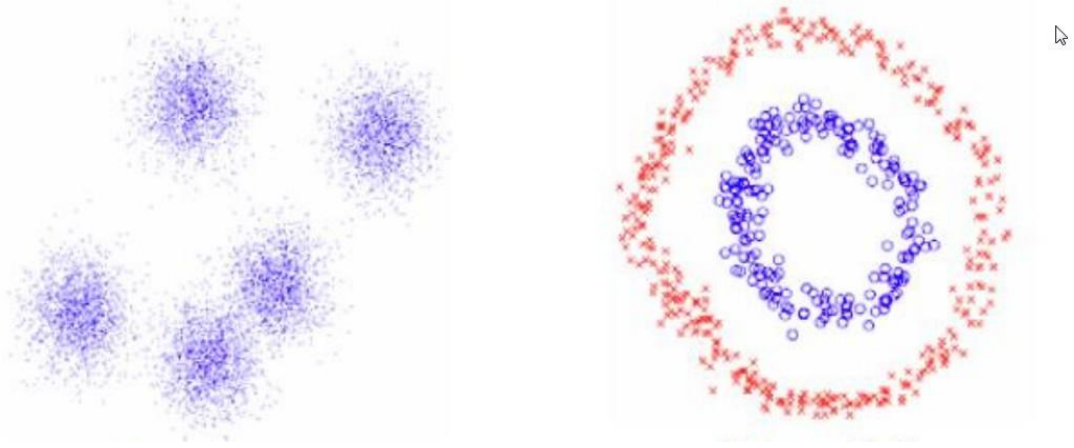
Algoritma, veri örneklerinin ikili yakınlık değerlerinden oluşan bir yakınlık matrisini girdi olarak kabul eder ve bu matrisin Laplace formunun özvektörleri üzerinden işlem yapar. Aslında spektral öbeleme algoritması, NP-zor karmaşıklığa sahip çizge

parçalama probleminin *yaklaşık* bir çözüdür. Laplace matrisinin özvektörleri, veri setinin en uygun kesimine (optimal cut) dair tahminde bulunmak için kullanılmaktadır. Bir kare matrisin Laplace matrisinin nasıl oluşturulacağı konusunda birkaç farklı yaklaşım mevcuttur. Bu çalışmada kullanılan yöntem, normalleştirilmiş Laplace matrisidir. Algoritmanın nasıl çalıştığını gösteren kod akışı, Şekil 5.1’de gösterilmiştir.

```
procedure SC(X,k)
   $A^{n \times n} \leftarrow X$ 'in benzerlik matrisi
   $D^{n \times n} \leftarrow A$ 'nın derece matrisi (degree matrix)
   $L^{n \times n} \leftarrow D^{-1/2} \cdot A \cdot D^{-1/2}$ 
   $E^{n \times k} \leftarrow L$ 'nin en küçük özvektörleri
   $Y^{n \times k} \leftarrow E$ 'nin normalize edilmiş hali
  Y'yi k-means ile k adet öbeğe ayır
  X'in her bir elemanını Y için bulunan öbeğe ata
```

Şekil 5.1 : Spektral Öbekleme Algoritması.

Spektral öbeklemede önemli olan veri sınıflarının / öbeklerinin kendi içinde bağlı yapılar halinde bulunmasıdır. Bağlı yapılar halinde bulunan fakat lineer olarak ayrılması zor olan veri kümelerini öbeklemek için başvurulabilecek bir algoritmadır. Şekil 5.2’de, içindeki öbeklerin karakteristiği bakımından farklı iki veri kümesi görselleştirilmiştir. Soldaki veri kümesindeki öbekler oldukça düzenli, konveks şekillerde ve kendi içlerinde sıkışık bir şekilde görülmektedir. Bu gibi veri kümeleri, k-means gibi yöntemler ile rahatlıkla öbeklenebilir. Sağdaki veri kümesinde ise, öbekler iç içe geçmiş iki çemberden oluşmaktadır. Aynı öbek içerisinde birbirine çok uzak, ama zincirleme yakın yollarla geçişliliğin sağlanabildiği elemanlar mevcuttur. Veri öbeklerinin bu şekilde dağıldığı durumlarda, spektral öbekleme yöntemi daha başarılı sonuçlar ortaya çıkartmaktadır.



Şekil 5.2 : İki farklı veri dağılımı

5.2 Merkezi Elemanların Referans Olarak Kullanılması

Metin dokümanlarının öğrenme algoritmalarıyla sınıflandırılması ya da öbeklenmesinde kullanılan nümerik vektörlerin oldukça büyük boyutlara sahip olması, önemli problemlere yol açabilmektedir. Bunlardan bir tanesi bellek ve işlemci ihtiyacının fazla olmasıdır. Bir diğeri ise yüksek veri boyutunda elde edilen sonuçların sapsmalara daha müsait olması ve genelleştirmenin daha zor olmasıdır. Bu bağlamda PCA, LDA gibi çeşitli boyut azaltma yöntemleri mevcuttur. Bu çalışma kapsamında kullanılacak olan boyut azaltma yöntemi, veri içerisindeki merkezi elemanlar kullanılarak yeni boyutlar belirlenmesine dayanmaktadır. Burada veri kümesi içerisindeki küçük lokal yoğunlaşmaların bir bilgi taşıdığı, bir özellik ifade ettiği düşünülerek bu lokal bilgi yoğunlaşmalarının, özellikler kümesi olarak kullanılması amaçlanmaktadır. Veri kümesinden rasgele seçilen elemanlara olan yakınlıklar kullanılarak boyut azaltma yöntemleri geçmişte uygulanmış ve sonuçların kabul edilebilir düzeyde olduğu, öte yandan PCA gibi yöntemlerden daha hızlı çalıştığına yönelik araştırmalar mevcuttur (Bingham ve Mannila, 2001). Bu çalışmada ise, yeni boyutları belirlemek için önce veri kümesindeki belirlenen sayıda merkezi elemanlar tespit edilmektedir.

Çözülmesi gereken problem, dokümanları k adet öbeğe ayırmaktır. Öncelikle, veri kümesi n adet doküman vektöründen oluşan bir matristir:

$$X^{n \times r} = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^n \end{bmatrix} \quad (5.1)$$

$$x^i = [x_1^i \ x_2^i \ \dots \ x_r^i] \quad (5.2)$$

Burada n veri sayısını, r ise TF-IDF vektörlerindeki boyut sayısını ifade etmektedir. Veri setindeki m adet merkezi eleman, k-medoids algoritması ile bulunur:

$$C^{m \times r} = \begin{bmatrix} c^1 \\ c^2 \\ \vdots \\ c^m \end{bmatrix} \quad (5.3)$$

Burada her bir c^i vektörü, k-medoids algoritması ile bulunan bir merkezi elemandır. K-medoids ile bulunacak merkezi vektör sayısı, 5.4 numaralı eşitsizlikteki gibi bir varsayım yapılarak tarafımızca keyfi olarak seçilmiştir.

$$k < m \ll r \quad (5.4)$$

Merkezi elemanlar bulunduktan sonra, TF-IDF vektörlerinden oluşan ve X matrisi ile ifade edilen veri setinin boyutu bu merkezi elemanlar kullanılarak azaltılarak P matrisi oluşturulur. P matrisi, X 'in boyutu indirgenmiş halidir:

$$\begin{bmatrix} x_1^1 & \dots & x_r^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \dots & x_r^n \end{bmatrix} \rightarrow \begin{bmatrix} p_1^1 & \dots & p_m^1 \\ \vdots & \ddots & \vdots \\ p_1^n & \dots & p_m^n \end{bmatrix} \quad (5.5)$$

Başlangıçta r boyutlu olan x^i vektörünün, m adet c^j vektörünün her biriyle arasındaki yakınlık bulunur ve bu m adet yakınlık değeri, artık x^i vektörünün boyutları azaltılmış hali olan p^i vektörüdür. Bu vektör, 3.4 numaralı eşitlikte belirtilen yakınlık fonksiyonu ile elde edilir:

$$p_j^i = sim(x^i, c^j) \quad (5.6)$$

5.3 Uygulamalar

İngilizce haber grupları veri kümesi, Apache Mahout kullanılarak 3 farklı öbikleme yöntemi ile gruplara ayrılmıştır. Bu üç yöntem aşağıda açıklanmıştır:

- **k-means:** TF-IDF vektörleri üzerinde bilindik k-means algoritması çalıştırılmıştır. Bu algoritmayı çalıştıran mahout komutu aşağıdaki gibidir:

```
bin/mahout kmeans \  
-i <girdi vektörlerinin bulunduğu dizin> \  
-o <çıktıların yazılacağı dizin> \  
-k <bulunacak öbek sayısı> \  
-dm <uzaklık ölçüsü> \  
-x <maksimum iterasyon sayısı>
```

- **düz spektral:** X matrisinin satırları olan TF-IDF vektörleri (x^i vektörleri) arasındaki benzerlikler üzerinden benzerlik matrisi (A) hesaplanarak, Bu matris üzerinde spektral öbikleme algoritması koşulmuştur.

$$A^{n \times n} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \quad (5.7)$$

$$a_{ij} = \text{sim}(x^i, x^j) \quad (5.8)$$

Bu algoritmayı çalıştıran mahout komutu aşağıdaki gibidir:

```
bin/mahout spectralkmeans \  
-i <benzerlik matrisinin bulunduğu dizin> \  
-o <çıktıların yazılacağı dizin> \  
-d <veri örneği sayısı> \  
-k <bulunacak öbek sayısı> \  
-x <maksimum k-means iterasyon sayısı>
```

- **gelişmiş spektral:** -medoids algoritması ile bulunmuş olan merkezi elemanlar kullanılarak 5.1 numaralı eşitlikteki X matrisi, öncelikle 5.4 numaralı dönüşüm uygulanarak m kolonlu P matrisine dönüştürülmüş ve benzerlik (A) matrisi, P 'nin satırları arasındaki yakınlıklar hesaplanarak oluşturulmuştur. Spektral öbekleme algoritmasına bu şekilde oluşturulan yakınlık matrisi girdi verilerek koşulmuştur.

$$A^{n \times n} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \quad (5.9)$$

$$a_{ij} = sim(p^i, p^j) \quad (5.10)$$

Bu algoritmayı çalıştıran mahout komutu, düz spektral öbekleme algoritmasındaki ile aynıdır. Fark eden yalnızca benzerlik matrisinin içeriğidir.

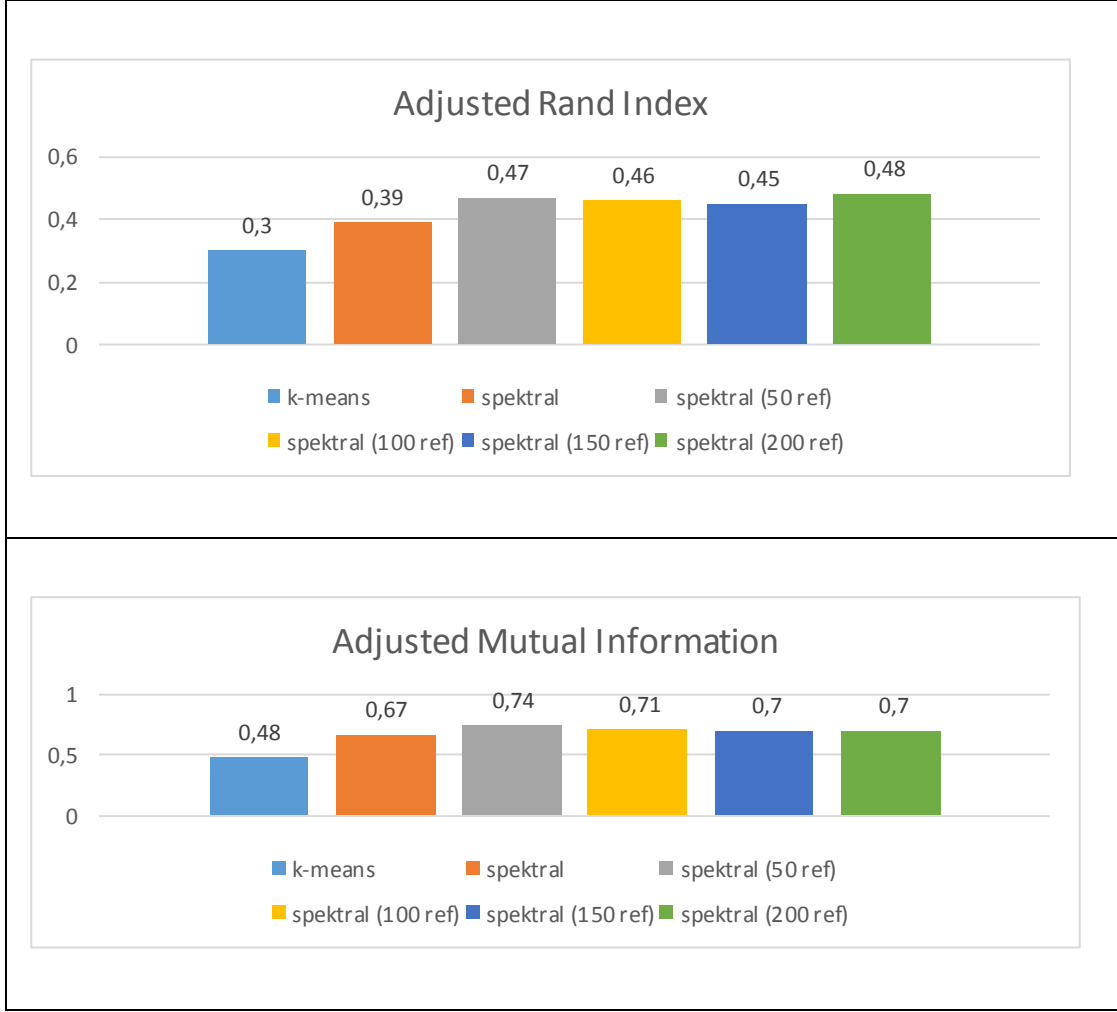
5.4 Elde Edilen Sonuçlar

Bahsedilen üç öbekleme yöntemi, 20 farklı haber grubundan dokümanlara sahip veri kümesine uygulandıktan sonra elde edilen öbeklerin doğruluğu test edilmiştir. Üçüncü öbekleme yöntemi için gerekli m değeri, 5.4 numaralı eşitsizlikteki varsayım dahilinde farklı farklı değerlerden seçilerek gelişmiş spektral öbekleme uygulamasına parametre olarak verilmiştir.

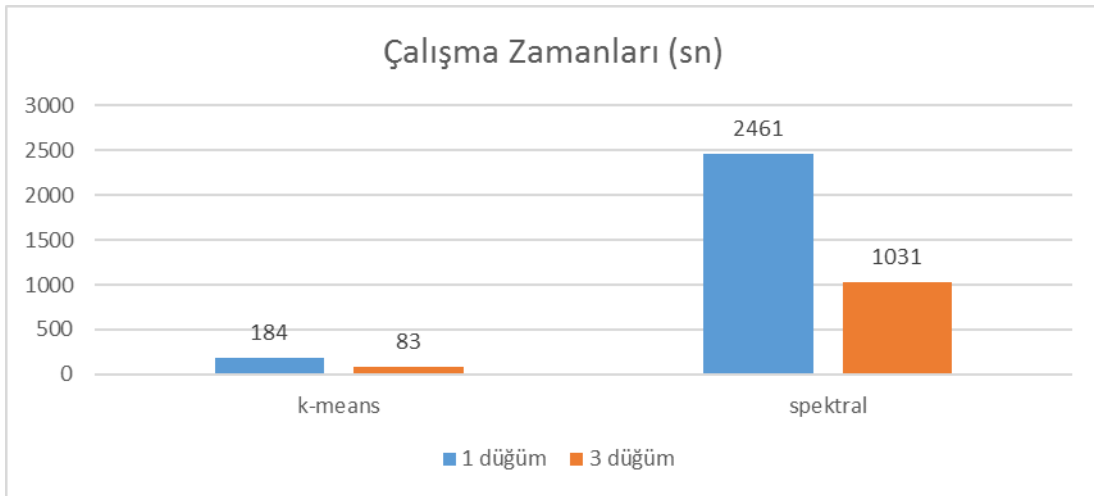
Öbekleme sonuçlarının testinde uyarlanmış rand indeks ve uyarlanmış karşılıklı bilgi ölçütleri kullanılmıştır ve bu iki öbekleme kalitesi ölçütü için iki ayrı grafik çizilmiştir. Elde edilen sonuçlar Şekil 5.3'teki grafiklerde gösterilmiştir. Grafiklerdeki her bir sütun, birer öbekleme yöntemini temsil etmektedir. İlk iki sütun k-means ve düz spektral yöntemleri için bulunan sonuçlardır. Gelişmiş spektral algoritma yöntemi 4 farklı m parametresi (50, 100, 150, 200) ile ayrı ayrı çalıştırılarak sonuçları ölçüldüğü için, grafiklerde bu yöntemle elde edilen sonuçlara dair 4 sütun mevcuttur. Grafiklerde sütunlara verilen nümerik değerler ise bu ölçütlere göre öbekleme yönteminin başarısını ifade etmektedir. Her bir yöntem için grafikte yazan skor, veri setinin her defasında karıştırılarak, söz konusu öbekleme yönteminin 10 defa arka arkaya çalıştırılmasıyla elde edilen ortalama skordur.

Grafiklerde görülen sonuçlar göstermektedir ki, spektral öbikleme algoritması, doküman öbiklemede k-means algoritmasından daha iyi bir yöntemdir. K-means ile bulunan öbeklerde uyarlanmış rand indeks değeri 0,3 ve uyarlanmış karşılıklı bilgi değeri 0,48iken, spektral öbikleme algoritması ile bulunan öbekler neticesinde hesaplanan skorlar sırasıyla 0,39 ve 0,67'dir. Buna ek olarak, merkezi elemanlar referans olarak kullanıp veri kümesi yeni bir boyut uzayına aktarıldığında, diğer bir deyişle bu tez kapsamında önerilen iyileştirici yöntem uygulandığında, spektral öbikleme algoritmasının ortaya çıkardığı sonuçlar daha da iyileşmektedir. Farklı sayıda referans doküman sayısı (m) için önerilen gelişmiş spektral öbelleme uygulamalarından ortaya çıkan uyarlanmış rand indeks skorları 0,45-0,50 değerleri arasında, uyarlanmış karşılıklı bilgi skorları ise 0,70-0,75 değerleri arasında değişmektedir. Dolayısı ile yapılan bu fazladan ön işleme, spektral öbikleme algoritmasının daha kaliteli öbekler ortaya çıkarmasını sağlamaktadır.

Diğer yandan, Mahout tarafından sağlanan k-means ve spektral öbikleme algoritmalarının dağıtık ortamda nasıl ölçeklendiği de önemli bir ölçüttür. Zira, hazırlanan çözümler, dağıtık mimaride çalışmaya uygun ve büyük veri kümeleri için çalıştırılması hedeflenerek geliştirilmiştir. Dolayısıyla kullanılan araçların genişlemeye karşı verdiği tepki bu çözümlerin gerçekten büyüyen veri kümeleri için kullanılabilir olmasında etkilidir. Şekil 5.4'te k-means ve spektral öbikleme algoritmalarının, tek düğümlü ve 3 düğümlü olarak konumlandırılmış Hadoop kullanıldığında elde edilen çalışma zamanları gösterilmiştir. Zaman değerleri saniye cinsindedir. Bu grafiklere göre, aynı veri seti 3 düğümlü yapıda çalıştırıldığında öbikleme algoritmalarının çalışma süreleri 1 düğümlü yapıya göre yarıdan daha az, 1/3'ten dahafazla bir zamanda sonlanmaktadır. Bu veriler ışığında algoritmaların tam olarak lineer ölçeklenebilir olduğu söylenemez ancak kabul edilebilir bir ölçeklenebilirlik söz konusudur.



Şekil 5.3 : Farklı m değerleri için öbikleme sonuçları.



Şekil 5.4 : Algoritmaların 1 ve 3 düğümde çalışma zamanları

6. SONUÇ VE ÖNERİLER

Üretilen dijital verilerin analiz edilerek değerlendirilmesi, birçok alanda önemli faydalar sağlamaktadır. Metin verilerinin otomatik kategorizasyonu da bu alanlardan biridir. Öğrenme algoritmalarıyla metin dokümanlarının otomatik öbeklenebilmesinin önemli bir faydası, otomatik tasnif sağlayarak farkındalık oluşturulmasıdır. Farkındalık ise, siber güvenliğin önemli bir unsurudur.

Doküman öbeklemede problem olarak görülen meselelerden birkaç tanesi ele alınarak bu problemler için sunulan alışlageldik yöntemler dışında öneriler, bu çalışmada sunulmuştur. İlk olarak metin dokümanlarının nümerik vektörlere çevrilmesinde, üslup ve içerik açısından referans olabilecek dokümanların kullanılması önerilmiştir. Bu öneriye binaen, 4. bölümde sunulan uygulamada Vikipedi dokümanları, 5. bölümde sunulan uygulamada ise veri kümesinin içindeki merkezi elemanlar referans olarak kullanılarak dokümanlar vektörel forma dönüştürülmüştür. İkinci olarak, metin veri kümelerinde birbirinden ayrı öbeklerin iç içe geçmiş formda ya da muhtelif karmaşık konkav biçimlerde bulunmasının muhtemel bir durum olması noktasında, spektral öbekleme yönteminin isabet oranı daha yüksek öbekler ortaya çıkaracağı ifade edilmiş ve önerilen bu yöntem 5. bölümdeki uygulamalarda kullanılmıştır. Her iki uygulamada da önerilen yöntemlerin iyileştirici sonuç verdiği, yapılan değerlendirmeler neticesinde ortaya konulmuştur.

Büyük veri kavramının oldukça yaygınlaştığı ve büyük ölçekli olarak anılan veri kümelerinin önemli bir kısmını metin verilerinin oluşturduğu günümüzde, bu tarz veri kümeleri üzerinde çalışan algoritmalarda, ölçeklenebilirlik önemli bir kıstastır. Bu bağlamda, geliştirilen uygulamalarda dağıtık mimaride çalışan veri saklama ve işleme araçları kullanılmıştır. Yeterli kaynaklar sağlandığı sürece büyük veri kümeleri üzerinde çalışmaya uygun, yatay olarak genişleyebilir çözümler sunulmuştur.

Sunulan yöntemlerin daha da gelişmesi, daha da iyi sonuç vermesi adına gelecekte yapılması muhtemel birtakım iyileştirmeler öngörülmektedir. Bunlardan ilki, 5.

bölümde kullanılan merkezi elemanların tespiti yönteminin, 4. bölümde bahsedilen Vikipedi dokümanlarına da uygulanmasıdır. Daha az sayıda ve öz bilgi taşıyan Vikipedi dokümanları, karmaşıklığı düşürebilir, performansı ise artırması muhtemeldir. Dolayısıyla referans olarak kullanılması daha verimli olabilecektir. Diğer bir iyileştirme ise, 5. bölümdeki spektral öbekleme uygulamasında, benzerlik matrisinin hesaplanması noktasında mümkündür. Burada da veri kümesini temsil eden çizge, zayıf bağlar gözardı edilerek daha seyrek bir halde oluşturulabilir. Uygulamanın mevcut halinde, her bir doküman, diğer bütün dokümanlara zayıf ya da güçlü bir bağla bağlı kabul edilmekte ve algoritma bu tamamen bağlı çizgeden elde edilen benzerlik matrisi ile çalışmaktadır. Bu matristeki zayıf bağların (nümerik değeri düşük yakınlıkların), en yakın n komşu yöntemi ile ya da belli bir eşik yakınlık değeri uygulanarak gözardı edilmesi, dolayısıyla yakınlık matrisinin seyreltilmesi yine karmaşıklığı azaltarak performansı artırabilecek bir yaklaşımdır. Bu ön işleme yöntemlerinin, sunulan metotlara eklenerek gelecekte daha da iyi sonuçlar alınabileceği değerlendirilmektedir.

KAYNAKLAR

- AbdelRahman O. H. ve Gelenbe, E.** (2014). Search in Big Networks and Big Data. Springer Proceedings in Mathematics & Statistics Analytic Methods in Interdisciplinary Applications, 1-15. doi:10.1007/978-3-319-12148-2_1
- Aggarwal, C. C., ve Zhai, C.** (2012). A survey of text clustering algorithms. Mining text data (Sf. 77-128). Springer US.
- Alpaydin, E.** (2014). Introduction to Machine Learning. Cambridge, MA: MIT Press. ISBN: 978-0-262-02818-9
- Bach, F. R. ve Jordan, M. I.** (2006). Learning Spectral Clustering, With Application to Speech Separation. Journal of Machine Learning Research, 7, 1963-2001.
- Bao, F., Tang, S., Li, J., Zhang, Y. ve Ye, W.** (2008). Document Clustering Based on Spectral Clustering and Non-negative Matrix. 21st International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Wroclaw, Polonya, 18-20 Haziran
- Bingham, E. ve Mannila, H.** (2001). Random projection in dimensionality reduction: applications to image and text data. 7. ACM SIGKDD international conference on Knowledge discovery and data mining (sf. 245-250). ACM.
- Cai, D., He, X., ve Han, J.** (2005). Document clustering using locality preserving indexing. IEEE Transactions on Knowledge and Data Engineering, Cilt 17, Sayı 12, Sf. 1624-1637.
- Berry, M. W.** (2003). Survey of Text Mining: Clustering, Classification, and Retrieval. Springer, New York.
- Datta, S., Bhaduri, K., Giannella, C., Wolff, R. ve Kargupta, H.** (2006). Distributed data mining in peer-to-peer networks. IEEE Internet Computing, Cilt 10 , sayı 4, Sf. 18-26.
- Dhillon, Inderjit S.** (2001), Co-clustering documents and words using BipartiteSpectral Graph Partitioning, Conference on Knowledge Discovery and Data Mining (KDD '01), San Francisco, California, ABD
- Dean, J. ve Ghemawat, S.** (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), sf. 107-113.
- Huang, Anna** (2008), Similarity Measures for Text Document Clustering, Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, Yeni Zelanda, Sf. 49-56

- Ienco, D. ve Meo, R.** (2008), Exploration and Reduction of the Feature Space by Hierarchical Clustering. SIAM Conference on Data Mining (SDM08), Atlanta, Georgia, ABD, 24-26 Nisan.
- Judith, J. E., ve Jayakumari, J.** (2015). Distributed document clustering algorithms: a recent survey. International Journal of Enterprise Network Management, Cilt 6, Sayı. 3, Sf. 207-221.
- Li, Q., Wang, P., Wang, W., Hu, H., Li, Z. ve Li, J.** (2014). An efficient K-means clustering algorithm on MapReduce. International Conference on Database Systems for Advanced Applications (Sf. 357-371). Springer International Publishing.
- Lu, C., Lam, W., ve Zhang, Y.** (2012). Twitter user modeling and tweets recommendation based on wikipedia concept graph. Twenty-Sixth AAAI Conference on Artificial Intelligence.
- Ng, A. Y., Jordan, M. I. & Weiss, Y.** (2001). On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems 14, Sf. 849-856, MIT Press.
- Reed, J. W., Jiao, Y., Potok, T. E., Klump, B. A., Elmore, M. T., ve Hurson, A. R.** (2006). TF-ICF: A new term weighting scheme for clustering dynamic data streams. 5th International Conference on Machine Learning and Applications (ICMLA'06) (Sf. 258-263). IEEE.
- Ribeiro, Marcelo N. and Neto, Manoel J. R. ve Prudencio, Ricardo B. C.** (2008), Local Feature Selection in Text Clustering. 15th International Conference on Neural Information Processing (ICONIP) Auckland, Yeni Zelanda, 25-28 Kasım
- Sathiyakumari, K., Manimekalai, G., Preamsudha, V. ve Scholar, M. P.** (2011), A Survey on Various Approaches in Document Clustering, International Journal of Computer Technology and Applications (IJCTA), Cilt 2, Sf. 1534-1539.
- Steinbach, M., Karypis, G. ve Kumar, V.** (2000). A comparison of document clustering techniques. KDD workshop on text mining (Cilt 400, Sayı. 1, Sf. 525-526).
- Svadas, T. ve Jha, J.** (2014), A Literature Survey on Text Document Clustering and Ontology based Techniques, International Journal of Innovative and Emerging Research in Engineering (IJIERE), Cilt 1 Sayı 2, Sf. 8-11.
- Thangamani, M. ve Thangaraj, N.** (2010), Survey on Text Document Clustering, International Journal of Computer Science and Information Security (IJCSIS), Cilt 8, sayı. 4, Sf. 174-178.
- Vempala, S. ve Wang, G.** (2005), On the Benefit of Spectral Projection for Document Clustering, SIAM Conference on Data Mining (SDM'05), California, ABD

- Xu, J. and Xu, B. and Zhang, W. and Cui, Z. ve Zhang, W.** (2007), A New Feature Selection Method for Text Clustering, Wuhan University Journal of Natural Sciences, Cilt 12, Sayı 5, Sf. 912-916
- Yan, D., Huang, L., ve Jordan, M. I.** (2009). Fast approximate spectral clustering. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (Sf. 907-916). ACM.
- Grimes, S.** (2007). A brief history of text Analytics - BeyeNETWORK. alındığı tarih: 07.07. 2016, adres: <http://www.b-eye-network.com/view/6311>
- Google trendler** – Tüm Kategoriler Liste Başlı Grafikleri (2014). Alındığı tarih: 01.07.2016, adres: <https://www.google.com/trends/topcharts#vm=cat&geo=TR&date=2014&cid>

ÖZGEÇMİŞ

Ad Soyad: Evren Pala
Doğum Yeri ve Tarihi: Eskişehir – 1989
E-Posta: palaevren@gmail.com
Lisans: ODTÜ Bilgisayar Mühendisliği

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- Pala, E., ve Yılmaz, G. (2016). Effects of Spectral Clustering on Document Categorization Using Distributed Tools. 3rd International Conference on Advanced Technology & Sciences, 1-3 Eylül, 2016 Konya, Turkey.