MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

METİN TABANLI TÜRKÇE İÇERİKLERİN ÇİZGE VERİ YAPISIYLA TEMSİL EDİLMESİ

TARIK ŞAHİN

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği
Anabilim Dalı
Bilgisayar Mühendisliği Programı

DANIŞMAN
Dr. Öğr. Üyesi Önder DEMİR

EŞ-DANIŞMAN
Dr. Öğr. Üyesi Kazım YILDIZ

İSTANBUL, 2019

MARMARA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

METİN TABANLI TÜRKÇE İÇERİKLERİN ÇİZGE VERİ YAPISIYLA TEMSİL EDİLMESİ

TARIK ŞAHİN

(523616016)

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği

Anabilim Dalı

Bilgisayar Mühendisliği Programı

DANIŞMAN

Dr. Öğr. Üyesi Önder DEMİR

EŞ-DANIŞMAN

Dr. Öğr. Üyesi Kazım YILDIZ

İSTANBUL, 2019

MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Marmara Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Öğrencisi Tarık ŞAHİN'in "**Metin Tabanlı Türkçe İçeriklerin Çizge Veri Yapısıyla Temsil Edilmesi**" başlıklı tez çalışması, 26 Ağustos 2019 tarihinde savunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Dr. Öğr. Üyesi Önder DEMİR (Danışman)
Marmara Üniversitesi Bilgisayar Mühendisliği Bölümü .....

Doç. Dr. Ali BULDU (Üye)
Marmara Üniversitesi Bilgisayar Mühendisliği Bölümü .....

Dr. Öğr. Üyesi Mustafa Cem KASAPBAŞI (Üye)
İstanbul Ticaret Üniversitesi Bilgisayar Mühendisliği Bölümü .....

ONAY

Marmara Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 04.09.2019 tarih ve 2019/18-03 sayılı kararı ile Tarık ŞAHİN'in Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programında Yüksek Lisans derecesi alması onanmıştır.

Fen Bilimleri Enstitüsü Müdürü
Prof. Dr. Bülent EKİCİ

ÖNSÖZ

Tez çalışmam boyunca benden bilgisini, desteğini ve sabrını esirgemeyen hocalarım Dr. Öğr. Üyesi Önder Demir'e ve Dr. Öğr. Üyesi Kazım Yıldız'a teşekkürü borç bilirim.

Haziran, 2019 **Tarık ŞAHİN**

İÇİNDEKİLER

ÖZET

METİN TABANLI TÜRKÇE İÇERİKLERİN ÇİZGE VERİ YAPISIYLA TEMSİL EDİLMESİ

Günümüzde gelişen internet teknolojileriyle birlikte artık insanlar daha fazla içerik üretmeye başladılar. Özellikle sosyal medya gibi araçların ve mobil cihazlardan internete erişimin yaygınlaşmasıyla üretilen içerikler devasa boyutlara ulaşmıştır. Ayrıca günümüzde iş dünyası da artık neredeyse tamamen dijitalleşmiş ve şirketlerin tamamına yakını verilerini elektronik ortamda saklar hale gelmiştir. Bunun gibi daha bir çok veri kaynağı örneği verilebilir.

Veriler bu denli arttıkça, insanlar bu verilerden anlamlı bilgiler elde etme arayışına girmişlerdir. Ham olarak duran verilerden çeşitli istatistiksel veya mantıksal çıkarımlar elde etmek insan hayatını kolaylaştırmaktadır. Bu kadar devasa verilerin olduğu sistemleri analiz etmek, hatta birbirinden bağımsız sistemlerden elde edilecek verileri entegre ederek analiz etmek üzerinde yoğun olarak çalışılan bir konu haline gelmiştir.

Bu çalışmada hızlı ve doğru analiz yapabilmek için metin tabanlı verilerin Neo4J veritabanında çizge yapısında saklanması sağlanmıştır. Birbirinden bağımsız farklı veri kaynaklarından elde edilen veriler tek bir veritabanında birbiriyle entegre bir şekilde saklanmıştır. İki farklı uygulama geliştirilmiş ve bu iki uygulama üzerinden yapılan metin tabanlı içerik analizlerinin sonuçları değerlendirilmiştir.

ABSTRACT

REPRESENTATION OF TEXT BASED TURKISH CONTENTS WITH GRAPH DATA STRUCTURE

As a result of development of internet technologies, poeple produce more contents. Especially because of development of tools like social media and growing up access to internet from mobile devices, these contents has reached enormous dimensions. In addition, nowadays, the business world is almost completely digitalized and almost all companies have stored their data electronically. Data sources examples like these can be increased.

As data increased so much, people sought to derive meaningful inferences from these data. Obtaining various statistical or logical inferences from raw data makes human life easier. Analyzing the systems with such enormous data and even analyzing them by integrating the data to be obtained from independent systems has become an area of intense work.

In this study, it is provided to store text based data in graph structure in Neo4J database for fast and accurate analysis. Data from different independent data sources are stored in a single database. Two different applications were developed and the results of the text-based content analyzes were evaluated.

KISALTMALAR

JVM : Java Virtual Machine

HL : High Level

DDİ : Doğal Dil İşleme

DOM : Document Object Model

HTTP : Hypertext Transfer Protocol

RSS : Rich Site Summary

KNN : K-Nearest Neighbor

XML : Extensible Markup Language

ŞEKİL LİSTESİ

SAYFA

1. GİRİŞ

Günümüzde hem internetin yaygın hale gelmesi hem de internete bağlanabilen cihazların artmasıyla, dijital ortamda toplanan veri miktarlarında çok önemli artışlar olmuştur. 2017 yılı haziran ayında 3,885,567,617 internet kullanıcısı tespit edilmiştir.Her dakika 3.8 milyon Google araması gerçekleşmektedir[1]. Bunun gibi birçok örnek ve daha detaylı istatistikler toplanabilir.

Veri miktarı bu denli büyük olunca, verileri işleyebilmek ve bunlardan anlamlı sonuçlar çıkarabilmek de önemli hale gelmiştir. Big data yani büyük veri denilen kavram yaygınlaşmaya başlamıştır. Facebook, Twitter gibi sosyal medya araçlarının da yaygınlaşmasıyla, özellikle metin madenciliği gibi büyük veri teknolojilerinden faydalanan konular daha çok gündeme gelmeye başlamıştır.

Metinlerin makineler tarafından işlenebilmesi insanlar açısından önemlidir. Bilgisayarların insanların konuştuğu dili anlayabilmesi ve yorum yapabilmesi insanlık için büyük kolaylıklar ve hizmetler sunmaktadır. Akıllı telefonlarda kullanılan kişisel asistan programları buna güzel bir örnektir. Google tarafından geliştirilen Google Asistan ürünü bu uygulamalardan biridir[2].

Verilerin istatistiksel olarak işlenebilmesi veya konuşma robotları gibi anlık reaksiyon gösteren programların gelişmesi birbirine bağlı konulardır. Büyük veri kümeleri şeklinde bulunan ham veriler işlenerek anlamlı hale getirilebilir, bu anlamlı verilerden de çeşitli uygulamalar geliştirilebilir.

Veri işleme konusunda veriyi saklama şekli büyük önem arz etmektedir. Yukarıda bahsedilen veriyi işleme ve işlenen veriden faydalı programlar geliştirmek için, bu verileri en uygun veri yapısı kullanarak saklamak gerekmektedir. Verinin saklama şekli iyi kurgulanmalı, iyi modellenmeli ve esnek olabilmelidir. Böylece aynı veri seti kullanarak birden farklı uygulamalar, hatta işlev olarak da birbirinden farklı uygulamalar yapılabilir ve veri setini geliştirmeye katkıda bulunulabilir. Bu tarz veri saklama yöntemleri bir çok değişik alanda kullanılabilir. Metin karşılaştırma, cümle analizi veya arama motoru gibi uygulamalar geliştirilebilir. Bölüm 2.3'de bu alanlarla ilgili örnek uygulamalardan bahsedilecektir.

1.1 NoSQL Veritabanı

NoSQL veritabanı ilişkisel olmayan bir veritabanıdır. Bu tür veritabanları yatay olarak ölçeklendirilebilmeleri de dikkate alınarak, ilişkisel veritabanlarına alternatif çözümler sunmaktadır. Özellikle sosyal medya uygulamalarında sıklıkla tercih edilmektedir[3].

NoSQL veritabanları esnek şemalara sahiptir ve belirli veri modelleri için özel olarak tasarlanmıştır. Sağladıkları yüksek performans, ölçeklenebilirlik ve esneklik ile geniş çapta kabul görmüşlerdir[4].

İlişkisel(relational) veritabanlarıyla kıyaslandığında özellikle esneklik konusu büyük avantaj sağlar. İlişkisel veritabanındaki gibi nesneler belli kalıplara ve tablolara uymak zorunda değildirler. Şekil 1.1 de görüleceği üzere veri, şema ve ölçeklenebilirlik özelliklerinin karşılaştırılması bu durumu açıkça ortaya koymaktadır.

	SQL	NoSQL
Tip	İlişkisel	İlişkisel olmayan
Veri	Tablolarda saklanan yapılandırılmış veri	Json dosyalarında saklanan yapılandırılmamış veri.Çizge veritabanı ilişki destekler
Şema	Statik	Dinamik
Ölçeklenebilirlik	Dikey	Yatay
Dil	Yapılandırılmış sorgu dili	Yapılandırılmamış sorgu dili
Birleşme	Karmaşık sorgular için kullanışlı	Karmaşık sorgular için güçlü bir arayüz yok
OLTP	OLTP sistemler için en iyi çözüm	OLTP sistemler için önerilmez
Destek	Oldukça iyi	Topluluklara bağlı.Desteği onlar sağlamaktadır
Entegre Cache	Bellekte (SQL2014 ve SQL2016)	Entegre Cache desteklenir
Esneklik	Esnek olmayan katı şemalar	Katı değildir.Esnektir
Transaction	ACID	CAP teorem
Otomatik esneklik	Çoğu zaman hizmet dışı kalma zamanı gerektirir	Otomatiktir.Kesinti gerekmez

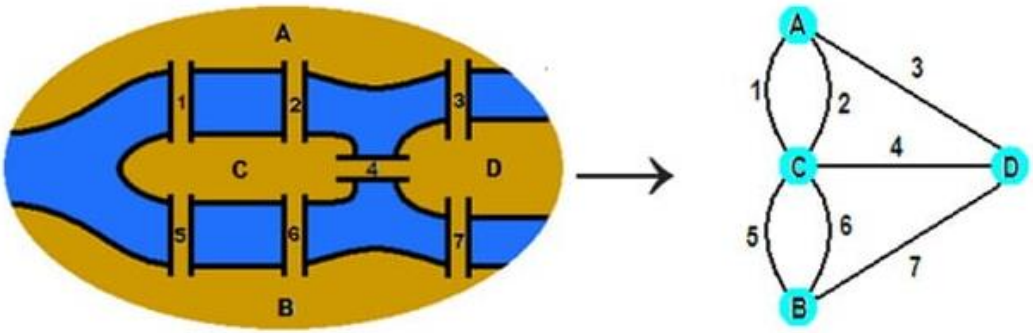
Şekil 1.1 SQL ve NoSQL veritabanlarının karşılaştırılması[44]

1.2 Graph(Çizge) Teorisi

Çizge teorisi işletme, matematik, bilgisayar bilimleri gibi bir çok farklı alanda kullanılmaktadır. Bu teori basitçe kenar(edge) ve düğümler(node) kullanarak bir problemi modelleme ve bu modeli bir çizge ile ifade etme esasına dayanır. G = (V,E) şeklinde tanımlanabilir. Bu tanım düğümler (vertices) ve kenarlar (edges) kümesi şeklinde bir çizgeyi ifade eder[5].

Bu teoriye dayanan çizge veri yapısını iyi anlamak önem arz etmektedir. 1.3 nolu bölümde anlatılmış olan çizge veritabanı konusu bu veri yapısına dayanmaktadır.

Çizge ile ifade modellenebilecek en güzel problemlerden birisi Euler problemidir. Königsberg Köprüleri Problemi olarak da ifade edilebilir. Königsberg şehrinde Eski ve Yeni Pregel nehirleri birleşerek Pregolya nehrini oluşturmaktadır. Bu nehirler şehri dört bölgeye ayırmaktadır ve nehir üzerine inşa edilen yedi köprü ile bu bölgeler birbirine bağlanmıştır. Peki şehrin herhangi bir noktasından başlayıp, her köprüden yalnızca bir defa geçmek şartıyla bir şehir turu yapılabilir mi? Euler bu problemi çözmek için kara parçalarını birer düğüm(node), köprüleri ise birer kenar(edge) olarak ifade etmiştir. Böylece şehri ve köprüleri bir çizge veri yapısına dünüştürmüş ve probleme bu açıdan bakmıştır. Şekil 1.2 de bu yaklaşım görselleştirilerek ele alınmıştır.



Şekil 1.2 Euler'in problemi çizge haline dönüştürmesi[6]

Bilgisayar programlamada da bu tarz yaklaşımlar büyük önem arz etmektedir. Bu alanda Euler'in yaptığı gibi problemleri çizge haline getirmenin yolları aranmış ve bazı yöntemler bulunmuştur. Adjacency Matris olarak ifade edilen iki boyutlu matrisler çizge temsilinde kullanılır. Java gibi bir çok programlama dili de iki boyutları matrisleri modellemek için uygun dillerdir. Aşağıda örnekleri gösterilmiştir.

Şekil 1.3 de gösterilen matrise göre aralarında ilişki olan elemanların satır-sütun eşleşmeleri 1 ile, olmayanlar ise 0 ile ifade edilmiştir. Bu matris java kodu ile aşağıdaki gibi temsil edilebilir:

```
int arr[][] adjacencyMatrix = {{0,1,0,1,0},{1,0,1,0,1},{0,1,0,1,0},{1,0,1,0,1},{0,1,0,1,0}}
```

	Bob	Alice	Mark	Rob	Maria
Bob	0	1	0	1	0
Alice	1	0	1	0	1
Mark	0	1	0	1	0
Rob	1	0	1	0	1
Maria	0	1	0	1	0

Şekil 1.3 Adjacency Matrix[7]

1.3. Çizge Veritabanı

Çok basit bir şekilde, bir çizge veritabanı, veriler arasındaki ilişkileri, veriler için eşit derecede önemli olarak ele almak için tasarlanmış bir veritabanıdır. Verileri önceden tanımlanmış bir modele kısıtlamadan tutulması amaçlanmıştır. Bunun yerine, veriler ilk çizildiği gibi saklanır ve her bir bireyin diğerleriyle nasıl bağlantı kurduğunu veya birbiriyle nasıl bağlantılı olduğunu gösterir[8].

Bu veritabanları bir çizge ifade eden verileri bu yapıya uygun olarak saklar ve üzerlerinde işlem yapabilmeye olanak sağlar. Özellikle sosyal medya tarzı uygulamalar için uygun bir ortam sağlayabilirler. Çünkü sosyal medya gibi platformlarda kişiler arasındaki ilişkiler önemlidir ve ilişkileri temsil etmek çizge veri yapısı, dolayısıyla çizge veri tabanları ile çok daha kolay ve yönetilebilir olmaktadır.Bölüm 1.5'de literatür taraması kısmında bu yapıdaki veritabanı örnekleri anlatılmaktadır.

Çizge veritabanları sadece sosyal medya gibi popüler konularda kullanılmamış, çok çeşitli alanlarda hizmete sunulmuştur. Örneğin birçok kaynaktan beslenen biyolojik datanın sorgulanması, görselleştirilmesi ve analiz edilmesi için uygulama geliştirilmiş, bu uygulama çizge veritabanı ile desteklenmiştir. Uygulamaya BioGraph denmiştir[47].

1.4 Amaç ve Önem

Bu tez çalışmasının amacı, metin tabanlı içeriklerin çizge yapısında modellenerek, bu modele uygun bir veritabanında saklanmasıdır. Bu şekilde saklanacak verinin, çeşitli amaçlara göre işlenmesinin daha kolay olacağı düşünülmüştür. Ayrıca doğru geliştirilmiş bir modelleme ile aynı veritabanından birden fazla bağımsız uygulamanın beslenebileceği ve birbirlerinin performanslarını olumsuz yönde etkilemeyeceği öngörülmüştür.

Bu tez çalışmasından hareketle, çizge yapısı ile modellenmiş Türkçe tabanlı bir veritabanı ile çeşitli şekillerde metin madenciliği yapılabileceği düşünülmüştür. Bu tezi desteklemek için web tabanlı iki farklı uygulama yazılmış, bu uygulamalar çizge yapısındaki aynı veritabanına bağlanmış ve elde edilen sonuçlar değerlendirilmiştir. Geliştirilen uygulamalardan ileriki bölümlerde detaylı bir şekilde bahsedilecektir.

Çizge veritabanı kullanılarak modellenen Türkçe metinlerin, yüksek performanslı arama motoru veya kişisel asistan programları gibi alanlarda gelişmelere katkıda bulunacağı düşünülmektedir.

1.5 Literatür Araştırması

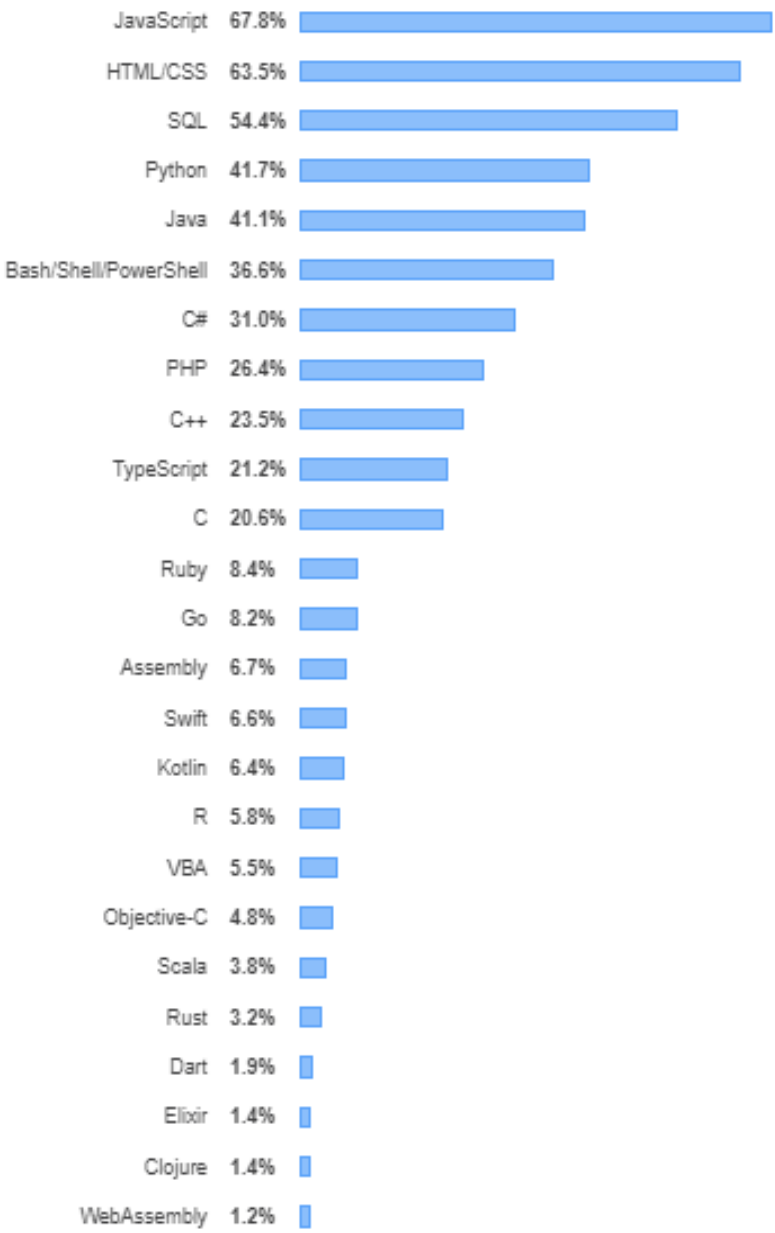
Araştırılması gereken en önemli nokta en uygun çizge veritabanı teknolojisini bulmaktır. Web tabanlı karşılaştırma programı[50] kullanılarak dünyadaki en popüler çizge veritabanları incelenmiştir ve tablolar halinde özellikleri karşılaştırılmıştır. Ayrıca Fernandes ve Bernardino tarafından yapılan çalışmada da yine popüler veritabanları birçok yönden karşılaştırılmıştır[51]. AllegroGraph, ArangoDB, InfiniteGraph, Neo4J ve OrientDB çizge veritabanlarının karşılaştırmaları bu makalede bulunabilir. Bu kapsamda yapılan araştırmalar sonucu tezin amacına en uygun veritabanının Neo4j olduğuna karar verilmiştir. Bunda en önemli etkenler şu başlıklar altında toplanabilir :

- Performans
- Ücretsiz sürüm
- Java uyumlu
- Yeterli kaynak
- Kolay öğrenilebilir sorgu dili
- Görselleştirme arayüzü

Neo4j ile ilgili yapılan araştırmalarda, Karagöz ve Komesli tarafından yazılan makale incelenmiştir[9]. Buradan sorgu diliyle ilgili örnekler ve veritabanının görsel hizmetleriyle ilgili bilgiler edinilmiştir. Ayrıca github üzerinden de çeşitli örnek uygulamalar incelenmiştir[10,11].

Türkçe ile ilgili yapılan bir doğal dil işleme çalışmasında, Agan ve Diri, Neo4j veritabanından bahsetmiştir. Dil bilimde, büyük miktarda metin içeren yapılandırılmış doküman kümelerine derlem denir. Derlemler üzerinde söz dizimsel sorgular yapılabilmesi için derlemlerdeki cümlelerin ilişki grafları bir graf veri tabanına kaydedilmiştir. Bunun için yüksek oranda bağlantılı verileri saklamakta kullanılan bir graf veri tabanı olan Neo4j kullanılmıştır. Neo4j, verileri düğümler ve bağlantılar şeklinde saklayan açık kaynaklı bir çizge veri tabanı olup, düğümlere ve bağlantılara etiket ve özellikler verilebilmektedir[12].

Veritabanı araştırmasından sonra diğer önemli konu uygulamaların hangi dille kodlanacağı konusudur. Java teknolojileri yaygın olduğundan dolayı, bilgi ve örnek kod bulma konusunda bir adım öne çıkmaktadır. Stack overflow 2019 anketi de şekil 1.4 de görüleceği üzere bu durumu desteklemektedir. Ayrıca açık kaynak kodlu olması da bir diğer avantajdır. Bu yüzden Java Virtual Machine(JVM – Java Sanal Makinesi) üzerinde çalışan dillerin araştırılması yönünde karar alınmıştır. Bu tarz diller bir başka java programı tarafından(interpreter) çalıştırılabilir veya bir derleyici(compiler) ile Java bytecode şeklinde derlenerek JVM üzerinde çalışabilir hale getirilir. Groovy, Kotlin, Scala gibi diller buna örnek gösterilebilir.



Şekil 1.4 Stack overflow 2019 yılı anketine göre en popüler teknolojiler[52]

JVM üzerinde çalışan diller incelendikten sonra özellikle listeler(array) üzerinde birçok işlemi yüksek seviyede – High Level(HL) - yapabilmeye olanak sağlayan Scala dilinde karar kılınmıştır. Bu tarz HL diller az miktarda yazılan kodla büyük işlemler yapılmasına yardımcı olur. Böylece daha anlaşılır program kodları ortaya çıkar.

Nesne tabanlı ve fonksiyonel tabanlı programlamayı bir arada sunan Scala programlama dili, geliştiricilere HL bir ortam sağlar[13]. Ayrıca Java kütüphanelerini native olarak kullanmaya olanak sağladığı için, Javanın tüm özelliklerinden faydalanmanın önünü açar.

Bu özellik çok kullanışlı olabilmektedir. Bazen istenilen bir kütüphanenin Scala ile yazılmış bir versiyonu olmadığında Java versiyonu kullanılabilir.

1.5.1 Büyük Veri

Dünyadaki veri büyüklüğü ve çeşitliliği, insanlık tarihinde daha önce hiç görülmeyen bir hızda artmaktadır. İnternet ve sosyal medyanın hayatımızın her alanına girmesi ve cep telefonlarının da bunu desteklemesiyle, insanlar günlük faaliyetlerinde bile veri üretir duruma gelmiştir. 2020 yılına kadar 2.5 milyar yeni katılan insanın çevrimiçi dünyaya adım atacağı ve 37 milyar yeni nesnenin de birbiriyle bağlı olacağı öngörüsü düşünüldüğünde gelecek yıllarda nesnelerin interneti ile birbirine bağlı nesneler,cihazlar ve sensörlerin daha da artacağı söylenebilir[34,46]. Bu kadar fazla veriyi işlemek ve analiz etmek ihtiyacı da bir o kadar artmış bulunmaktadır. İnsanlar da bu verileri işleyebilmek için yeni yöntemler, yeni algoritmalar kısacası yeni çözüm yöntemleri geliştirmektedir.

Büyük veri denilince akla sadece çok büyük miktardaki veriler gelmemelidir. Bu veriler aynı zamanda akışkandır. Örneğin, twitter gibi bir sosyal medya aracında atılam tweetler boyut olarak büyük yer kapladığı gibi aynı zamanda devamlılık da göstermektedir. Her saniye tweet atılmaktadır. Aynı şekilde youtube videoları da örnek gösterilebilir. Büyük veriyle ilgilenen teknolojilerin ele aldığı konulardan birisi de işte bu akan verilerin gerçek zamanlı işlenebilmesini sağlamaktır. Devasa boyutlardaki büyük veriden fayda sağlamak için gereken analizlerin karmaşıklığı, yeni teknolojilerin ve bunları yönetecek araçların gelişmesine neden olmuştur[33].

Büyük veri uygulamaları bir çok alanı etkilemektedir. Aslan ve Özerhan büyük verinin gelecek 10 yıl içinde muhasebe mesleğine olacak etkisi ile ilgili güzel bir çalışma yapmıştır[35]. Yapısal olmayan(unstructured) verileri anlamlı hale getirebilmek de büyük verinin ilgilendiği bir konudur. Warin ve Sanger finansal modellerden faydalanarak yapısal olmayan verilerin analiziyle ilgili çalışma yapmış ve finans dünyasının büyük veriyle ilişkisini ortaya koymuştur[36]. Keskin ve Yıldız tarafından yapılan çalışmada havayollarının sorunları büyük veri yaklaşımıyla ele alınmıştır. Havayolu şirketlerinin uçtukları mesafeler ve uçuş gecikme performanslarına yönelik kümeleme analizi yapılmıştır. Karar ağacı gibi veri yapılarından faydalanılmıştır[40]. Sağlık alanında da bir çok amaç için büyük veri teknolojilerinden faydalanılabilir. Farklı kaynaklardan üretilen

veriler; klinik uygulama ve araştırma, hastalık sürveyansı ve toplum sağlığı yönetimi, sağlık eğitimi ve öğrenimi, sağlık hizmetleri sunum kalitesinin ve verimliliğinin arttırılması, hastalıkların erken evrelerinde saptanması, klinik karar desteği alma, ilaç geliştirme ve pazarlama gibi alanlarda kullanılabilir[41].

1.5.1.1 Apache Spark

Apache Spark büyük verileri işlemeye yarayan bir büyük veri uygulamasıdır. Verileri paralel olarak işlemeye olanal sağlar. Ayrıca uygulamalar üzerinde küme(cluster) yapısı kurmak mümkündür. Böylece bir çok uygulama birbiriyle uyumlu cluster yapısında büyük verileri hızlı bir şekilde işleyebilir. Apache spark uygulamasının kendi resmi web adresinde bilinen en büyük spark cluster yapısının 8000 düğümden oluştuğu ifade edilmektedir[31]. Günümüzde büyük veri ile uğraşan bir çok kuruluş tarafından çokça tercih edilmektedir.

Spark, cluster ve veriyi paralel işlemenin yanı sıra, akan veriyi toplayıp üzerinde işlem yapabilmeye olanak sağlayan streaming arayüzü ve makine öğrenmesi için geliştirilmiş arayüzlere sahiptir. Java, Scala, Python dilleriyle geliştirme yapılabilir.

Apache Spark genellikle Hadoop ile kıyaslanan bir teknolojidir. Özellikle her iki teknoloji de "mapreduce" yöntemi kullandığı ve büyük veriyle ilgilendiği için bu kıyaslama önemlidir. Spark mapreduce yöntemini memory üzerinde yaparken Hadoop disk üzerinde uygular[42]. Bu durumda memory üzerinde çalıştığı için Spark çok daha hızlı işlem yapabilecektir. Fakat memory büyüklük açısından disk kadar kapasiteye sahip değildir. Bu durumda Hadoop daha büyük miktarda veriler üzerinde daha sağlıklı çalışabilir.

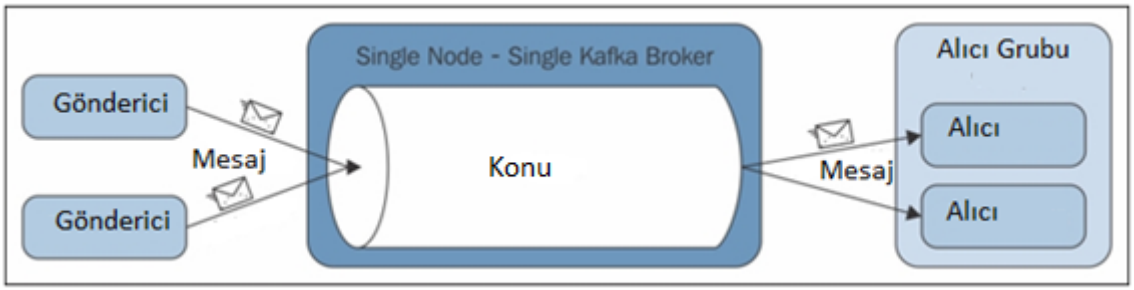
1.5.1.2 Apache Kafka

Apache Kafka, dağıtık mesajlaşma sistemleri kurmaya yarayan bir mesajlaşma sistemidir. Mesaj üreten(producer) ve mesaj işleyen(consumer) bileşenlerinden oluşur. Üreten ve mesaj işleyen bileşenleri ayrı uygulamalar olabileceği gibi, aynı uygulamanın farklı bileşenleri şeklinde de olabilir. Bir uygulama hem üreten, hem mesaj işleyen olabilir.

Kafka tek bir makine üzerine kurulup çalıştırılabilir veya küme(cluster) yapısını desteklemesi sayesinde birden fazla makine üzerinde kurulup küme yapılabilir. Böylece düğümlerden(node) birine bir şey olsa dahi diğer düğümler ayakta olduğu sürece sistem kesintisiz çalışmaya devam eder.

Mesajlar gönderilirken konu(topic) denen başlık altında mesaj işleyen bileşenlerine gönderilir. Mesaj işleyen bileşenleri bu konuları dinler ve yeni mesajlar geldikçe bu mesajları alıp işleyebilirler. Örneğin “araba” isimli konu için üretilen mesajlar bu konuyu dinleyen mesaj işleyen bileşenleri tarafından alınıp işlenebilirken, “bilgisayar” isimli konu içinse bu başlığı dinleyen mesaj işleyenler bu mesajları alabilir. Mesaj işleyen, hangi konuyu dinliyorsa, ona ait mesajları alır.

Mesaj işleyenler birbirinden grup id adı verilen etiketlerle ayrılır. Aynı grup etiketine sahip mesaj işleyenler dinledikleri konuyla ilgili bir mesaj geldiğinde bu mesajı sadece bir tane mesaj işleyen alır. Eğer tüm mesaj işleyenlerin grup etiketi farklıysa o zaman mesaj hepsine gönderilir. Bu şekilde yönetilebilir bir küme yapısı sağlanır. Şekil 1.5 de mesaj işleyenlerin bir grup olduğu görülmektedir. Aynı konuyu dinleyen bu gruba dinledikleri konu üzerinden farklı üretenler mesaj gönderebilir.



Şekil 1.5 Apache Kafka mimarisi[32]

Büyük veri uygulamalarında Kafka teknolojisi tampon gibi kullanılabilir. Devamlı akan verileri işlemek üreten ve mesaj işleyen bileşenlerinden oluşan, farklı ortamlarda çalışan uygulamalar geliştirilebilir. Üreten bileşenleri verileri işlemez, Kafka ile mesaj işleyenlere gönderir. Mesaj işleyenler de eldeki kaynaklara göre bu verileri alıp işleyebilir. Böylece olası dar boğazların önüne geçilmiş olur. Ayrıca mesaj işleyen bileşenlerinin tamamı devre dışı kalsa da üretilen mesajlar Kafka içerisinde kaybolmaz.Daha sonradan bu veriler işlenebilir. Veri kaybının önüne geçilmiş olur.

Kafka ile ilgili olarak başlangıç seviyesinde projeler bulmak mümkündür. Özellikle github üzerinde her konunun ayrı başlık altında değerlendirildiği örnek projelere ulaşılabilir[45]. Özellikle docker teknolojisiyle birlikte kullanımı daha modern ve popüler sistemler yaratabilmeye yardımcı olacaktır.

1.5.2 Doğal Dil İşleme

İşlenecek olan veri Türkçe metinler olunca Doğal Dil İşleme(DDİ) ile ilgili araştırmalar önem kazanmaktadır. Bu alanda özellikle ülkemizde yapılan çalışmaların, veri modelleme konusunda fikir vermesi açısından incelenmesi gerekir.

Adalı tarafından ele alınan DDİ ile ilgili makale bu konu hakkında genel bilgiler içermektedir[14]. Bu kapsamda makalede geçen konular şu alt başlıklarda toplanabilir:

- Yazım Yanlışlarının Düzeltilmesi
- Bul ve Değiştir
- Basılı Bir Metni Okuma
- Bir Metnin Özetini Çıkarma
- Metnin İçerdiği Bilgiyi Çıkarma
- Bilgiye Erişim
- Metni Anlama
- Metni Seslendirme
- Konuşmayı Yazıya Dökme
- Soru Yanıtlama
- Çeviri

Oflazer tarafından yapılan çalışmada da Türkçe'nin tarihi ve dil yapısı ile ilgili bilgiler verilmektedir. Heceler ve kök kelimeler üzerine anlatılanlar veri modelleme konusunda fikir verebilecek çalışmalardır[37].

Metin sınıflandırmayla ilgili çalışmalar da veri modelleme konusunda fikir verici olabilir. Metinle ilgili yazar, yazarın cinsiyeti, metnin konusu, yazarın ruh hali gibi konuları metinden çıkarabilmek için cümle, kelime, ek sayıları gibi özellikler kullanılabilir. Bu özelliklerin seçimi sınıflandırma performansına doğrudan etki etmektedir[43].

Metin madenciliği metni veri kaynağı olarak kabul eden veri madenciliği (data mining) çalışmasıdır. Metin üzerinden yapısallaştırılmış (structured) veri elde etmeyi amaçlar. Bunlardan bazıları şu şekilde sıralanabilir[38]:

- Metinlerin sınıflandırılması
- Bölütlenmesi (clustering)

- Metinlerden konu çıkarılması (concept/entity extraction)
- Sınıf taneciklerinin üretilmesi (production of granular taxonomy)
- Duygusal analiz (sentimental analysis)
- Metin özetleme(document summarization)
- Varlık ilişki modellemesi (entity relationship modelling)

Metinler işlenirken, bazı veri madenciliği algoritmaları da önem kazanmaktadır. K-Nearest Neighbor (KNN) algoritması gibi sınıflama yöntemleriyle metinler üzerinde sınıflama çalışması yapılabilir. Aynı zamanda bu tez çalışmasındaki örnek uygulamalardan da bir tanesinin konusu olan akademik çalışmalarla ilgili sınıflandırma çalışmaları mevcuttur. KNN algoritması ve R programlama dili kullanılarak tasnif etme çalışmaları yürütülebilir[39]. Özellikle matematiksel işlemler konusundaki başarısıyla bilinen R dili bu tarz çalışmalar için özellikle düşünülebilir. İstanbul Teknik Üniversitesinin geliştirdiği web tabanlı doğal dil işleme programı da cümleyi kelimelere ayırma, heceleri doğrulama gibi birçok konuyu içeren örnekler içermektedir[48]. Bu program da doğal dil işleme ile ilgili güzel bir kaynak teşkil etmektedir.

2. MATERYAL VE YÖNTEM

Bu bölümde hangi teknolojilerin kullanıldığı, çizge veritabanı üzerinde metinlerin nasıl modellendiği ve geliştirilen uygulamalardan bahsedilecektir.

2.1 Kullanılan Teknolojiler

Çizge veritabanı olarak Neo4j kullanılmıştır. Örnek uygulamalar play framework altyapısı kullanılarak, Scala programlama dili ile geliştirilmiştir. Ayrıca web üzerinden html olarak çekilen içerikleri ayrıştırabilmek için Jsoup kütüphanesi kullanılmıştır. Detayları alt başlıklarda anlatılmıştır.

2.1.1 Play Framework

Play framework Java ve Scala kullanarak web uygulamaları geliştirmeyi kolaylaştıran bir teknolojidir[15]. Bu tez çalışması kapsamında geliştirilen her iki uygulamada da bu teknoloji kullanılmıştır. Bu framework, hem sunucu tarafında kod yazmaya olanak sağlar, hem de kendine özgü html şablon oluşturma yapısıyla html tabanlı kullanıcı arayüzü oluşturmayı mümkün kılar.

2.1.2 Scala

Scala, Java sanal makinesi üzerinde çalışan, hem fonksiyonel hem de nesneye dayalı programlamayı destekleyen HL bir programlama dilidir. Listeler(array) üzerinde işlem yapılabilmesini kolaylaştıran metotlara sahiptir. Multithread programlamayı kolaylaştıran yapısıyla, işlemlerin paralel şekilde yürümesine olanak tanır. Fonksiyonel olması paralel programlama konusunda da kullanıcılara avantajlar sağlamaktadır. Paralel programlama prensipleriyle daha hızlı çalışan programlar geliştirmek mümkündür. Sayar ve Ergün ele aldıkları makalede Erlang ve Scala ile paralel programlama örnekleri göstermişlerdir[53].

2.1.3 Jsoup

Jsoup, html tabanlı dökümanlarla çalışmaya yarayan bir java kütüphanesidir. Jquery benzeri metotlarla html üzerinde işlemler ve manipülasyon yapmaya yardımcı olur[16]. Bu kütüphane ile html tabanlı metinler(string), Document Object Model(DOM) ağacına dönüştürülür ve tıpkı Jquery metotları gibi metotlarla DOM üzerinde işlemler yapılabilir.

Bu kütüphane sayesinde, geliştirilen uygulamalarda web üzerinden içerik kaynaklarına istek yapılmıştır. Gelen response Jsoup ile DOM ağacına dönüştürüldü ve işlem yapılacak kısım diğer alakasız kısımlardan ayrıştırılmıştır.

2.1.4 Neo4j

Neo4j, geliştiriciye görsel arayüz sunan, yüksek performanslı bir çizge veritabanıdır. Türkçe metinler bu veritabanı üzerinde modellenerek uygulamalar geliştirilmiştir.

Bu veritabanının sorgu diline Cypher denmektedir. Kullanımı ve öğrenmesi oldukça basittir. Sorgu hazırlarken yazmaktan ziyade şekil çizme hissi verir.Şekil 2.1 de bazı sorgu örnekleri gösterilmiştir.

```
//Yönlü çizge oluşturma
CREATE (p:Person)-[:LIKES]->(t:Technology)

//Oluşturulan yönün tersine yapılan sorgular sonuç döndürmez
MATCH (p:Person)<-[:LIKES]-(t:Technology)

//Yön kesin belli değilse sorguyu yönsüz yapmak daha iyi olabilir
MATCH (p:Person)-[:LIKES]-(t:Technology)
```

Şekil 2.1 Cypher diline ait bazı sorgu örnekleri[17]

2.2 Türkçe Metinlerin Neo4j Üzerinde Temsil Edilmesi

Cümle; “Bir düşünceyi, bir duyguyu, bir olayı, bir hareketi, bir isteği, bir yargı biçiminde anlatan kelimeler topluluğudur.” şeklinde tanımlanabilir[18]. Cümleyi oluşturan bu kelimeler birbirini ardışık olarak takip eden bir sıra şeklinde dizilir.

Eğer kelimeler üzerinde bu sıralamalar gösterilebilirse ve bu şekilde kelimeler arasındaki ilişki açıkça ifade edilebilirse, Türkçe metinler çizge üzerinde modellenebilir. Bu tez çalışmasında da bu düşünceden hareketle kelimeler arasındaki bu sıralı ilişki nasıl ifade edilir konusu üzerinde çalışılmış ve buna göre bir modelleme çalışması yapılmıştır.

Modelleme yapılırken bir kelime veritabanına sadece bir defa eklenmiştir. Bu kelimenin geçtiği içeriklere ait cümleler aynı kelime üzerinden, farklı ilişkiler yoluyla ifade edilmiştir. Bir kelime üzerindeki ilişki sayısı kadar cümle ifade eder. Kelimelerin

üzerindeki ilişkiler takip edilerek elde edilen yollar(path) cümlelerin oluşmasını sağlar. Cümle sonunu belirtmek içinse yolun sonunu ifade eden düğümler(node) kullanılır.

İlişkiler ifade edilirken, yolların birbirinden ayrıştırılabilmesi çok önemlidir. Her yol bir cümle ifade ettiği için kelimelerin arasındaki ilişkiler oluşturulurken ortak bir ifade olan anahtar kelime ve cümleyi yani yolu temsil eden bir numara verilmiştir. Aynı cümlede geçen kelimeleri birleştiren bu ilişkilerin hepsi aynı numaraya sahiptir. Aşağıda örneklerle daha detaylı anlatılmıştır.

Şekil 2.2 de "Ali ata bak" örnek cümlesinin yaratılması için kullanılan sorgu gösterilmiştir.

```
create ({value:'Ali',bilgi:'birincil'})-[:next1]->({value:'ata',bilgi:'birincil'})-[:next1]->
({value:'bak',bilgi:'birincil'})-[:next1]->({value:'endOfSentence'})
```

Şekil 2.2 Neo4j üzerinde cümle yaratılmasına örnek bir sorgu

Şekil 2.2 de görüldüğü üzere "Ali ata bak" cümlesi yaratılırken, kelimeler arasındaki ilişki "next1" adlı ilişki ile temsil edilmiştir. En sonda da "endOfSentence" değerli bir node eklenmiştir. Bu örnek üzerinden hareketle şöyle bir yorum yapılabilir: "endOfSentence" değerli düğüme(node) ulaşana kadar "next1" ilişkisiyle birbirine bağlı kelimeler takip edilirse bir cümle elde edilmiş olunur.

"endOfSentence" değerli node cümle sonunu ifade eder ve tüm cümlelerin sonu bu düğüme bağlanır. Burada önemli bir husus vardır. Bu düğüm daha değişik şekillerde de ifade edilebilir.Veritabanı modelinin daha anlaşılır olması açısından bu şekilde sade ve anlaşılır bir ifade kullanılmıştır. Örnek uygulamalarda daha başka şekillerde ifade edilmiştir. Sonraki bölümlerde detaylarıyla anlatılacaktır.

Kelimeler yaratılırken "bilgi : birincil" diye bir öznitelik dikkati çekmektedir. Bu özniteliğin amacı, cümle oluşturulurken bir kelime cümle içerisinde birden fazla yer alıyorsa, izlenen yolun bir cycle(döngü) ifade etmesinin önüne geçmektir. Yollar doğrusal olmazsa cümleleri doğru sorgulamak da mümkün olmayacaktır. Çünkü cümleler gerçek hayatta da doğrusaldır. Bir kelime cümle içinde birden fazla geçiyorsa, bir tanesi tüm cümleler tarafında kullanılan "birincil" öznitelikli kelimedir. Diğer kelimeler sadece o

cümle için istisnai olarak “bilgi : ikincil” özniteliği ile veritabanına eklenir ve “next” ilişkisi bu düğüm üzerinden sağlanır.

Şekil 2.3 de “sen ve ben mutlu ve mesut yaşamalıyız” örnek cümlesinin yaratılması için kullanılan sorgu gösterilmektedir.

```
match (a{value:'endOfSentence'}) create ({value:'sen',bilgi:'birincil'})-[:next2]->
({value:'ve',bilgi:'birincil'})-[:next2]->({value:'ben',bilgi:'birincil'})-[:next2]->
({value:'mutlu',bilgi:'birincil'})-[:next2]->({value:'ve',bilgi:'ikincil'})-[:next2]->
({value:'mesut',bilgi:'birincil'})-[:next2]->({value:'yaşıyoruz',bilgi:'birincil'})-[:next2]->(a)
```

Şekil 2.3 Neo4j veritabanında cümle yaratılması

Şekil 2.3 de bakılırsa, “sen ve ben mutlu ve mesut yaşıyoruz” cümlesi veritabanına eklenirken ikinci ve kelimesi “bilgi:ikincil” özniteliği ile eklenmiştir. Böylece aynı node üzerinden aynı cümlede iki ilişki geçmesi ve döngü oluşturması önlenmiş, doğrusallık korunmuştur.

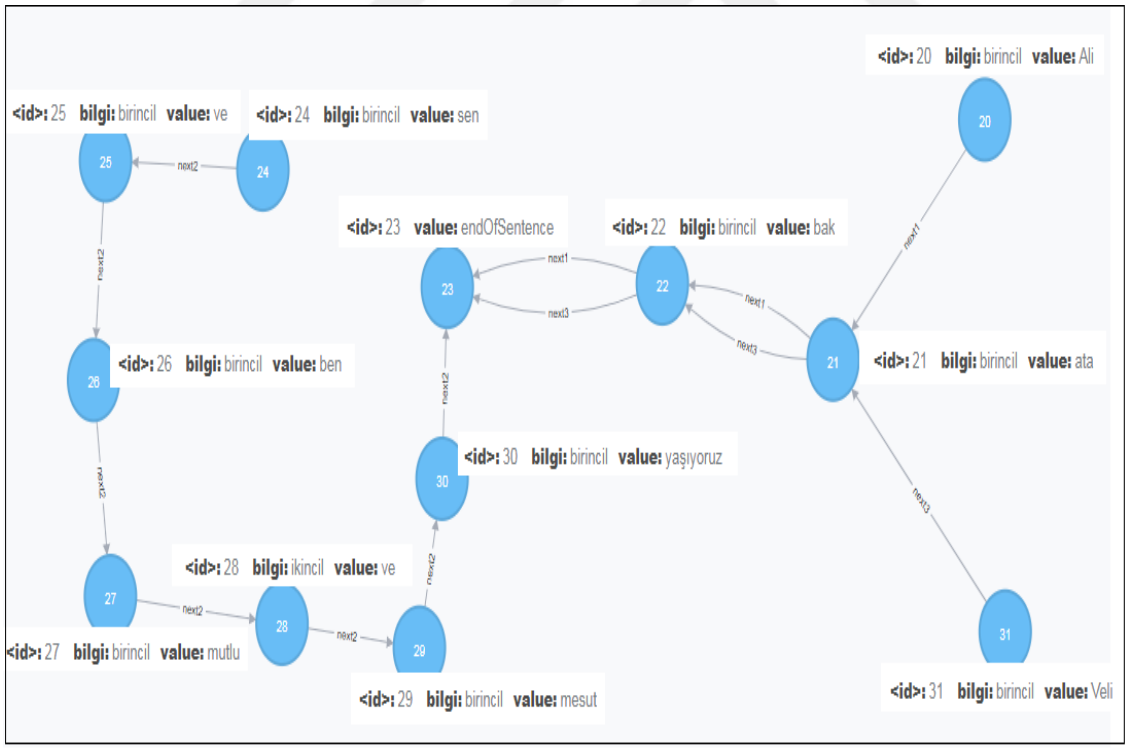
“endOfSentence” düğümü sorgunun başında “match” komutuyla eşlenmiş ve cümle sonu buna bağlanmıştır. Bu şekilde eklenen tüm cümleler aynı cümle sonu düğümüne bağlanır. Buna benzer bir ekleme koşulu veritabanına daha önce eklenmiş olan kelimelere de uygulanır. Veritabanına daha önce eklenmiş olan bir kelime match komutuyla bulunur ve yeni eklenen cümlede geçen bu kelime aynı düğüme next(n) ilişkisi ile bağlanır. Burada amaç her bir kelimeyi veritabanına yalnızca bir kere ekleyerek(ikincil öznitelikli kelimeler hariç) veritabanı boyutunun içerik ekledikçe büyümesini engellemektir. Bu çalışmada veritabanının Türkçe içerikler için oluşturulduğu düşünülürse, ne kadar içerik eklenirse eklensin, düğüm sayısı maximum Türkçe’deki kelime sayısı kadar olacaktır. ”Veli ata bak” cümlesinin veritabanına eklenme sorgusuyla(Şekil 2.4) konu daha iyi anlaşılabilir.

```
match (a{value:'ata',bilgi:'birincil'}),(b{value:'bak',bilgi:'birincil'}),(c{value:'endOfSentence'})
create ({value:'Veli',bilgi:'birincil'})-[:next3]->(a)-[:next3]->(b)-[:next3]->(c)
```

Şekil 2.4 Neo4j veritabanında cümle yaratılması

Örnek cümlelerin eklenmesiyle elde edilen veritabanının son hali Şekil 2.5 de olduğu gibidir. Eklenen cümleler:

- Ali ata bak
- sen ve ben mutlu ve mesut yaşıyoruz
- Veli ata bak



Şekil 2.5 Örnek cümleler ile Neo4j üzerinde oluşturulmuş veriler

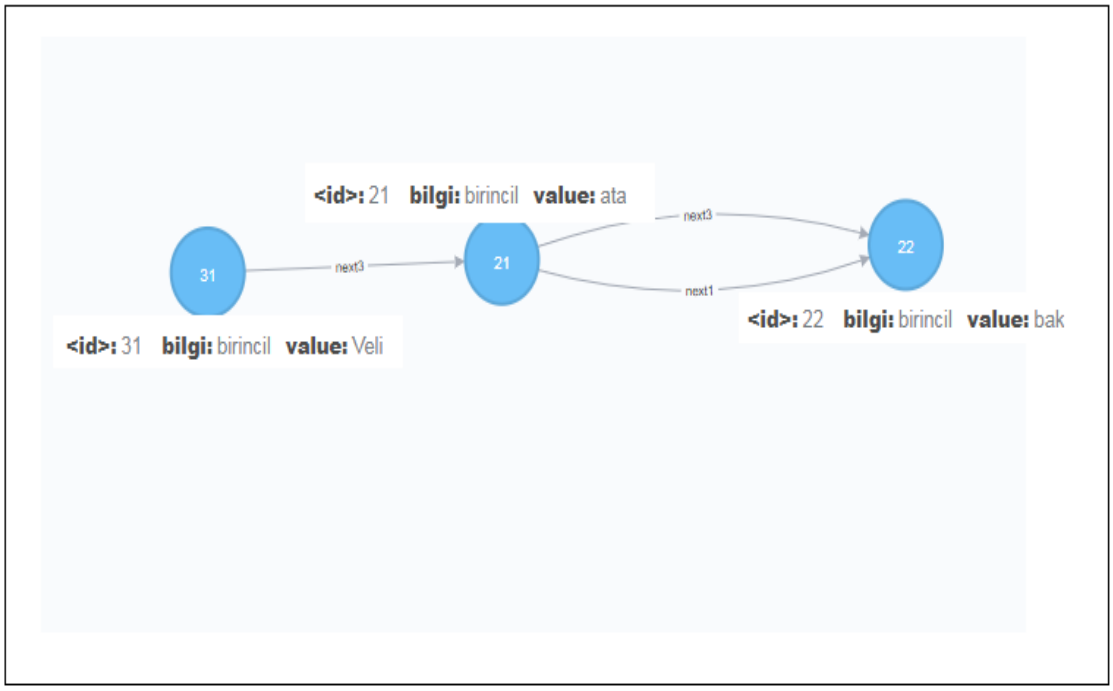
Şekil 2.5 de görüldüğü üzere “ata” ve “bak” kelimeleri iki farklı cümlede geçmesine rağmen her iki kelime de aynı düğümler kullanılarak temsil edilmektedir. ”ve” kelimesi aynı cümlede iki defa geçtiği için bir tanesi “birincil” diğeri “ikincil” özniteliği ile eklenmiştir. ”ve” kelimesi geçen bir başka cümle veritabanına eklenecek olursa cümledeki ilk “ve”, “birincil” öznitelikli düğüme bağlanacaktır. Varsa eğer diğer “ve” kelimeleri “ikincil” olarak ayrı düğümler halinde eklenecektir.

Buradan cümle sorguları yapılacak olursa path numarasıyla birlikte(next(i)) “endOfSentence” düğümüne kadar yolu takip etmek ve düğümlerin değerini almak gerekir. ”Veli ata bak” cümlesi Şekil 2.6 daki gibi sorgulanabilir.

```
match (a)-[n:next3*]->(b{value:'endOfSentence'}) return a
```

Şekil 2.6 Neo4j veritabanında temsil edilen bir cümleyi sorgulama

“Veli ata bak” cümlesi veritabanına 3 nolu yol ile eklenmiş ve “next3” ilişkisi ile kelimeler birbirine bağlanmıştır. ”endOfSentence” düğümüne “next3” ilişkisinin ifade ettiği path ile bağlanan düğümler sorgulandığında, cümleyi oluşturan düğümler elde edilecektir. Sorgu sonucu şekil 2.7 de gösterilmiştir.



Şekil 2.7 Neo4j üzerinde temsil edilen cümlenin sorgu sonucu

Sorgu sonucu Neo4j görselleştirme arayüzü ile bu şekilde gözükmektedir. Fakat bu sorgu bir program tarafından yapılıp sonuçlar kullanılmak istenirse, sorgu biraz revize edilerek kelimelerin sadece değerleri dönecek şekilde yazılabilir ve liste şeklinde sonuç alınabilir. Sorgunun bu şekilde revize edilmiş hali Şekil 2.8 de, sonucu ise şekil 2.9 da gösterilmiştir.

```
match (a)-[n:next3*]->(b{value:'endOfSentence'}) return a.value as value
```

Şekil 2.8 Neo4j veritabanında cümle sorgulama

Şekil 2.9 Neo4j veritabanında yapılan cümle sorgusu sonucunun tablo görseli

2.3 Geliştirilen Uygulamalar

Bu bölümde tez çalışması kapsamında geliştirilen iki farklı uygulamadan bahsedilecektir.

2.3.1 Makale Benzerlik Analizi

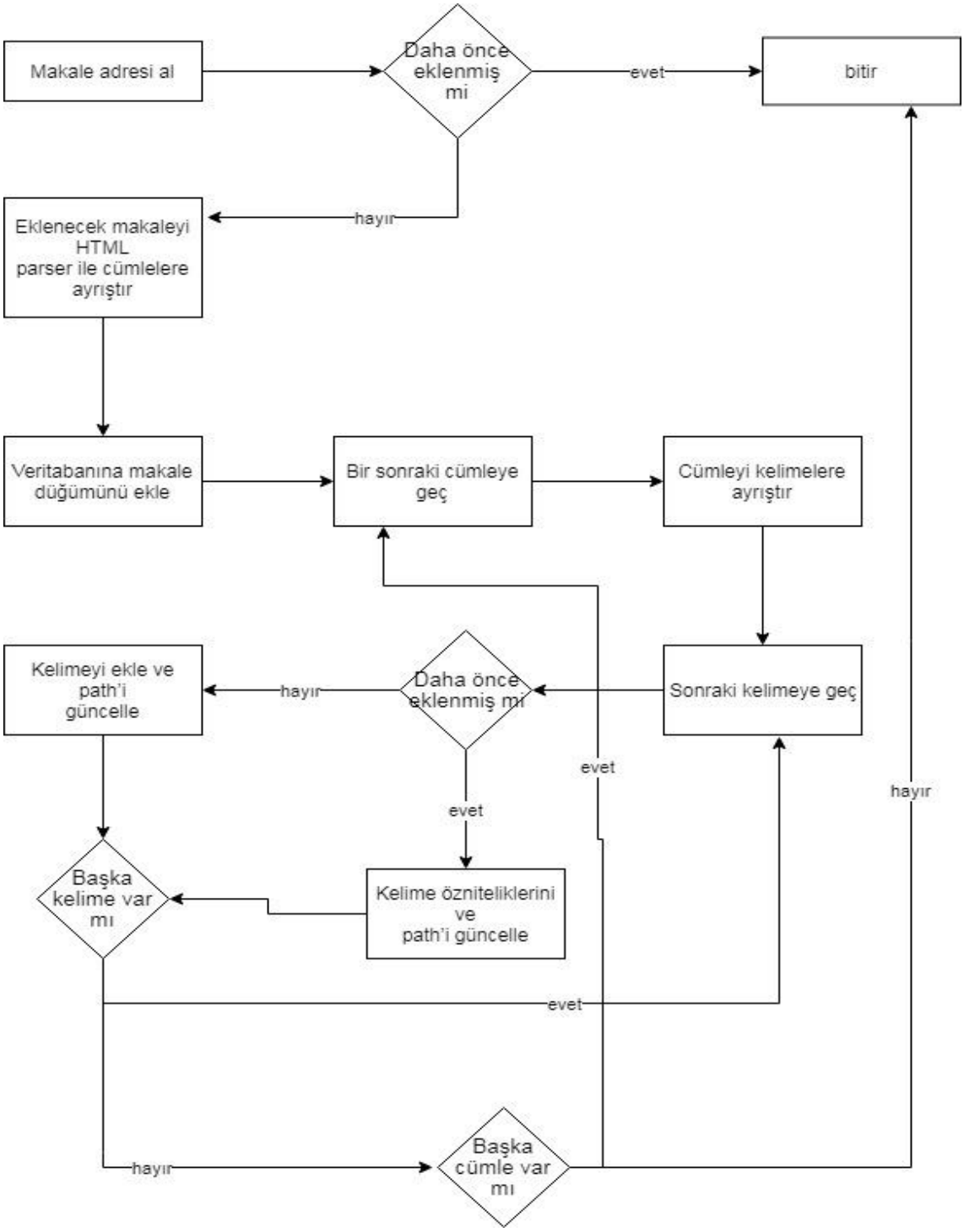
Veritabanında içerik olarak makale özetleri kullanılmıştır. Makale kaynağı olarak da TÜBİTAK ULAKBİM tarafından akademik dergiler için elektronik bir ortam sunan dergipark[19] sitesi kullanılmıştır. Uygulama kısaca dergiparktan 2.2 de anlatılan yönteme göre Neo4j veritabanına aktarılan makalelerin özet(abstract) kısımlarının benzerlik yönünden birbirleriyle karşılaştırılmasında kullanılmaktadır.

Web tabanlı bu uygulamanın başlıca bileşenleri şunlardır :

- Web adresi verilen bir makalenin özetini veritabanına ekleme
- Web adresi verilen bir makalenin özet kısmını veritabanındaki makalelerle karşılaştırma
- Arama motoru

2.3.1.1 Web adresi verilen bir makalenin özetini veritabanına ekleme

Makale özetini veritabanına eklemeye ilişkin akış diyagramı şekil 2.10 da gösterilmektedir.



Şekil 2.10 Veritabanına makale ekleme akış diyagramı

Veritabanındaki bazı önemli düğümlerden bahsetmek gerekir.Bu önemli düğümler şunlardır:

- key: maxSequence, sequenceValue : n (tamsayı)
- key: maxArticleNo, no : n (tamsayı)
- metadata: makale

maxSequence : next(n) ilişkisindeki, yani kelimeleri cümle olarak bağlayan yol ilişkisindeki(2.2) en son n değerini tutar. Böylece eklenen en son cümlenin kaçıncı cümle olduğu bilinir. Tüm cümleler bu değere göre bir döngü içerisinde gezilebilir.

maxArticleNo : maxSequence düğümüne benzer bir görevi vardır. Bu düğüm de eklenen makalelerden en sonuncusunun kaçıncı makale olduğunu tutar. Her yeni makale eklendiğinde 1 artar. Makaleler bir döngü içerisinde bu değere göre gezilebilir.

metadata : makale : Öznitelik değerinden anlaşılacağı gibi eklenen her bir makalenin bilgilerini tutar. Bu düğümün sakladığı diğer öznitelikler şekil 2.11 de gösterilmiştir. 2.2 bölümünde cümlelerin sonunu belirten "endOfSentence" düğümünden bahsedilmiş ve ihtiyaca göre düğüme öznitelik eklenebileceği belirtilmiştir. Burada metadata düğümü de tam olarak bu durumu ifade eder. Cümle sonlarını belirtir. Her makale için ayrı ayrı tutulur ve cümlelerin makalelere göre gruplanabilmesini sağlar.

Şekil 2.11 Makale belirten bir düğümün görseli

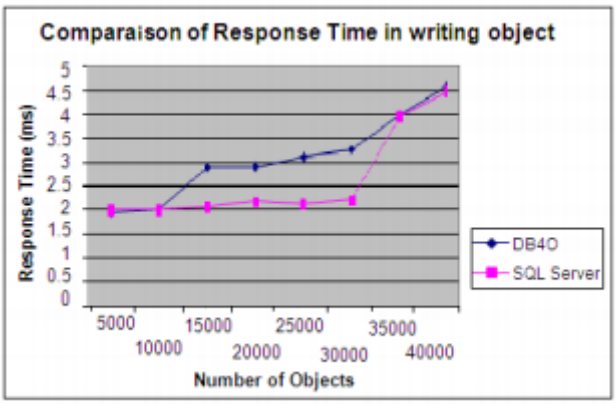
“makaleNo” özniteliği makalenin numarasıdır. Her makale için tekildir. ”rangeStart” ve “rangeEnd” makaleyi oluşturan cümlelerin kaçıncı cümleler olduğunun aralık bilgisidir. ”rangeStart” ile “rangeEnd” değerleri(bu değerler dahil) arasındaki cümleler makaleyi oluşturan cümlelerdir. ”subject” özniteliği makalenin başlığıdır. ”link” ise makalenin web adresidir.

Uygulamada kullanıcı tarafından girilen makale adresi (şekil 2.12) arka tarafa (server side) iletilir. Scala wsClient api’si ile bu adrese Hypertext Transfer Protocol (HTTP) tabanlı istek yapılır. HTTP, html gibi dökümanların transferini sağlamaya yönelik bir uygulama katmanı protokolüdür. Web tarayıcılar (Chrome, Firefox vs) tarafından desteklendiği gibi başka şekilde de kullanılabilir[20]. Yapılan istek sonucu gelen Html tabanlı cevap Jsoup kütüphanesi ile Html DOM ağacına çevrilir. Buradan makale konusu ve özet bilgileri ağaç içinden çıkarılır ve içinden özel özel karakterler(‘ , “ * vs) temizlenerek saf metin haline getirilir. Metin önce cümlelere sonra da kelimelere ayrıştırılarak veritabanına ekleme işlemine başlanır.

Kelime eklenirken ilk önce kelimenin veritabanında mevcut olup olmadığına bakılır. Eğer mevcut değilse kelime yeni bir düğüm olarak ait olduğu path ile birlikte yaratılır. Kelime veritabanında mevcut ise bu sefer veritabanından kelime getirilir ve bu düğüm üzerindeki path bilgisi güncellenir. Hangi makalelerde geçtiğinin bilgisi güncellenir. Kelime eklenirken dikkat edilen bir başka husus da kelimenin cümle içinde birden fazla geçip geçmediği konusudur. Eğer kelime cümle içinde birden fazla kez geçiyorsa birinci kelimeden sonrası “ikincil” olarak eklenir(2.2). Kelimeler eklenirken hem makale analizi hem de haber analizi uygulamasında kelimenin geçtiği cümle yapısı bozulmadan eklenir. Akış diyagramlarından görüleceği gibi (şekil 2.10) önce cümlelerden sonra kelimelerden kendi aralarında döngü kurulur. Her kelimenin geçtiği cümle döngü içinde bellidir.

Şekil 2.12 Veritabanına eklenecek makale adresi girilmesi(uygulama ekran görüntüsü)

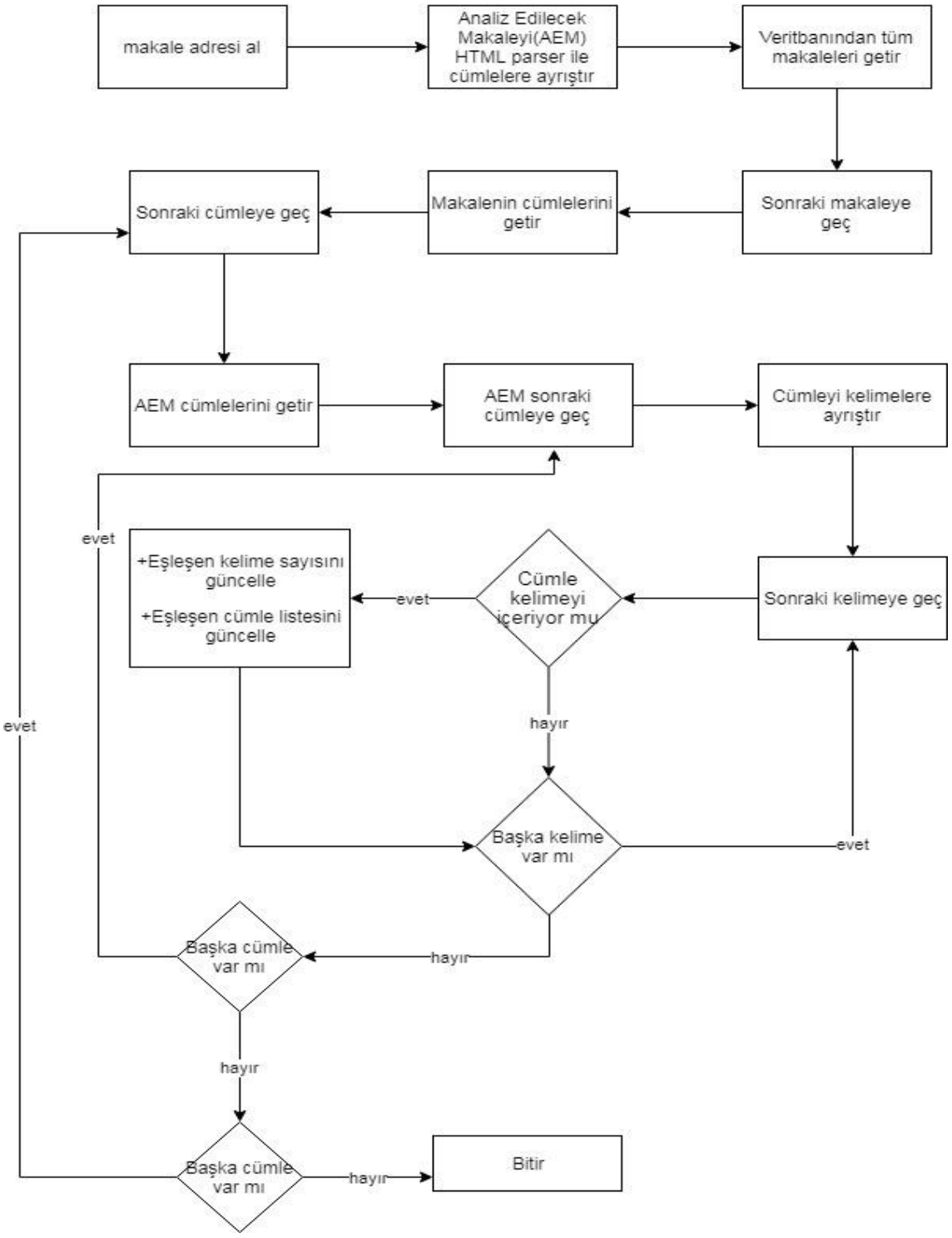
Çizge veritabanına nesne eklerken ilişkilerin de doğru şekilde eklenmesi gerekmektedir. Bu da hem veritabanı seviyesinde hem de programlama seviyesinde fazladan maliyet gerektirebilir. Bu yüzden insert işlemleri çizge veritabanlarında yavaş olabilmektedir. Şekil 2.13 de SQL veritabanıyla insert işleminin karşılaştırmasına dair grafik bulunmaktadır. Burada özellikle belli büyüklükte SQL server response zamanının biraz daha iyi olduğu görülmektedir.



Şekil 2.13 : İlişkisel ve nesne tabanlı veritabanı insert işlemi karşılaştırması[24]

2.3.1.2 Web adresi verilen bir makalenin özet kısmını veritabanındaki makalelerle karşılaştırma

Makale özetini karşılaştırmaya ilişkin akış diyagramı şekil 2.14 de gösterilmektedir.



Şekil 2.14 Makale analiz akış diyagramı

Adresi girilen makalenin saf metin olarak hazırlanması kısmı bir önceki kısımda anlatıldığı gibidir. Sadece ekranda input girilen kısım farklıdır (şekil 2.14). Özet kısım analiz edilmeye hazırlandıktan sonra analiz işlemlerine devam edilir.

Veritabanından makaleler gruplar halinde çekilir. Bunu yaparken 1'den "maxArticleNo" değerine kadar döngü kurulur. Her bir makale için "rangeStart" değerinden "rangeEnd" değerine kadar cümleler için döngü kurulur. Saf metin haline getirilen analiz edilen makalenin özet kısmıda cümlelere ayrılır. Veritabanından çekilen makalenin cümleleri için kurulan döngüde, döngü bitene kadar analiz edilen makalenin her bir cümlesi ile bu cümleler karşılaştırılır. Eğer analiz edilen makalenin cümlesindeki kelimelerden biri döngüdeki cümlenin içinde geçiyorsa benzerlik taşıyan cümleler listesine hem analiz edilen makalenin cümlesi, hem de veritabanındaki makalenin cümlesi ayrı listelere eklenir. Cümlede eşleşen kelime sayısının cümledeki toplam kelime sayısına oranı(analiz edilen makalenin cümlesi için) karşılaştırılan cümleler için benzerlik oranıdır. Bu şekilde tüm makaleler ve tüm cümleler için döngü sonlandığında cümle cümle benzerlik analizi yapılmış olur. Ayrıca her kelime eşleşmesinde toplam eşleşen kelime sayısı bir artırılır ve böylece toplam eşleşme oranı da toplam eşleşen kelime sayısının toplam kelime sayısına bölünmesi (analiz edilen makale için) ile bulunmuş olur. Sonuçların kullanıcıya gösterim için şekil 2.15 ve şekil 2.16 ya bakılabilir.

Eşleşen Makale

Makale Adresi	https://dergipark.org.tr/jeps/issue/43950/486478
Makale Başlığı	Türkiyede Metal Sektöründe Meydana Gelen İş Kazalarının Analizi
+	Cümle Bazında Eşleşmeleri Görmek İçin (+) İşaretine Tıklayınız
----------	----------
Makalenin Kelime Sayısı	104
Eşleştiği Makalenin Kelime Sayısı	120
Toplam Eşleşen Kelimeler	de ve ile Bu edilmiştir verileri Ayrıca
Toplam Eşleşen Kelime Sayısı	7
Toplam Eşleşme Oranı	6.730769230769231

Şekil 2.15 Makale sorgulama sonuç ekranı

Eşleşen Makale

Makale Adresi	https://dergipark.org.tr/jeps/issue/43950/486478
Makale Başlığı	Türkiyede Metal Sektöründe Meydana Gelen İş Kazalarının Analizi
+	Cümle Bazında Eşleşmeleri Görmek İçin (+) İşaretine Tıklayınız
----------	----------
Eşleşen Cümle	Ayrıca analizler sonucunda örgütsel desteğin özdeşleşmeye etkisinde örgütsel sinizmin de kısmi aracı rol üstlendiği görülmüştür
Eşleştiği Cümle	Metal sektörü dünyada olduğu kadar ülkemizde de önemli bir sektör konumundadır
Eşleşen Kelimeler	de
Eşleşme Oranı	6.666666666666667
----------	----------
Eşleşen Cümle	Bu araştırmada örgütsel destek sinizm özdeşleşme işe bağlanma ve politik davranış algısı arasındaki ilişkiler ele alınmıştır
Eşleştiği Cümle	Bütün sanayilerin lokomotifi niteliğinde olan metal sektörü barındırdığı iş gücü ve ekonomik büyüklüğü bakımından Türkiyenin en önemli sanayi kolu durumundadır
Eşleşen Kelimeler	ve
Eşleşme Oranı	6.25
----------	----------
Eşleşen Cümle	Araştırmada örgütsel desteğin sinizm aracılığıyla özdeşleşme üzerindeki etkisi ve örgütsel desteğin özdeşleşme aracılığıyla işe bağlanma ile politik davranış algısı üzerindeki etkisi incelenmiştir
Eşleştiği Cümle	Bütün sanayilerin lokomotifi niteliğinde olan metal sektörü barındırdığı iş gücü ve ekonomik büyüklüğü bakımından Türkiyenin en önemli sanayi kolu durumundadır
Eşleşen Kelimeler	ve
Eşleşme Oranı	4.545454545454546

Şekil 2.16 Makale sorgulama sonuç ekranı

2.3.1.3 Arama Motoru

Arama motoru, çıktı olarak sonuca ulaşmış bilgiler ve kayıtların her birini kıyaslayarak sorgulayan, bir sorgunun kabul edilebilirliği için gerekli işlemleri yapan, ele geçirilen verilerin yüksek performansta olmasını sağlayan bir sorgulama ve elde etme mekanizmasıdır[49].

Uygulamanın bileşenlerinden birisi de arama motorudur. Arama motoru, kullanıcının girdiği bir kelimenin, veritabanına kayıtlı hangi makalelerde geçtiğini bulur. Aranan kelimenin geçtiği makalelerin adresleri liste halinde kullanıcıya sunulur.

Arama motorunun nasıl çalıştığını anlatmadan önce “inverted index” kavramından bahsetmek gerekir. Metin tabanlı bir döküman kümesi vardır. Bu dökümanlar içerisindeki terimlerin hangi dökümanlar ile ilişkili olduğunun bilgisi bir yerde tutulur. Bu terimler içerisinde bir arama yapıldığında terimlere karşılık gelen dökümanlar hızlıca bukunabilecektir. Yani dökümanlara terimler üzerinden ulaşılıyor olunacaktır. İşte buna

inverted(ters) index denir[22]. Bu algoritmayı kullanan arama motoru uygulamasına örnek olarak elasticsearch gösterilebilir[23].

Arama motoru bileşeni de bu algoritmadan yola çıkılarak geliştirilmiştir. Bölüm 2.3.1.1'de anlatıldığı üzere makale içeriği veritabanına eklenirken kelime kelime ayrıştırılarak içerikler eklenmektedir. Kelime veritabanına eklenirken, "makaleler" özniteliği de güncellenir. Bu öznitelik kelimenin geçtiği makalelerin adreslerinin listesidir. Kelime veritabanına eklenirken, eğer veritabanına daha önce eklenmemişse yeni kelime olarak eklenip, "makaleler" özniteliği yaratılır ve tek elemanlı bir liste olarak makale adresini içerecek şekilde ekleme yapılır. Daha önce eklenmişse de bu sefer "makaleler" listesine makale adresi eklenerek güncelleme yapılır.

Kullanıcı ekrandan arama yapmak için bir kelime girdiğinde, bu kelime veritabanındaki tüm içerikler kelime kelime gezerek aranmaz. Çünkü her kelimenin hangi makale içerisinde geçtiğinin bilgisi zaten önceden bellidir. Bu yüzden veritabanından doğrudan aranan kelime düğüm olarak getirilir ve "makaleler" özniteliği liste olarak alınır. Tüm içerikler tek tek aranmayıp nokta atışı kelime aranmış olur. Bu algoritma sayesinde veritabanı ne kadar büyük olursa olsun arama performansı etkilenmemektedir. Milisaniye seviyelerinde bir arama performansı görülmektedir. Arama motorunu performansına ilişkin detaylı veriler 3 numaralı bölümde anlatılmıştır. Şekil 2.17 de "için" kelimesinin veritabanında düğüm olarak saklanışı gösterilmiştir. Şekil 2.18 de ise bu kelime için arama yapıldığında elde edilen sonuçların kullanıcıya gösterimi yer almaktadır.

Şekil 2.17 "için" kelimesinin düğüm olarak gösterimi

Şekil 2.18 “için” kelimesinin arama motorunda aratılması

2.3.2 Spor Haberi Analiz Uygulaması

Bu uygulama spor haberleri içerikleri sunan iki farklı web adresinden beslenmektedir [54,55]. Bu iki adresten alınan spor haberleri Neo4j veritabanına kaydedilmekte ve analiz için kullanılmaktadır.

Cümleler veritabanına kaydedilirken cümle sonundaki yükleme bakılır. Eğer yüklem gramer olarak olumsuz bir ifade ise cümle “negative”, olumlu ise “positive” düğümüne bağlanır. Örneğin yüklem “yaptı” ise olumlu yani “positive”, “yapmadı” ise olumsuz yani “negative” düğümüne bağlanır. Bu iki düğüm cümle sonunu ifade eden düğümlerdir. Tıpkı bir önceki uygulamada cümlelerin “makale” düğümüne bağlandığı gibi, burada da cümleler bu iki düğüme bağlanır.

Analiz edilen haberler de olumlu veya olumsuz cümlelere benzerlikleriyle ilgili değerlendirilirler. Analiz edilecek haber cümlelere ayrıştırılır ve her bir cümle veritabanındaki olumlu ve olumsuz cümlelerle karşılaştırılır. Benzerlik oranları hesaplanır.

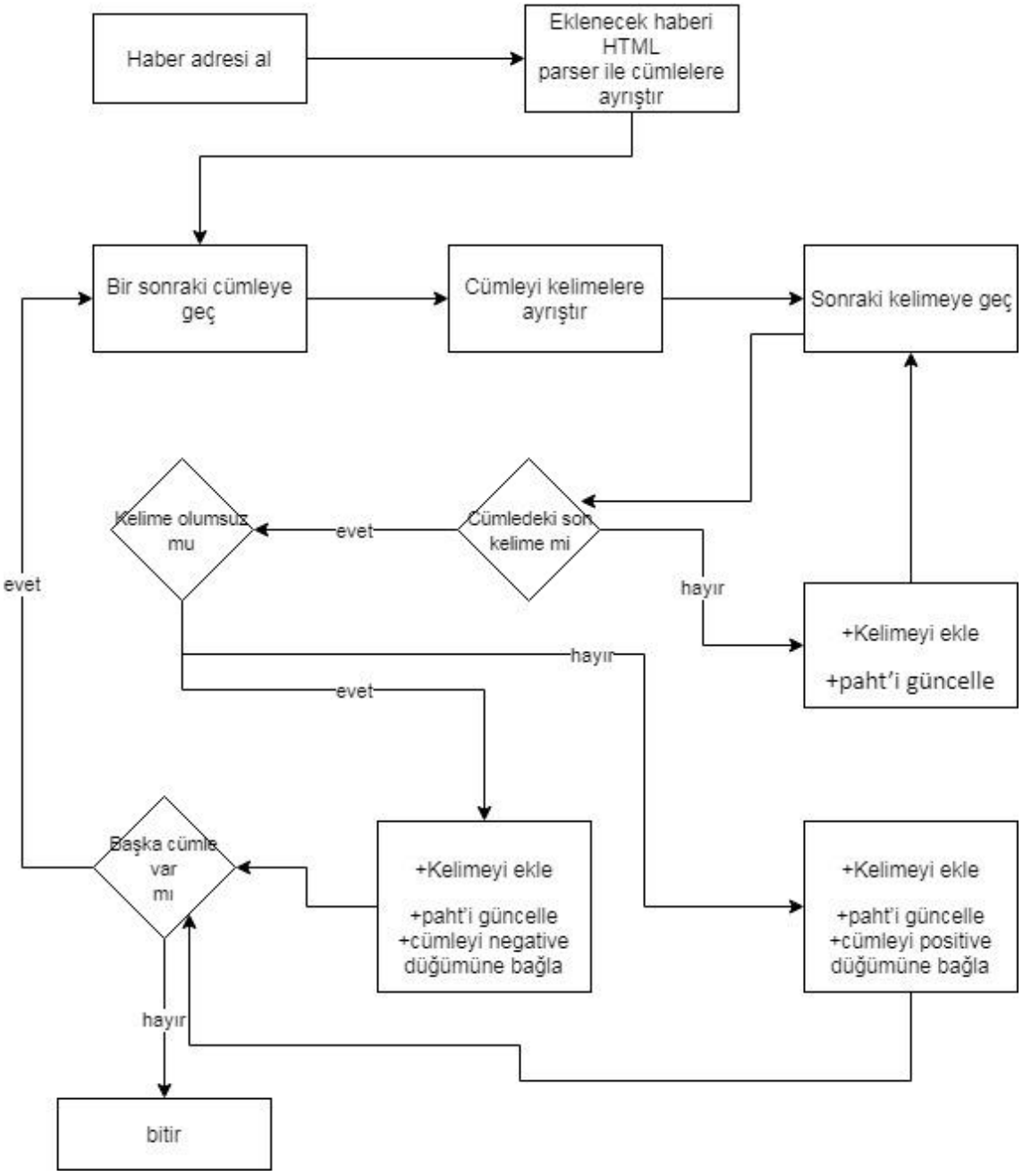
Uygulama iki ana bileşenden oluşmaktadır :

- Veritabanına haber ekleme

- Adresi verilen bir haberi analiz etme

2.3.2.1 Veritabanına Haber Ekleme

Veritabanına haber eklemeye ilişkin akış diyagramı şekil 2.19 da gösterilmektedir.



Şekil 2.19 Veritabanına haber ekleme akış diyagramı

Haberler iki şekilde eklenebilir. Ekrandan kullanıcı haber adresini girebilir veya her 10 dakikada bir çalışan periyodik fonksiyon Cnn Rich Site Summary(RSS)[26] adresinden haberlerin adreslerini çekerek ekleme işlemini yapan fonksiyonu çağırır.

RSS, özellikle haber sağlayan sitelerin, blogların ve podcastler gibi sürekli olarak güncellenen sitelerin güncellemelerinden geri kalmamak için kullanılan ve bazı site ve dosya formatları ile çalışan bir doküman takip sistemidir[27]. Bu servise istek yapıldığında güncel haberlere ait içeriklerin özeti, web adresi gibi bilgiler XML formatında gelmektedir. Bu XML ayrıştırma işlemlerinden geçirilerek istenen haber bilgilerine erişilebilir.

Bazı düğümlerin bilinmesi gerekir. Bunlar :

ftmaxSequence : Diğer uygulamada olduğu gibi bu uygulamada da cümleleri dolaşabilmek için en büyük path numarası tutulur. Path isimleri diğer uygulamadakinden farklı olarak "ftnext" anahtar kelimesi ile ifade edilir. Böylece iki uygulamanın ifade ettiği cümleler birbiriyle karışmamış olur.

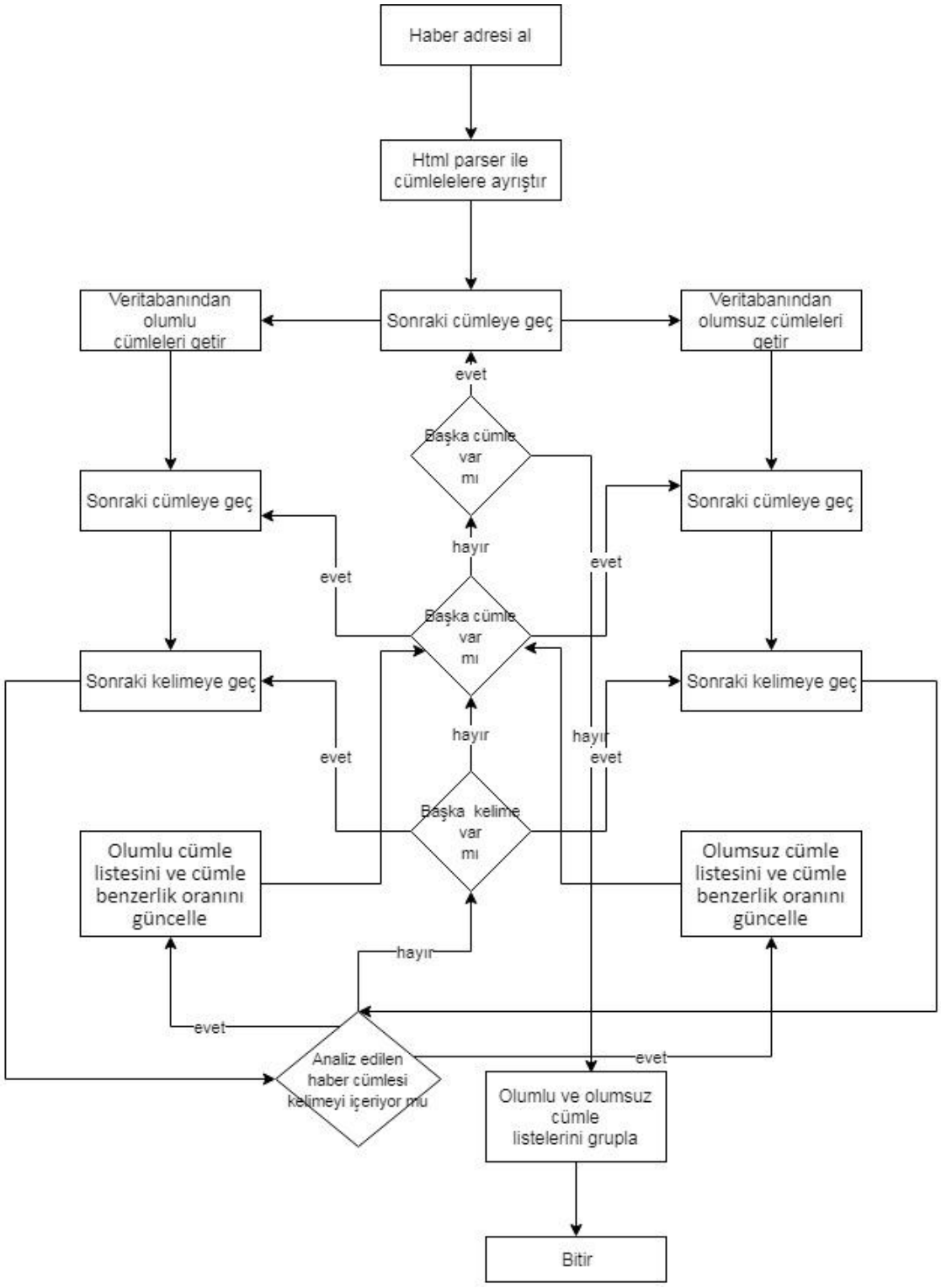
positive : Gramer bakımından olumlu cümlelerin bağlandığı düğümdür. Cümle sonu bu düğüm ile anlaşılabilir.

negative : Gramer bakımından olumsuz cümlelerin bağlandığı düğümdür. Bu düğüm de cümle sonunu temsil eder.

Haber ekleme kısmı makale benzerlik analizi uygulamasıyla benzerdir (2.3.1.1). Eklenecek haberin adresine http isteği gönderilir ve gelen cevap html ağacına Jsoup ile çevrilir. Haberin içerik kısmı ayrıştırılır. Saf metin haline getirilen içerik cümlelere ve kelimelere ayrıştırılarak veritabanına eklenir. Yüklem olumlu ise cümle "positive", olumsuz ise "negative" düğümüne bağlanır.

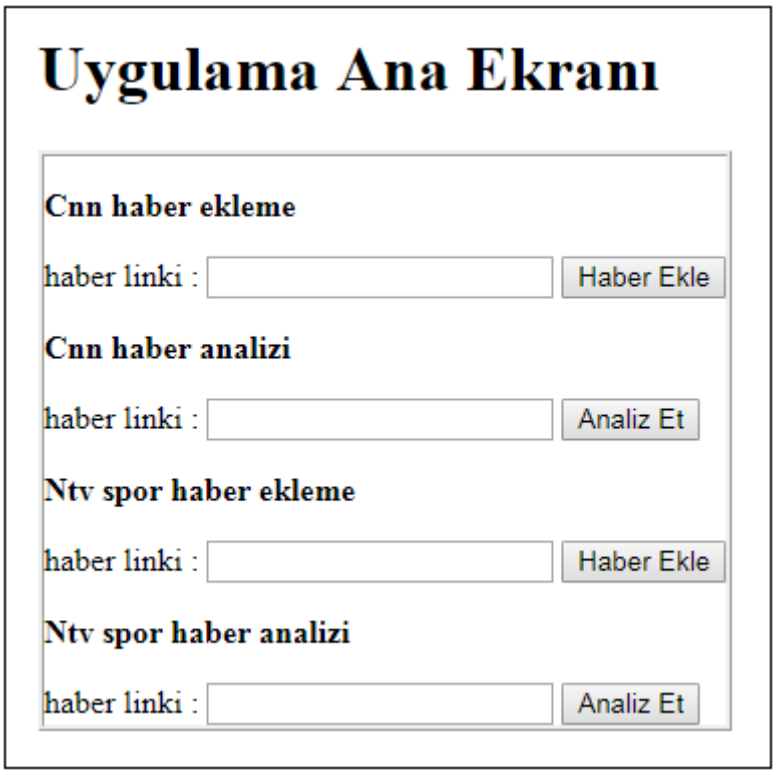
2.3.2.2 Adresi verilen bir haberi analiz etme

Spor haberinin analizine ilişkin akış diyagramı şekil 2.20 de gösterilmektedir.



Şekil 2.20 Haber analizi akış diyagramı

Kullanıcı şekil 2.21 deki ekran aracılığıyla makale adresini girer.



Şekil 2.21 Analiz edilecek haber adresini yazma ekranı

Kullanıcının girmiş olduğu adres (Şekil 2.21) arka tarafa iletilir ve daha önce anlatılan adımlar izlenerek haber içeriği saf metin haline getirilir. Metinden cümleler ayrıştırılır. ftmaxSequence düğümündeki değer kullanılarak veritabanındaki cümleler dolaşılır. Bunu yaparken "positive" ve "negative" düğümlerinin yardımıyla cümleler olumlu ve olumsuz olarak gruplanır. Analiz edilen haber içeriğindeki cümleler de bu gruplanmış cümleler ile karşılaştırılır. İki cümlede de geçen ortak kelimeler bulunur. Benzerlik oranları hesaplanır. Analiz edilen haberin cümlelerinin olumlu cümlelerle mi yoksa olumsuz cümlelerle mi daha fazla benzerlik gösterdiği ortaya koyulmuş olur (Sonuç ekranı için şekil 2.22). Eşleşmelerin doğru çalıştığını göstermek için veritabanında olan bir haberin adresi girilmiştir. Haberin kendisiyle başarılı bir şekilde eşleştiği görülmüştür.

POZİTİF EŞLEŞMELER

EŞLEŞEN CÜMLE	Konyada konakladığı otelden otobüsle Konya Havalimanına gelen ay-yıldızı ekip, Türk Hava Yollarına ait özel uçakla Keflavik Uluslararası Havalimanına doğru yola çıktı
EŞLEŞEN PATTERN	Konyada konakladığı otelden otobüsle Konya Havalimanına gelen ay-yıldızı ekip, Türk Hava Yollarına ait özel uçakla Keflavik Uluslararası Havalimanına doğru yola çıktı
EŞLEŞME ORANI	100.0
EŞLEŞEN KELİME(LER)	çıktı ---- yola ---- doğru ---- Havalimanına ---- Uluslararası ---- uçakla Keflavik ---- özel ---- ait ---- Yollarına ---- Hava ---- Türk ---- ekip, ---- ay-yıldızı ---- gelen ---- Havalimanına ---- Konya ---- otobüsle ---- konakladığı otelden ---- Konyada ----

EŞLEŞEN CÜMLE	A milliler, buradan kara yoluyla başkent Reykjavike geçecek
EŞLEŞEN PATTERN	A milliler, buradan kara yoluyla başkent Reykjavike geçecek
EŞLEŞME ORANI	100.0
EŞLEŞEN KELİME(LER)	geçecek ---- Reykjavike ---- başkent ---- yoluyla ---- kara ---- buradan ---- milliler, ---- A ----

EŞLEŞEN CÜMLE	Millileri uğurlamaya gelen bazı taraftarlar, oyunculara sevgisi gösterisinde bulundu ve fotoğraf çektirdi
EŞLEŞEN PATTERN	Millileri uğurlamaya gelen bazı taraftarlar, oyunculara sevgisi gösterisinde bulundu ve fotoğraf çektirdi
EŞLEŞME ORANI	100.0
EŞLEŞEN KELİME(LER)	çektirdi ---- fotoğraf ---- ve ---- bulundu ---- gösterisinde ---- sevgisi ---- oyunculara ---- taraftarlar, ---- bazı ---- gelen ---- uğurlamaya ---- Millileri ----

EŞLEŞEN CÜMLE	A Milli Futbol Takımı Teknik Direktörü Şenol Güneş ve bir milli futbolcu, maça ev sahipliği yapacak Laugardalsvöllur Stadyumunda yarın basın toplantısı düzenleyecek
EŞLEŞEN PATTERN	A Milli Futbol Takımı Teknik Direktörü Şenol Güneş ve bir milli futbolcu, maça ev sahipliği yapacak Laugardalsvöllur Stadyumunda yarın basın toplantısı düzenleyecek
EŞLEŞME ORANI	100.0
EŞLEŞEN KELİME(LER)	düzenleyecek ---- toplantısı ---- basın ---- yarın ---- Stadyumunda ---- Laugardalsvöllur ---- yapacak ---- sahipliği ---- ev ---- maça ---- futbolcu, ---- milli ---- bir ---- ve ---- Güneş ---- Şenol ---- Direktörü ---- Teknik ---- Takımı ---- Futbol ---- Milli ---- A ----

Şekil 2.22 Haber karşılaştırma eşleşme sonuç ekranı

3. BULGULAR VE TARTIŞMA

Bu çalışmada, çizge veritabanı üzerinde metinlerin temsil edilmesine yönelik modelleme çalışması yapılmış, geliştirilen modelin çalışmasını test etmek için iki adet web tabanlı uygulama geliştirilmiştir. Bu programlar farklı programlar olsa da aynı veritabanı kullanılmıştır. Farklı veritabanları yaratılmamıştır.

Uygulamalar veritabanına içerik ekleme ve içerikleri analiz etme konuları üzerine geliştirilmiştir. Makale analiz programında ayrıca arama motoru geliştirilmiştir. Değerlendirme kısmı üç ana başlık ile incelenecektir.

- Veri ekleme
- Analiz etme
- Arama motorunun analizi

3.1 Veri Ekleme

Her iki uygulamada da veri eklenirken kelimelerin daha önce eklenip eklenmediği veya aynı cümlede birden fazla yer alıp almadığı gibi kontroller yapılmaktadır. Bu kontroller belli ölçüde veri ekleme işlemlerini yavaşlatabilmektedir. Sayısal veriler üzerinden performans değerlendirmeleri detaylandırılmıştır.

3.1.1 30 Adet Haber Bulunan Veritabanına Spor Analizi Programı ile Haber Ekleme

Daha önceden 30 adet spor haberi eklenmiş olan veritabanına yeni bir haber eklendiğinde elde edilen sonucun milisaniye cinsinden değerlendirilmesi şekil 3.1 de gösterilmiştir. Bu sonuç elde edilirken veri ekleme işlemi başladığında zaman tutulmuş ve sonlandığında bir daha zaman tutulmuştur. Aradaki fark milisaniye cinsinden program loglarına yazılmıştır.

```
[info] 2019-06-09 20:37:36 - application - Cnn graph added for address : https://www.cnnturk.com/spor/futbol/ingiltere-uefa-uluslar-liginde-ucuncu-oldu
[info] 2019-06-09 20:37:36 - application - createGraphByCnnPost finished : 1197 ms
```

Şekil 3.1 Spor analizi uygulaması veritabanına içerik ekleme performansı

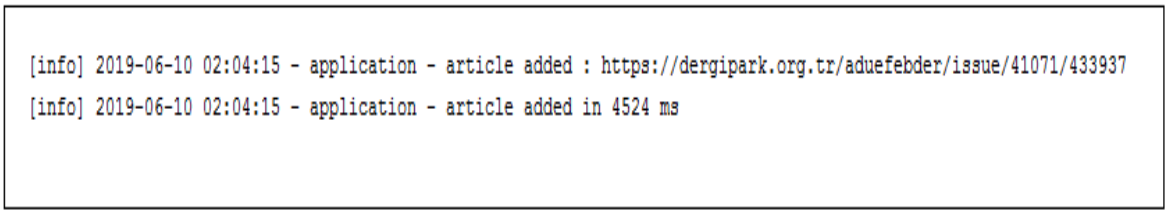
Şekil 3.1'de görüldüğü gibi haber içeriği ekleme işlemi 1197 milisaniye sürmüştür. Bu süre eklenen haberin içerik olarak büyüklüğüne bağlıdır. İçerik ne kadar büyük olursa kontrolü yapılacak kelime sayısı o kadar fazla olacaktır. Veritabanının büyüklüğü ise belli bir noktaya kadar etkili olacaktır. Çünkü bir kelimenin eklenip eklenmediği kontrol edilirken, her kelime bir defa eklendiği için Türkçe'deki tüm kelimeler eklendiğinde artık performans sabitlenecektir. Çünkü sorgulama yapılırken ilişkilere bakılmaksızın kelimeler doğrudan sorgulanmaktadır.

Şekil 3.2 de bir başka haberin içerik ekleme performansı görülmektedir. Bu sefer içerik olarak daha büyük bir haber seçilmiş ve ekleme süresinin 2878 milisaniye sürdüğü görülmüştür.

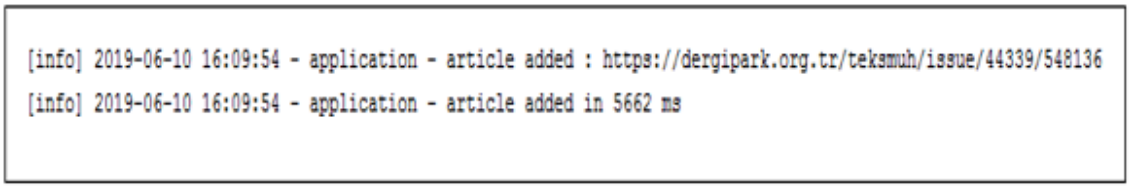
Şekil 3.2 Spor analizi uygulaması veritabanına içerik ekleme performansı

3.1.2 20,40 ve 50 Adet Makale Bulunan Veritabanına Makale Analiz Programı ile Makale Ekleme

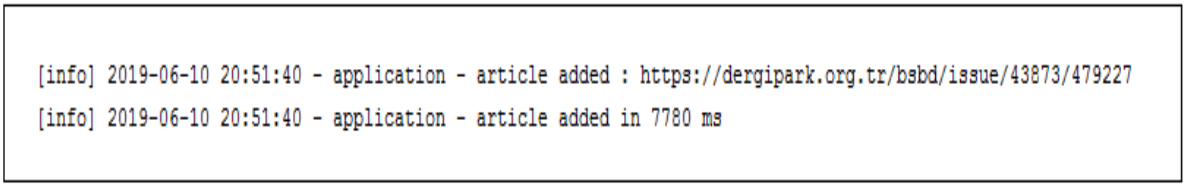
Önce içerisinde 20 adet makale bulunan veritabanına yeni içerik eklendiğinde ne kadar zaman aldığı analiz edilmiştir. Daha sonra bu sayı 40 ve 50 ye çıkarılıp sonuçlar karşılaştırılmıştır. Elde edilen bulgular şekil 3.3, şekil 3.4 ve şekil 3.5'te gösterilmiştir.



Şekil 3.3 20 makale bulunan veritabanına yeni makale ekleme performansı



Şekil 3.4 40 makale bulunan veritabanına yeni makale ekleme performansı



Şekil 3.5 50 makale bulunan veritabanına yeni makale ekleme performansı

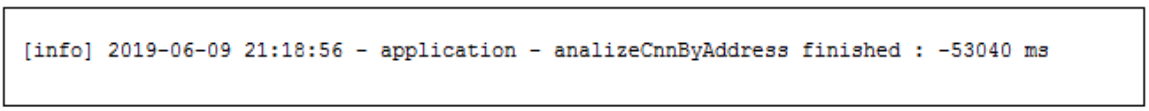
Makale sayısı 40'ın üzerinde çıkarıldıktan sonra yeni içerik ekleme performansı ortalama olarak 5-10 saniye aralığında seyretmiştir.

3.2 Analiz Etme

Bu bölümde haber analiz ve makale analiz uygulamalarının analiz sonuçları incelenmiştir. Her iki uygulama için de örneklem kümeleri oluşturulmuş, daha sonra analizler yapılmıştır. Uygulamalar aynı veritabanını kullandığı için, her iki uygulamanın da örneklem kümesi oluşturulduktan sonra analize başlanmıştır.

3.2.1 30 Adet Haber Bulunan Veritabanında Spor Analiz Programı ile Haber Analizi

Önceden 30 adet haber eklenmiş veritabanında adresi girilen bir haberin analizine ilişkin performans değerlendirmesi şekil 3.6 da gösterilmiştir.



Şekil 3.6 Haber analiz performansı

Şekil 3.6 da görüldüğü gibi analiz 53040 milisaniye sürmüştür.Analizi yapılan haberin içeriğinin büyüklüğüne bağlı olarak bu süre değişkenlik gösterebilir.Yapılan değerlendirmelerde analiz süresinin ortalama olarak 50 – 70 saniye aralığında değişkenlik gösterdiği görülmüştür.

3.2.2 20, 40 ve 50 Adet Makale Bulunan Veritabanında Makale Analiz Programı ile Makale Benzerlik Analizi

Performans değerlendirmesi yapılırken karşılaştırmaların daha sağlıklı olması açısından aynı makale üzerinden test yapılmıştır[21]. Şekil 3.7, şekil 3.8 ve şekil 3.9'da program loglarından elde edilen veriler gösterilmiştir.

```
[info] 2019-06-10 01:31:55 - application - article analyzed in 2884 ms
```

Şekil 3.7 20 makale bulunan veritabanında yapılan makale analiz sonucu

```
[info] 2019-06-10 16:46:59 - application - article analyzed in 7706 ms
```

Şekil 3.8 40 makale bulunan veritabanında yapılan makale analiz sonucu

```
[info] 2019-06-10 20:58:59 - application - article analyzed in 6733 ms
```

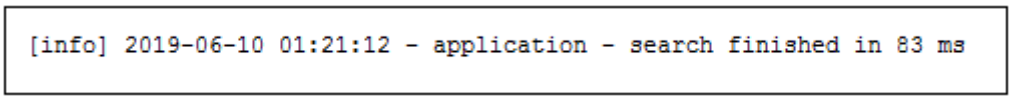
Şekil 3.9 50 makale bulunan veritabanında yapılan makale analiz sonucu

3.3 Arama motoru

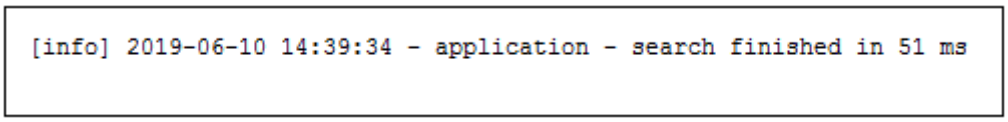
2.3.1.3 numaralı bölümde arama motorundan bahsedilmişti. Arama motorunun veritabanının büyüklüğünden bağımsız bir şekilde performans göstereceği öngörülmektedir. Aşağıda bu öngörüyü destekleyen performans testlerinin sonuçları yer almaktadır.

3.3.1 20,40 ve 50 Adet Makale Olan Veritabanında Arama Motoru Performansı

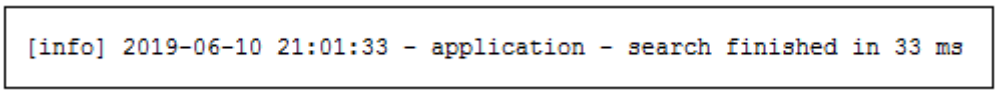
Arama motorunun performansı ölçülürken özellikle "için" kelimesi seçildi. Çünkü bu tür kelimeler genel kavramlar olduğu için hemen hemen her içerikte geçmesi muhtemeldir. Şekil 3.10, şekil 3.11 ve şekil 3.12'de program loglarından elde edilen veriler gösterilmiştir.



Şekil 3.10 20 makale olan veritabanında “için” kelimesi için yapılan arama



Şekil 3.11 40 makale olan veritabanında “için” kelimesi için yapılan arama



Şekil 3.12 50 makale olan veritabanında “için” kelimesi için yapılan arama

3.4 Genel Değerlendirme

İçerik ekleme ile ilgili performanslara bakıldığında spor haberleri analizi ve makale analizi arasında ciddi farklar bulunmamaktadır. Her iki uygulama da aynı şekilde veritabanına kelime kelime kayıt eklemektedir. Her kelime için önceden eklenip eklenmediğinin ve cümle içerisinde birden fazla kez tekrar edip etmediğinin kontrolü yapılmaktadır.

İçerik ekleme kısmı saniyeler seviyesine çıktığı için yavaşa yakın bir performans olarak kabul edilebilir. Oracle gibi dünyanın en yaygın ilişkisel veritabanı teknolojilerinden biri olan uygulamalarda milisaniyeler düzeyinde içerik eklemek mümkündür[25]. Burada dikkat çekilmesi gereken bir nokta vardır. Eğer bu tez çalışmasındaki uygulamalara içerik eklenirken belli bir modelleme yapılmadan içerikler doğrudan metin olarak eklenseydi, içerik ekleme kısmı ilişkisel veritabanı düzeyinde performans gösterebilirdi. Ancak bu sefer de çizge modelinin avantajlarından yararlanılamazdı.

Analiz ekleme performanslarına bakıldığında haber analizi ve makale analizi performansları arasında ciddi bir fark oluştuğu görülmektedir. Bu farkın temel sebebi içerik uzunlukları arasındaki ciddi farklardır. Makale özet kısımları ortalama 10-15 cümleden oluşurken spor haberleri 45-50 cümle seviyelerine kadar çıkabilmektedir. Bu

durumda haber analizi yapmak için taranması gereken cümle ve kelime sayısı da çok daha fazla olacaktır.

Makale haberlerinin analizlerine ait performanslara bakıldığında 20,40 ve 50 haber için çok fazla fark oluşmadığı görülebilir.Analiz için özellikle aynı makale adresi kullanılmıştır. Ortalama 5 saniye civarı süren performans söz konusudur.

Bu tez çalışmasındaki en önemli performans analizi arama motorunun performans testidir. Çünkü üzerinde çalışılan modellemenin getirmiş olduğu bir avantaj kullanılarak geliştirilmiştir. Arama motorunun performansının veritabanındaki içerik büyüklüğünden bağımsız olacağından daha önce bahsedilmişti. Performans testlerine bakıldığında 20,40 ve 50 makale içeriği bulunan veritabanı büyüklüklerinde milisaniye düzeyinde arama performansı görülmüştür.

Günümüzde birçok uygulamada metin tabanlı içerikler Oracle gibi ilişkisel veritabanlarında tutulmaktadır. Bu durumda saklanan veri üzerinde analiz edilmek için işlemler yapılmak istendiğinde veriler kütle(bulk) halinde tutulduğu için tek tek her dosyanın içerikleri taranmak zorundadır. Veritabanı belli bir büyüklüğe ulaştığında ise bu işlemler dramatik zaman seviyelerinde gerçekleşmektedir. Bu yüzden uygulama geliştiriciler veritabanı haricinde ayrıca elasticsearch gibi arama uygulamaları kullanıp verilerini bir de bu ortamda tutmaktadırlar. Elasticsearch doğru konfigürasyonlarla kullanıldığında yüksek arama pğerformansları sağlar. E-bay tarafından yayınlanan makaledeki Şekil 3.13 deki görselden görüleceği gibi milisaniye düzeyinde ortalamalar görülmektedir. Ancak elasticsearch kurulumu, bakımı ve sunucu üzerinde koşturulması gibi maliyetler getirmektedir. Metin tabanlı içeriklerin bu tez çalışmasındaki gibi Neo4j üzerinde saklanması sistem ile doğrudan entegre ve geliştirilmeye açık bir arama motoru altyapısı sağlayacağından tüm bu maliyetlerden kurtulmayı sağlayacaktır.

SeqId	test_name.keyword: Descending	Mean(ms)	Min	Max	50%	75%	95%	99%	<800ms(%)	800ms< x <1200ms(%)	> 1200ms(%)	Error percentage	count	throughput
0	replica_4	17	4	3,140	13	16	54	79	100	0	0	0	1	4,600.917
1	replica_3	20	4	2,273	14	18	61	86	100	0	0	0	1	4,115.166
2	replica_2	23	4	1,163	16	25	69	95	100	0	0	0	1	3,520.977
3	replica_1	33	4	2,491	26	40	81	109	100	0	0	0	1	2,590.521
4	replica_0	63	5	598	58	75	112	144	100	0	0	0	1	1,464.229

Gatling report

Şekil 3.13 elasticsearch performans analizi[28]

Bu tez çalışmasında kullanılan veri modeli içerik saklamada kullanılırsa, hem arama motoru gibi performans gerektiren işlemler için ekstra bir ortam gerekmemiş olur, hem de çizge veri yapısının avantajlarından faydalanılmış olunur.

Kelimelerin geçtiği makalelerin adresleri bir öznitelik olarak her kelime için saklanmıştır. Kelimeler birer düğüm(node) ifade ettiği için ihtiyaca göre istenildiği kadar öznitelik eklenebilir.İstenildiği kadar da ilişki eklenebilir. Böylece her uygulama için ayrı bir veritabanına ihtiyaç kalmamış olur. Her uygulama aynı veritabanını kullandığı için ve her uygulama kendi ihtiyacı olan öznitelikleri ve ilişkileri ekleyebildiği için, kullanılan verilerden uygulama bazında istatistik elde etmek de mümkün olacaktır. Bu yüzden geliştirilen bu veri modeli metin madenciliği ve veri madenciliği için de farklı yaklaşımlara yol açabilecektir.

Haber analizi uygulaması geliştirilirken literatür taraması bölümünde anlatılan Apache Kafka kullanılarak(1.4.1.2) üreten ve mesaj işleyen iki uygulama olarak geliştirilmek istenmiştir. Producer bileşeninin anlık olarak haber toplayıp saf metin haline getirmesi ve bu metinleri kafka aracılığı ile mesaj işleyen uygulamaya ulaştırması hedeflenmiştir. Mesaj işleyen uygulamanın da bu metinleri alıp veritabanına ekleme kısımlarını gerçekleştirmesi düşünülmüştür. Böylece yoğun haber toplama işlemleri olduğunda dar boğaz ihtimalinin önüne geçebilmek hedeflenmiştir. Ancak windows işletim sisteminde Kafka uygulamasının stabil çalışmadığı, özellikle kapatılıp açıldığında verilerin kaybolduğu ve mesaj işleme işlemlerinin hata aldığı aldığı görülmüştür. Bu nedenle Kafka uygulamasından vazgeçilmiştir. Apache Spark ve Kafka uygulamalarının kaynak sıkıntısı olmayan Linux tabanlı server üzerinde kurulması sistemin sağlıklı ve performanslı çalışmasını sağlar. Yeterli kaynak ve Linux tabanlı server makinesinin olmamasından dolayı Apache Spark ile planlanan cluster yapısından da vazgeçilmiştir.

4. SONUÇLAR

Çizge veritabanları günümüz dünyasında daha çok sosyal ağ ile ilgili çalışamalarda kullanılmaktadır. Doğal dil işleme ile ilgili konularda da kullanılmaktadır fakat sosyal ağ kadar yaygın değildir. Neo4j ise kullanımı henüz çok yaygınlaşmamış bir veritabanıdır. NoSQL veritabanları içerisinde Mongo ve Cassandra öne çıkmaktadır. Neo4j ile ilgili ülkemizde yapılan akademik çalışma sayısı oldukça azdır. Bu bağlamda tez çalışması yapılırken ilham alınabilecek akademik çalışma konusunda sıkıntı yaşanmıştır.

Elde edilen test sonuçlarında, veritabanının yatay olarak büyüdüğünde içerik ekleme ve analiz sonuçlarında ciddi performans kayıplarının olmadığı görülmüştür. Veritabanı ilişkisel olmadığı için istenen nesnelere erişmek hızlı olmaktadır. Veriler çekilip üzerlerinde işlem yapılırken de Scala programlama dilinin liste(array) işleme fonksiyonları kullanılmıştır. Bu işlemlerin de milisaniyeler seviyesinde bittiği görülmüştür. İlişkisel veritabanlarında veritabanı büyüdükçe ilişkili tablolar arasında kartezyen çarpımı kadar karşılaştırma yapıldığında belli bir büyüklükten sonra sorgular geç sonuç getirmekte ve zaman performansları yaşanmaktadır. Neo4j'de bu tarz bir sorun olmadığı görülmüştür. Veritabanlarında indexleme adı verilen yöntemle sorgu sonuçları çok daha hızlı gerçekleşebilir. İndexleme kısaca bazı nesnelerin bazı öznitelikleriyle beraber bir arama ağacına koyulmasıdır. Ağaçlarda arama yapıldığında zaman karmaşıklığı O(logn) seviyesindedir[29].

Elde edilen önemli sonuçlardan birisi de Scala programlama dilinin listeler üzerinde çok etkin çalıştığıdır. Dilin kendi fonksiyonları sıralama, gruplama, filtreleme gibi birçok işlemi performanslı bir şekilde yapabilmektedir. Listeleme, gruplama gibi işlemlerin çokça yapılacağı uygulamalar için Scala programlama dilinin oldukça ideal olduğu görülmüştür.

İki farklı uygulamanın aynı veritabanında çalışması uygulamalar arası entegrasyonu daha kolay olmasını sağlayabilir. Bu şekilde birbirinden farklı bir çok uygulama aynı veritabanından beslenebilir. Fakat burada güvenliğin de düşünülmesi gerekir. Uygulamaların veritabanına erişimiyle ilgili kısıtlamaların üzerinde çalışılmalı ve nasıl bir yöntem izlenebileceği üzerinde düşünülmelidir.

Türkçe'nin bu çalışmadaki gibi modellenmesinin konuşma robotları ve kişisel asistan tarzı programların geliştirilmesi için önemli olacağı düşünülmektedir. Makine öğrenmesi gibi metotlarla birlikte veritabanı güncellenip, kullanıcıdan gelen girdilere(insanların kurduğu cümleler) uygun yollar(path) bulunarak bu girdilere uygun cevaplar verilebilir.

Çalışmada dikkat edilmesi gereken önemli konulardan birisi de şudur: Bu çalışma Windows 7 işletim sistemi üzerinde çalışan 4 GB bellekli, intel i5 işlemcili kişisel bilgisayar üzerinde gerçekleştirilmiştir. Tüm veritabanı ve web tabanlı uygulamaların aynı bilgisayar üzerinde ayağa kaldırılıp çalıştırılması, performansa doğrudan etki etmektedir. Bilgisayar üzerinde zaten windows gibi çok kaynak tüketen bir işletim sistemi varken, bir de diğer uygulamaların da aynı bilgisayar üzerinde çalışması, performans kayıplarını kaçınılmaz hale getirmektedir. Normalde web tabanlı uygulamalar Linux işletim sistemli sunucular üzerinde çalışır.Veritabanı farklı bir server üzerinden ayağa kaldırılır. Aralarında gerekli protokoller üzerinden(TCP) veri alışverişi olur. İşlemci mimarisinin yapısı da performans konusunda önemli bir etkendir[30]. Tüm bunlar dikkate alındığında geliştirilen uygulamaların Linux tabanlı server'lar üzerinde gerçek birer ürün haline getirilerek çalıştırıldığında, performansların daha çok daha iyi olabileceği söylenebilir. Ayrıca Linux tabanlı server'larda kurulacak Apache Kafka ve Apache Spark uygulamalarıyla da çok daha kararlı, cluster yapıda uygulamalar geliştirilebilir. Hem performans olarak çok daha iyi sonuçlara ulaşılabilir, hem de sistemin ayakta kalma dayanıklılığı artırılmış olur.

KAYNAKLAR

[1] nodegraph, https://www.nodegraph.se/big-data-facts/ , Son erişim Haziran 2019

[2] Google asistant, https://assistant.google.com/intl/tr_tr/platforms/phones/ ,Son erişim Haziran 2019

[3] Öztürk, S. , Atmaca, H. (2017) . İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim BilişimSistemleri Kapsamında İncelenmesi. Bilişim Teknolojileri Dergisi ,10(2), 199-209

[4] aws, https://aws.amazon.com/tr/nosql/ , Son erişim Haziran 2019

[5] Seker, S.E. (2015) . Çizge Teorisi (Graph Theory) .YBS ansiklopedi, 2(2), 17-29

[6] matematikciler, https://www.matematikciler.com/konigsberg-in-yedi-koprusu/ , Son erişim Haziran 2019

[7] baeldung, https://www.baeldung.com/java-graphs , Son erişim Haziran 2019

[8] neo4j, https://neo4j.com/developer/graph-database/ , Son erişim Haziran 2019

[9] Karagöz, G.N., Komesli, M. (2017). Çizge Veri Tabanı Kullanılarak Geliştirilen Yazılım Lisans Yönetimi Amaçlı Veri Görselleştirmesi Uygulaması: BigLogVis. Dokuz Eylül Üniversitesi-Mühendislik Fakültesi Fen ve Mühendislik Dergisi , 19(57),779-789

[10] github , https://github.com/neo4j-examples/movies-java-spring-data-neo4j , Son erişim Haziran 2019

[11] github, https://github.com/neo4j-examples/neo4j-movies-java-bolt , Son erişim Haziran 2019

[12] Agan, C., Diri, B. (2016). Türkçe Derlemler İçin Söz Dizimsel Görselleştirme ve Sorgulama Aracı. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 9(1),1-10

[13] scala-lang, https://www.scala-lang.org/ , Son erişim Haziran 2019

[14] Adalı, E. (2012). Doğal Dil İşleme. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 5(2), http://dergipark.org.tr/tbbmd/issue/22245/238797

[15] playframework , https://www.playframework.com/ , Son erişim Haziran 2019

[16] jsoup, https://jsoup.org/ , Son erişim Haziran 2019

[17] neo4j , https://neo4j.com/developer/cypher-query-language/ , Son erişim Haziran 2019

[18] Akçataş, A. (2007). Türkiye Türkçesinde Yapı, İşlev Ve Anlam İlişkileri Açısından Cümle Grupları Ve Cümle Türleri Üzerine Bir Deneme II. Türk Dili Araştırmaları Yıllığı – Belleten , 55(2007/2) , 7-14

[19] dergipark, https://dergipark.org.tr/page/about , Son erişim Haziran 2019

[20] mozilla, https://developer.mozilla.org/en-US/docs/Web/HTTP , son erişim Haziran 2019

[21] Kerse, G. , Karabey, C. (2019). Örgütsel Sinizm ve Özdeşleşme Bağlamında Algılanan Örgütsel Desteğin İşe Bağlanma ve Politik Davranış Algısına Etkisi. Eskişehir Osmangazi Üniversitesi İktisadi ve İdari Bilimler Dergisi, 14 (1), 83-108.

[22] kulekci , https://elasticsearch.kulekci.net/blog/2016/03/26/inverted-index-nedir.html , Son erişim Haziran 2019

[23] elasticsearch , https://www.elastic.co/guide/en/elasticsearch/guide/current/inverted-index.html , Son erişim Haziran 2019

[24] Saxena, V. , Pratap , A. (2013) . Performance Comparison between Relational and Object-Oriented Databases . International Journal of Computer Applications , 71(22) , 6-9

[25] oracle database , https://www.oracle.com/tr/database/ ,son erişim Haziran 2019

[26] cnn rss , https://www.cnnturk.com/feed/rss/spor/news , Son erişim Haziran 2019

[27] webtekno , https://www.webtekno.com/yazilim/rss-nedir-en-iyi-rss-okuyucuları-h3161.html , Son erişim Haziran 2019

[28] ebay , https://www.ebayinc.com/stories/blogs/tech/elasticsearch-performance-tuning-practice-at-ebay/ ,son erişim Haziran 2019

[29] Kumar, P. (2013) .Quadratic Search: A New and Fast Searching Algorithm

(An extension of classical Binary search strategy). International Journal of Computer Applications , 65(14) , 43-46

[30] Bhandarkar, D. , Clark , W.,D. (1991) . Performance from Architecture: Comparing a RISC and a CISC with Similar Hardware Organization. ACM SIGPLAN Notices , 26(4) , 310-319

[31] spark , https://spark.apache.org/faq.html , Son Erişim Haziran 2019

[32] Gokalp G., http://www.gokhan-gokalp.com/wp-content/uploads/2016/10/kafka-flow-2.jpg , Son erişim Haziran 2019

[33] Doğan, K., Arslantekin, S. (2016) . Büyük Veri: Önemi, Yapısı Ve Günümüzdeki Durum. DTCF Dergisi, 56(1) , 15-36

[34] Aktan, E. (2018) . Büyük Veri: Uygulama Alanları, Analitiği ve Güvenlik Boyutu. Bilgi Yönetimi Dergisi, 1(1) , 1-22

[35] Aslan, Ü., Özerhan, Y. (2017). Big Data, Muhasebe Ve Muhasebe Mesleği . Muhasebe Bilim Dünyası Dergisi, 19(4) , 862-883

[36] Warin, T., Sanger, W. (2014). Structuring Big Data: How Financial Models May Help. Journal of Computer Science and Information Technology, 2(1) , 1-20

[37] Oflazer, K. (2012). Türkçe ve Doğal Dil İşleme. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 5(2), http://dergipark.org.tr/tbbmd/issue/22245/238795

[38] Seker, S.E.(2015) .Metin Madenciliği (Text Mining). YBS Ansiklopedi, 2(3), 30-32

[39] Kılınç, D., Borandağ, E., Yücalar, F., Tunalı, V., Şimşek, M., Özçift, A. (2016). KNN Algoritması ve R Dili ile Metin Madenciliği Kullanılarak Bilimsel Makale Tasnifi. Marmara Fen Bilimleri Dergisi, 3 , 89-94

[40] Keskin, M.V., Yıldız, D. (2018). Büyük Veri Araçları Ve R Kullanarak Amerikan Havayolu Firmalarının Sorunlarının Keşfedilmesi. Uygulamalı Sosyal Bilimler Dergisi, 2(2), 1-19

[41] Altındiş, S. (2018). Büyük Verinin Sağlık Hizmetleri Kalitesindeki Rolü. Sakarya Tıp Dergisi, 8(2),205-213

[42] scnsoft , https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce , Son erişim haziran 2019

[43] Amasyalı M.,F., Balcı, S., Mete, E. , Varlı, E. (2012) . Türkçe Metinlerin Sınıflandırılmasında Metin Temsil Yöntemlerinin Performans Karşılaştırılması / A Comparison of Text Representation Methods for Turkish Text Classification. EMO Bilimsel Dergi, 2(4) , 95-104

[44] microsoft, https://social.technet.microsoft.com/wiki/cfs-filesystemfile.ashx/__key/communityserver-wikis-components-files/00-00-00-00-05/6433.SQL-vs-NoSQL.jpg , Son erişim temmuz 2019

[45] github, https://github.com/confluentinc/kafka-streams-examples , Son erişim temmuz 2019

[46] Altınpulluk, H. (2018). Nesnelerin interneti teknolojisinin eğitim ortamlarında kullanımı. AUAd, 4(1), 94-111.

[47] Messina, A., Fiannaca, A., Paglia, L., Rosa, M., Urso, A. (2018). BioGraph: a web application and a graph database for querying and analyzing bioinformatics resources. BMC Systems Biology,12(5),75-89

[48] Cebiroğlu, G., 2013 , http://tools.nlp.itu.edu.tr/Tokenizer , Son erişim Haziran 2019

[49] Bat, D . (2011). Şirketler İçin Rekabette Sanal Farkındalık: Arama Motoru Pazarlaması. Gümüşhane Üniversitesi İletişim Fakültesi Elektronik Dergisi, 1 (1), 44-60

[50] predictiveanalyticstoday , https://www.predictiveanalyticstoday.com/top-graph-databases/ , son erişim temmuz 2019

[51] Fernandes, D., Bernardino, J. (2018). Graph Databases Comparison: AllegroGraph,ArangoDB,InfiniteGraph, Neo4J, and OrientDB. 7th International Conference on Data Science, Technology and Applications , 373-380

[52] stackoverflow, https://insights.stackoverflow.com/survey/2019#technology-_-programming-scripting-and-markup-languages , son erişim temmuz 2019

[53] Sayar, A., Ergün, U. (2014). Fonksiyonel Programlama Dilleri İle Paralel Programlama. Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi, 3(2), 1-17

[54] ntv spor, https://www.ntvspor.net/ , son erişim temmuz 2019

[55] cnn spor, https://www.cnnturk.com/spor , son erişim temmuz 2019

EK A. Makale Analizinde Veritabanı Örneklemi Oluşturulurken Kullanılan Makaleler

1. https://dergipark.org.tr/tjls/issue/38586/487466
2. https://dergipark.org.tr/tbbbd/issue/41828/504831
3. https://dergipark.org.tr/tbbbd/issue/22381/239630,
4. https://dergipark.org.tr/diledeara/issue/44040/542581
5. https://dergipark.org.tr/diledeara/issue/44040/542593
6. https://dergipark.org.tr/diledeara/issue/44040/542621
7. https://dergipark.org.tr/jeps/issue/43950/444190
8. https://dergipark.org.tr/gbesbd/issue/44318/472541
9. https://dergipark.org.tr/gbesbd/issue/44318/525245
10. https://dergipark.org.tr/khosbd/issue/45016/561199
11. https://dergipark.org.tr/khosbd/issue/45016/561195
12. https://dergipark.org.tr/jeps/issue/43950/451301
13. https://dergipark.org.tr/jeps/issue/43950/486478
14. https://dergipark.org.tr/pek/issue/41949/471152
15. https://dergipark.org.tr/folklor/issue/44848/533432
16. https://dergipark.org.tr/folklor/issue/44848/539632
17. https://dergipark.org.tr/ksusbd/issue/40204/349974
18. https://dergipark.org.tr/ksusbd/issue/40204/375636
19. https://dergipark.org.tr/usdad/issue/42045/480645
20. https://dergipark.org.tr/khosbd/issue/45016/561186
21. https://dergipark.org.tr/entoted/issue/42741/484979
22. https://dergipark.org.tr/eku/issue/44145/447167
23. https://dergipark.org.tr/eku/issue/44145/458121
24. https://dergipark.org.tr/enad/issue/45025/559640
25. https://dergipark.org.tr/enad/issue/45025/560595
26. https://dergipark.org.tr/enad/issue/45025/560770
27. https://dergipark.org.tr/enad/issue/45025/560823
28. https://dergipark.org.tr/gumusfenbil/issue/44470/410731
29. https://dergipark.org.tr/gumusfenbil/issue/44470/397074
30. https://dergipark.org.tr/gumusfenbil/issue/44470/418418

31. https://dergipark.org.tr/epfad/issue/43968/525223
32. https://dergipark.org.tr/epfad/issue/43968/532991
33. https://dergipark.org.tr/epfad/issue/43968/518344
34. https://dergipark.org.tr/comufbed/issue/45518/539309
35. https://dergipark.org.tr/comufbed/issue/45518/496286
36. https://dergipark.org.tr/teksmuh/issue/44339/548040
37. https://dergipark.org.tr/teksmuh/issue/44339/548113
38. https://dergipark.org.tr/tushad/issue/42516/449868
39. http://www.emakalat.com/issue/37830/476647
40. https://dergipark.org.tr/iisbf/issue/41627/494303
41. https://dergipark.org.tr/mbdd/issue/44252/453610
42. https://dergipark.org.tr/mamad/issue/41612/487431
43. https://dergipark.org.tr/oybd/issue/36073/405029
44. https://dergipark.org.tr/oybd/issue/36073/405032
45. https://dergipark.org.tr/umssd/issue/41788/461091
46. https://dergipark.org.tr/mbud/issue/41514/287911
47. https://dergipark.org.tr/gaunjss/issue/43763/518840
48. https://dergipark.org.tr/erusosbilder/issue/41602/486481
49. https://dergipark.org.tr/by/issue/40526/491969
50. https://dergipark.org.tr/bsbd/issue/43873/479227

ÖZGEÇMİŞ

Adı Soyadı : Tarık ŞAHİN
Doğum Yeri ve Tarihi : Kütahya - 1991
Yabancı Dili : İngilizce
E-Posta : tariksahin4391@gmail.com

Öğrenim Durumu

Derece	Bölüm/Program	Üniversite/Lise	Mezuniyet Yılı
Lise	Sayısal	Tavşanlı İMKB Anadolu Öğretmen Lisesi	2009
Lisans	Bilgisayar Mühendisliği	Ege Üniversitesi	2015

İş Deneyimi

Yıl	Firma/Kurum	Görevi
2015	Merkezi Kayıt Kuruluşu	Software Developer