EÜ FEN BİLİMLERİ ENSTİTÜSÜ

EGE ÜNİVERSİTESİ

YÜKSEK LİSANS TEZİ

YAPISAL OLMAYAN TÜRKÇE VERİLERİN BAĞLI VERİ KAYNAKLARIYLA ETİKETLENMESİ

Enes BULUT

Tez Danışmanı : Doç. Dr. Rıza Cenk ERDUR

Bilgisayar Mühendisliği Anabilim Dalı

Sunuş Tarihi : 18.08.2015

Bornova-İZMİR

2015

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

YAPISAL OLMAYAN TÜRKÇE VERİLERİN BAĞLI VERİ KAYNAKLARIYLA ETİKETLENMESİ

Enes BULUT

Tez Danışmanı : Doç. Dr. Rıza Cenk ERDUR

Bilgisayar Mühendisliği Anabilim Dalı

Sunuş Tarihi : 18.08.2015

Bornova-İZMİR
2015

Enes BULUT tarafından Yüksek Lisans tezi olarak sunulan "YAPISAL OLMAYAN TÜRKÇE VERİLERİN BAĞLI VERİ KAYNAKLARIYLA ETİKETLENMESİ" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 18/08/2015 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri: **İmza**

Jüri Başkanı : Doç. Dr. Rıza Cenk ERDUR .....
Raportör Üye : Doç. Dr. Murat Osman ÜNALIR .....
Üye : Doç. Dr. Murat Komesli .....

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "YAPISAL OLMAYAN TÜRKÇE VERİLERİN BAĞLI VERİ KAYNAKLARIYLA ETİKETLENMESİ" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

18 / 08 / 2015

Enes BULUT

ÖZET

YAPISAL OLMAYAN TÜRKÇE VERİLERİN BAĞLI VERİ KAYNAKLARIYLA ETİKETLENMESİ

BULUT, Enes

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Doç. Dr. Rıza Cenk ERDUR
Ağustos 2015, 45 sayfa

Bu tezde yapısal olmayan Türkçe özgeçmişler ve LinkedIn profesyonel sosyal ağ sitesinden sağlanan yarı yapısal Türkçe veriler bağlı veri kaynaklarıyla etiketlenerek anlamsallaştırılmıştır. Verilerin etiketlenmesinde kullanılan Türkçe doğal dil işleme aracının geliştirilmesinde kullanılabilecek kütüphaneler incelenmiş ve bu kütüphanelerden Zemberek kütüphanesi kullanılmıştır.

Etiketlenmiş verilerin ve çalışma kapsamında kullanılan diğer verilerin saklanmasında kullanılabilecek veri depolama altyapıları incelenmiştir. Verilerin saklanacağı en etkin veri depolama altyapısını belirlemek için anlamsal verilerin saklandığı üçlü depolama altyapıları ile NoSQL veritabanları incelenmiştir. Çalışmada melez veri altyapısını destekleyen Polyglot Persistence yaklaşımı benimsenmiştir.

Anahtar sözcükler: doğal dil işleme, melez veri, polyglot persistence, anlamsal ağ, bağlı veri, yarı yapısal veri, yapısal olmayan veri.

ABSTRACT

TAGGING UNSTRUCTURED DATA IN TURKISH LANGUAGE WITH LINKED DATA SOURCES

BULUT, Enes

MSc in Computer Eng.
Supervisor: Assoc. Prof. Dr. Rıza Cenk ERDUR
August 2015, 45 pages

In this thesis, unstructured resumes written in Turkish and semi-structured data retrieved from LinkedIn professional social website are tagged to make them semantic data. Zemberek NLP Library has been chosen following the investigation of natural language processing libraries which would be used in the development of the Turkish natural language processing tool that is used on tagging of text data.

Data storage structures that tagged data and the data used in this study are stored in are examined. Triple stores and NoSQL databases are examined in the decision of best options which keep semantic data. In this work, Polyglot Persistence approach is chosen which supports hybrid data infrastructure.

Keywords: natural language process, hybrid data, ployglot persistence, semantic web, linked data, semi-structured data, unstructured data.

TEŞEKKÜR

Bu çalışma ve yüksek lisans eğitimim süresince danışman desteği dışında da her türlü desteğini esirgemeyen Doç. Dr. Rıza Cenk Erdur'a,

Her koşulda beni destekleyen ve yanımda olan aileme, en içten saygı ve teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
W3C	World Wide Web Consortium
RDF	Resource Description Framework
URI	Universal Resource Identifier
RDFS	Resource Description Framework Schema
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language
DDİ	Doğal Dil İşleme
NLP	Natural Language Processing
NER	Named Entity Recognition
API	Application Programming Interface
REST	Representational State Transfer

1. GİRİŞ

Günümüzde teknolojinin gelmiş olduğu noktada bilgisayarlar insanlarla ve diğer bilgisayarlarla doğal üzerinden de iletişim kurabilir hale gelmiştir. Ancak bu kapsamdaki çalışmalar genel olarak İngilizce dili üzerinde yapılmaktadır. Bu tez çalışmasındaki amaç Türkçe dilinde yazılmış yapısal olmayan özgeçmişleri ve LinkedIn profesyonel sosyal ağ sitesinde yer alan yapısal olmayan ve yarı yapısal doğal dil içerikleri anlamsallaştırmaktır. Anlamsallaştırmakta kastedilen şey, ilgili verilerin diğer bilgisayarlar tarafından da anlaşılması ve insanların yapacakları sorgulara doğru bir şekilde cevap verebilmesidir.

Tez kapsamında geliştirilen uygulama ile yeni özgeçmişler ve LinkedIn sitesinde kullanıcıların profillerine ekleyecekleri yeni bilgilerin de takip edilmesi, bu verilerin etiketlenmesi (anlamsallaştırılması) ve alana ait anahtar kelimelerdeki olası değişikliklere sistemin sorunsuz uyum sağlayabilmesi gerçekleştirilmiştir.

Çalışmada tartışılan bir diğer önemli konu ise kullanılacak olan veri depolama mimarisidir. Bu kapsamda bağlı veri yaklaşımı ve melez veri altyapıları incelenmiştir. Bağlı veri anlamsal verilerin saklanması için geliştirilen bir depolama yöntemidir. Ancak, geliştirilen uygulama kapsamında sadece yapısal olmayan verilerin etiketlenmesi yapılmamakta olup etiketlenen veriler üzerinden kullanıcılara öneriler sunuyor olması nedeniyle melez veri altyapısının kullanılması uygun görülmüştür.

Türkçe doğal dil işleme çalışmaları birkaç farklı grup tarafından yürütülmektedir. Tüm çalışmalar incelenmiştir. Bu çalışmalar arasından açık kaynaklı olup en çok bilineni Zemberek kütüphanesidir. Tez kapsamında geliştirilen uygulamanın metin etiketleyici araç geliştirilmesi kısmında Zemberek kütüphanesinden yararlanılmıştır. Zemberek kütüphanesi kullanılarak ayrıştırılan metinsel veriler analiz edilerek alana özel veri kaynaklarıyla etiketlenmiştir. Bu işlemler sayesinde yarı yapısal ve yapısal olmayan Türkçe veri kaynakları anlamsallaştırılmıştır.

Çalışmada önerilen yöntem ile şirketlerin çalışanlarının LinkedIn sayfalarındaki yer alan yarı yapısal yetenek verileri ve yapısal olmayan özgeçmiş içerikleri kullanılarak çalışan profil haritası çıkarmaları hedeflenmiştir. Bu çalışan profil haritası ile yeni başlayacak bir projede hangi çalışanların yer alacağına veya herhangi bir geliştirmeyi hangi çalışanın yapacağına karar vermede kullanması öngörülmüştür. Kısacası şirketlerin çalışanları arasında iş bölümü ve görevlendirme yapması sırasında doğru çalışma ekiplerinin kurulmasında rol oynaması hedeflenmiştir. Doğru çalışma ekiplerinin kurulması ile iş gücünde verimlilik elde edilecektir.

1.1. Bu Alandaki Önceki Çalışmalar

Türkçe özgeçmişlerden anlamsal veri çıkarımı üzerine Çelik ve Elçi (2012) tarafından yapılan çalışmada metinlerden veri çıkarımının gerçekleştirilmesi açıkça göz önüne serilmemiştir. Kısaltmalar, son ekler vb. için çeşitli işlemler yapıldığı belirtilirken, örneğin metinde geçen bir mesleğin nasıl belirlendiği (etiketlendiği) net değildir. Ayrıca ilgili çalışmada, yetenek terimlerinin metinlerden çıkarılması ve anlamsal bağlı veriye dönüştürülmesi çalışmanın üstünde durduğu konularından biri değildir. Tez projesinde özgeçmişlerden Türkçe ve İngilizce dilinde yazılmış yetenek terimlerinin çıkarılması (etiketlenmesi) önde gelen amaçlardan biridir.

Tez projesi kapsamında Türkçe metinler üzerinde çalışan bir metin etiketleyici, veri çıkarıcı araç geliştirilmiştir. Bu araç, İngilizce ve başka yabancı diller üzerinde çalışabilen OpenCalais ve PoolParty (Schandl and Blumauer, 2010) araçlarının yapmakta olduğu metinlerden kişi, olay, etiket, lokasyon bilgisi vb. verilerini çıkarmayı (etiketlemeyi) ve bu verileri bağlı veri kaynaklarıyla bağlama işlemini Türkçe dilinde de gerçekleştirmeyi hedeflemektedir. Verilerin bağlı veri kaynaklarıyla bağlanması, verinin en iyi şekilde kullanılmasını sağlamakta ve kullanıcıların istedikleri verilere uzun aramalar yapmadan kolayca ulaşmasını sağlamaktadır. Böyle bir aracın gerçekleştirilmesi hedefi doğrultusunda, Türkçe dilinin yapısı ve dilbigisi kuralları gereği bazı zorluklar öngörülmektedir. Eş sesli sözcükler, sözcük ekleri, noktalama işaretleri vb. gibi bazı durumlarda da aracın en az hatayla çalışması beklenmektedir. Metin etiketleme aracı, belli bir alana özgü olmayıp, istenilen her türlü alan üzerinde çalışması ve istenilen sonuçları vermesi hedeflemekte ve beklenmektedir.

GATE (Cunningham et al.,2002) bir takım dil işleme araçlarını içeren, Sheffield Üniversitesi tarafından geliştirilmiş bir mimaridir. GATE'in bilgi çıkarıcı sistemi kural tabanlı, öğrenme gerektirmeyen bir sistemdir. Evrensel kod (Unicode) desteği sayesinde çoklu dil desteği bulunmaktadır. Ancak şu anda 28 dili desteklemekte olup, Türkçe desteği bulunmamaktadır. Metin ayrıştırma yöntemi olarak metin önce sayılar, noktalama işaretleri, semboller ve kelimelerin farklı tipte kullanımları gibi belirteçlerine (token) ayrılıp sonlu durum dönüştürücü (finite state transducer) aracılığıyla ayrıştırma işlemi gerçekleştirilmektedir. Bu yöntemin Türkçe diline uyarlanması yerine gerçekleştirimini kendimizin yaptığı metin ayrıştırma yöntemin kullanılmasına karar verilmiştir.

Metinlerden insan kaynaklarıyla ilgili yetenek terimleri ve iş tanımları etiketleri çıkarmada DBpedia Spotlight (Bizer et al., 2011) kullanılabilir. Bu araç, metinlerde geçen DBpedia kaynaklarının belirlenmesini (etiketlenmesini) gerçekleştirmektedir. Ancak çalışma alanımız Türkçe dili ağırlıklı olduğundan ve

DBpedia kaynaklarının çalışma alanımızda henüz yetersiz olduğundan başarı oranı düşük olacaktır.

Bir mikroblog sitesi olan Twitter'dan anlık olarak alınan metinler üzerine Schulz et al. (2013) tarafından yapılan çalışmada anlık olarak toplanan tvitler (tweet) üzerinden acil durumları, araba kazalarını belirlemede yüksek başarı oranı elde edilmiştir. Metinlerin işlenmesinde kullanılan yöntemde, metinler (tvitler) bir takım ön işlemlerden geçirilip daha sonra yukarıda bahsedilen DBpedia Spotlight ve benzeri bilgi çıkarıcı araçlar birlikte kullanılarak istenilen bilgilere erişilmektedir. Ancak bu çalışmada, makine öğrenimi algortimaları kullanılıp, üzerinde çalışılacak olan alanla ilgili aracın eğitilmesi gerekmektedir. Bu çalışmada DBpedia Spotlight kullanımı ve metne uygulanan önişlemler bizim için farklı ve güzel bir bakış açışı sunmaktadır.

Sonuç olarak, metinlerden bilgi çıkarımı ve etiketlenmesi alanında çeşitli çalışmalar yapılmış olup, çeşitli araçlar bulunmaktadır. Ancak bu çalışmalarda ve araçlarda ya Türkçe dili için gerekli destek bulunmamakta ya da gerçekleştirilen sistem üzerinde çalışılan veriden ve alandan bağımsız olarak tüm alanlarda başarılı bir şekilde çalışmamaktadır. Geliştirilen metin etiketleyici araç ile bu eksikliklerin giderilmesi ve birden fazla dilden kelimeler içeren (Türkçe ve İngilizce yetenek terimleri, iş tanımları vb.) metinlerde de başarılı sonuçlar alınması hedeflenmiştir.

2. ANLAMSAL AĞ VE WEB İÇERİKLERİ

Web içeriği insanlar için devasa bir bilgi kaynağıdır. Web üzerinde yer alan kaynaklardan aranılan her şeye, istenilen her türlü bilgiye ulaşmak mümkündür. Gelişen teknoloji ile birlikte bilgisayarlar hayatımızın her alanında yer almaya başladı. İnsanların kolayca anlayabildiği, ancak bilgisayarların anlamlandırmada güçlük çektikleri web içeriğini bilgisayarın da anlayabileceği bir biçime getirmek için World Wide Web Consortium (W3C) tarafından anlamsal ağ (semantic web) genişletmesi tanımlanmıştır.

Anlamsal ağ ile web içerikleri belirli modellerle tanımlanmaktadır. Web içeriklerinin anlamsallaştırılması web içeriği sağlayacılar ve kullanıcılar tarafından yapılabilmektedir. Ancak tüm web içeriğinin kısa sürede anlamsallaştırılması mümkün olmadığı için var olan web içeriklerinde bulunan veriler işlenerek anlamlı hale getirilebilir.

Web içeriklerinde bulunan veriler yapısal, yarı yapısal ve yapısal olmayan hallerde bulunabilir. Yapısal web içerik verisi daha önceden tanımlanmış bir veri modeline uygun içeriklerdir. Yarı yapısal veri ise belirli bir veri modeline uygun veri içermemesine rağmen veri içerisinde anlamsal etiketler içeren, verinin bazı kısımlarını tanımlayan etiketlere sahip olan veri içeriğidir. Yapısal olmayan veriler ise herhangi bir veri modeline sahip olmamakla beraber herhangi bir anlamsal etiketlemeye de sahip olmayan veri içerikleridir.

Yapısal olan verilerde sahip oldukları beri modeli uygun bir üstveri modeliyle eşlenerek anlamsal ağ tanımlamalarına kolayca uygun hale getirelebilir. Belirli bir veri modeline göre yapılanmış olmaları bu imkanı sağlamaktadır. Ancak yarı yapısal ve yapısal olmayan verilerde böyle bir imkan bulunmamaktadır. Bu yüzden yarı yapısal ve yapısal verileri ön bir işlemden geçirmek ve bu doğal dildeki verileri ayrıştırmak gerekmektedir.

Anlamsal ağ ile yarı yapısal ve yapısal olmayan web içerik verilerinin işlenerek anlamsallaştırılması ile ilgili ayrıntı bilgiler takip eden bölümlerde verilmiştir.

2.1. Anlamsal Ağ

Genel Ağ'ın (*World Wide Web*) mucidi Tim Berners-Lee 2001 yılında web içeriklerinin bilgisayarların da anlayabileceği bir biçime getirilmesi için Anlamsal Ağ kavramını ortaya atmıştır. Berners-Lee et al. yayınladıkları makalede (2001) mevcut ağın anlamsal ağa nasıl dönüşeceğini tanımlamış, yapılması gerekenleri, kullanılması gereken biçimleri belirtmişlerdir. Ancak, anlamsal ağ tanımlamalarının ilk halinin gerçekleştirilmesinin çok zor olduğu fark edilerek revize edilmiştir. Berners-Lee et al. The Semantic Web Revisited (2006) isimli

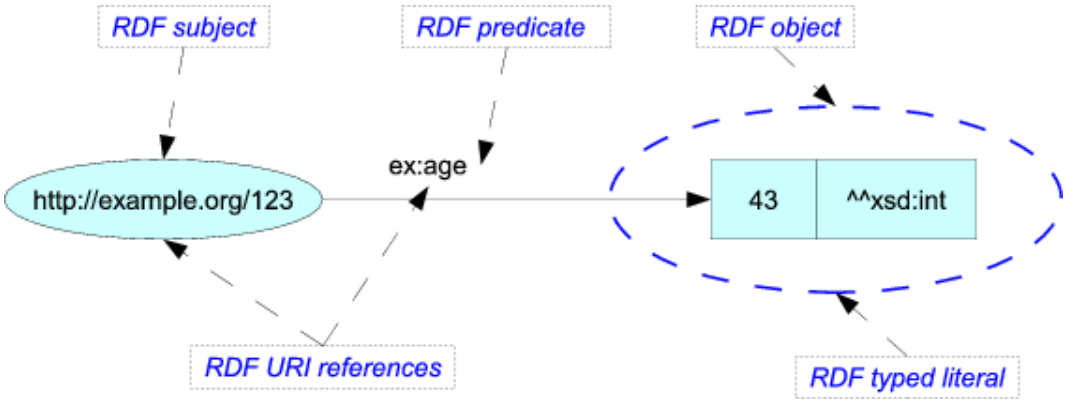
çalışmalarında anlamsal ağ tanımlamalarını gerçekleştirilebilir bir şekilde yeniden tanımlamışlardır.

Anlamsal ağdan Web 3.0 olarak da bahsedilmektedir. Web 1.0 internetin ilk dönemidir. Web 1.0 döneminde kullanıcılar sadece internetteki içerikleri erişir ve onları okuyabilir. İnternetin gelişimiyle kullanıcıların da internet içeriğine katkıda bulunması, yorumlar ekleyebilmesi gibi sosyal bir internete yönelim ile Web 2.0 dönemine geçilmiştir. Web'in günümüzde kullanılmakta olan şu anki versiyonu Web 2.0 olarak tanımlanmaktadır. Web 2.0 döneminde en popüler internet siteleri sosyal ağ siteleridir. Web 2.0 içeriğinin üstverilerle modellenerek bilgisayarların anlayabileceği hale getirilmesi ise anlamsal ağ, yani internetin Web 3.0 versiyonudur. Anlamsal ağ ile ilgili bazı kavramlardan ve kullanılan yapılardan devam eden bölümde bahsedilmiştir.

Kaynak Tanımlama Çerçevesi (Resource Description Framework – RDF)

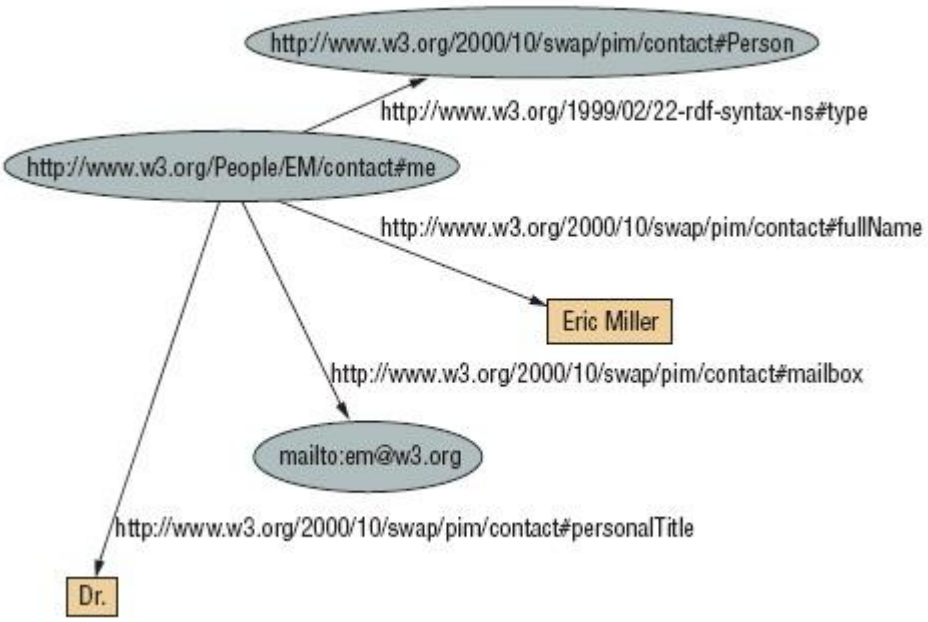
Kaynak Tanımlama Çerçevesi (RDF), özelleştirilmiş Evrensel Kaynak Tanımlayıcı (*Universal Resource Identifier - URI)*'lardır. Anlamsal ağ tanımlamasında verilerin saklanması ve gösterilmesi için kullanılan üçlü tabanlı güçlü bir gösterim dilidir. İlk olarak W3C tarafından 1997 yılında tanımlanmıştır. Daha sonra 1999 yılında WC3 tarafından önerilmeye başlanmıştır. İlk olarak üstveri veri modelleri için tanımlanmıştır. Daha sonra anlamsal ağ ile birlikte web kaynaklarının modellenmesinde kullanılmaya başlanmıştır. RDF kullanımı ve gösterimi ile ilgili kuralları içeren *RDF Primer*'lerin en sonucusu *RDF Primer 1.1* 2014 yılında yayınlanmıştır. Her yayınlanan yeni *RDF Primer*'de RDF yapısının kullanımı ile ilgili geliştirmelerle beraber RDF yapılarının kullanılmasının kolaylaştırılması hedeflenmektedir. Daha kolay kullanım sunulması RDF yapılarının kullanımın ve anlamsal ağın yaygınlaşmasına katkı sağlamaktadır.

RDF üçlüleri doğal dil cümle yapısına benzer şekilde özne (RDF subject), yüklem (RDF predicate) ve nesne'den (RDF object) oluşur. Yüklemler (predicate) RDF üçlülerinde özellik (property) olarak isimlendirilmektedir. Şekil 2.1'de gösterildiği üzere özne üçlüde tanımlanacak olan varlığın kaynağını, özellik (yüklem) üçlüde tanımlanan varlığın gösterilen verisinin türünü (özne ile nesne ilişkisini), nesne ise saklanan verinin değerini (RDF typed literal) gösterir.



Şekil 2.1 Bir RDF üçlüsü (W3C'den)

Şekil 2.1'de gösterilen RDF çizgesinde,"*http://www.w3.org/People/EM/contact#me*" URI'yla tanımlanan kişinin tam adı Eric Miller, e-posta adresi em@w3.org'tur. Bu kişinin ünvanının ise "Dr" olduğu tanımlanmaktadır. Bu çizgede tanımlanan verinin tipinin kişi olduğu "*http://www.w3.org/2000/10/swap/pim/contact#Person*" URI'yla, kişi olduğunu belirten özellik "*http://www.w3.org/1999/02/22-rdf-syntax-ns#type*" (rdf:type) URI'yla,e-posta adresini belirten üçlünün özelliği (property) "*http://www.w3.org/2000/10/swap/pim/contact#mailbox*" URI'yla,kişinin ünvanını belirten üçlünün özelliği "*http://www.w3.org/2000/10/swap/pim/contact#personalTitle*" URI'yla, kişinin tam adını belirten üçlünün özelliği ise "*http://www.w3.org/2000/10/swap/pim/contact#fullName*" URI'yla tanımlanmıştır.



Şekil 2.2 - Bir RDF çizgesi (Berners-Lee'den 2006)

RDF çizgeleri basit bir şekilde XML biçimine dönüştürülerek XML biçiminde de gösterilebilir. Şekil 2.2'de gösterilen RDF çizgesinin XML formatında gösterimi aşağıdaki gibidir:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
        <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <contact:fullName>Eric Miller</contact:fullName>
        <contact:mailbox rdf:resource="mailto:em@w3.org"/>
        <contact:personalTitle>Dr.</contact:personalTitle>
        </contact:Person>
</rdf:RDF>
```

RDF'lerin tanımlı sınıf (class) ve özellikleri (property) bulunmaktadır. Kullanılabilecek tüm RDF sınıf ve özellikler Çizelge 2.1'de açıklanarak sıralanmıştır.

Sınıflar:

rdf:XMLLiteral - *XMLliteral*değerlerin sınıfı
rdf:Property–özelliklerin sınıfı
rdf:Statement - *RDFstatement*'ların (bir RDF üçlüsü) sınıfı
rdf:Alt, rdf:Bag, rdf:Seq–alternatif kapsayıcılar, sıralanmamış kapsayıcılar vesıralı kapsayıcılar (rdfs:Container bu üçünün üst sınıfıdır)
rdf:List - RDF Listeleri sınıfı
rdf:nil - bir rdf:List oluşumu olup boş listeyi temsil eder

Özellikler

rdf:type - rdf:Property'nin bir örneği olup ilgili kaynağın hangi sınıfın bir örneği olduğunu belirtir
rdf:first– RDF üçlüleri listesindeki ilk nesne
rdf:rest–RDF üçlüleri listesindeki rdf:first'ten sonraki nesneler
rdf:value–yapısal değerler için kullanılır
rdf:subject–RDF üçlüsünün öznesi
rdf:predicate–RDF üçlüsünün yüklemi
rdf:object–RDF üçlüsünün nesnesi

Çizelge 2.1 – RDF sınıf ve özellikleri

Resource Description Framework Schema (RFDS)

RDF Şema (Resource Description Framework Schema - RFDS), isminden de anlaşıldığı üzere RDF verilerinin veri şemalarını tanımlamak için kullanılan bir

gösterim dilidir. Kullanılan RDFS sınıf ve özellikleri Çizelge 2.2'de açıklanmaktadır.

Sınıflar

rdfs:Resource–kaynak sınıfı, her şeyi temsil eder
rdfs:Literal - karakter değerleri sınıfı(örnek: tam sayılar,karakter (string) dizileri)
rdfs:Class–sınıfların sınıfı
rdfs:Datatype - RDF veri tiplerinin sınıfı
rdfs:Container - RDF kapsayıcılarının sınıfı
rdfs:ContainerMembershipProperty–kapsayıcı üyelik özelliklerinin sınıfı, rdf:_1, rdf:_2, ... ve diğerleri rdfs:member'ın bir alt özelliğidir.

Özellikler

rdfs:subClassOf –kaynağın başka bir sınıfın alt sınıfı olduğunu belirtir
rdfs:subPropertyOf– kaynağın başka bir özelliğin alt özelliği olduğunu belirtir
rdfs:domain–kaynağın ana özelliği
rdfs:range - kaynağın alt özelliği
rdfs:label–kaynağın insanlar tarafından okunabilir ismi
rdfs:comment–bir kaynağın açıklaması
rdfs:member–kaynağın bir üyesi
rdfs:seeAlso–bir kaynak hakkında daha fazla bilgi
rdfs:isDefinedBy–bir kaynağın tanımlaması

Çizelge 2.2 – RDFS sınıf ve özellikleri

Evrensel Kaynak Tanımlayıcı (Universal Resource Identifier – URI)

Evrensel Kaynak Tanımlayıcı'lar (Universal Resource Identifier – URI) kaynakları tanımlamaktadırlar. Bu yüzden anlamsal ağ çalışmalarının merkezindedir. Anlamsal ağı oluşturan verilerin çoğu ilişkisel veritabanlarında tutulmaktadır. Bu verilerin anlamsal ağa aktarılması için o veritabanının nesnelerinin URI sistemi içerisinde tanımlanması gerekmektedir.

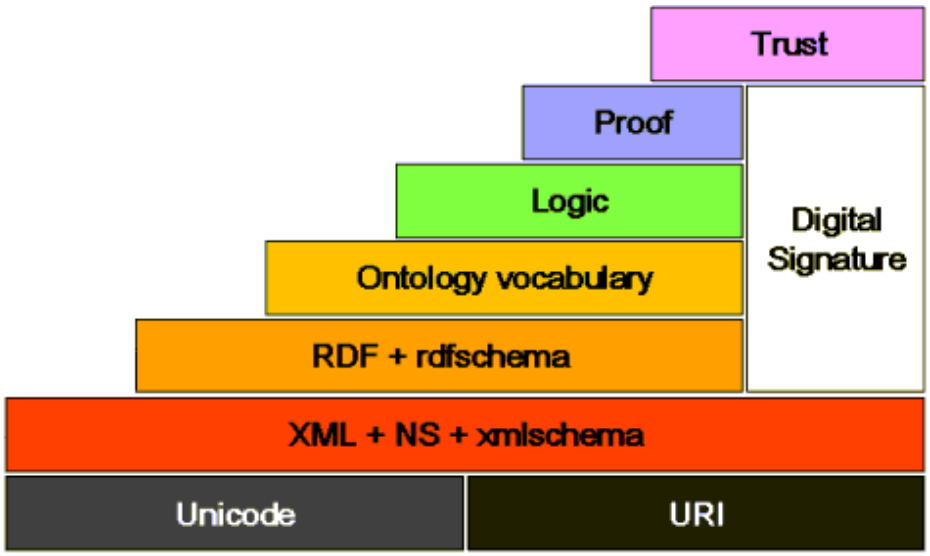
URI'lar küresel bir kapsama sahiptir. Bir kaynağı bir URI ile kimliklendirmek o kaynağı herhangi birinin referans edebileceği, o kaynağa erişebileceği anlamına gelir.

Anlamsal ağın temel yapıtaşları Şekil 2.3'teki anlamsal ağ kulesinde gösterilmektedir. Anlamsal ağ üzerinde çalışan kişiler çalışmalarını bu anlamsal ağ kulesindeki katmanlar üzerine kurmaktadır. Tez çalışması kapsamındaki anlamsal ağ çalışmaları RDF katmanı temel alınarak yapılmıştır.

Üçlü Depoları

Anlamsal verilerin, RDF üçlülerinin, bir veri deposunda tutulması gerekmektedir. Bunun için çeşitli üçlü depoları geliştirilmiştir.

Tez çalışmalarında Jena kütüphanesinin sağlamakta olduğu TDB isimli üçlü deposu kullanılmıştır.



Şekil 2.3 – Anlamsal Ağ kulesi (W3C'den)

Simple Knowledge Organization System (SKOS)

Simple Knowledge Organization System (SKOS), yapısal kontrollü sözlüklerin (Bilgi Organizasyon Sistemleri (Knowledge Organizastion System – KOS)) anlamsal ağ ile ilişkilendirmesinde kullanılan ortak bir veri modelidir.

SKOS konseptlerinin tamamı Çizelge 2.3'te listelenmiştir.

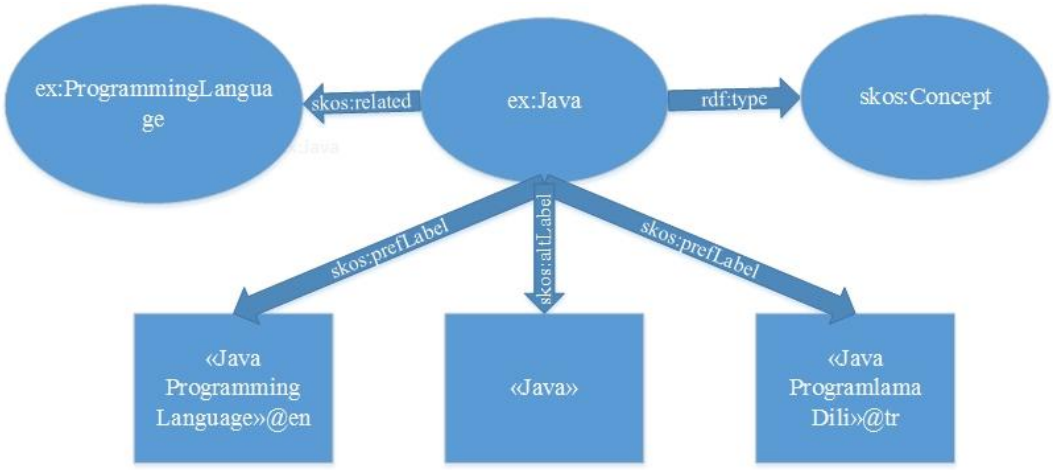
skos:Concept	skos:broader
skos:ConceptScheme	skos:broaderTransitive
skos:inScheme	skos:narrower
skos:hasTopConcept	skos:narrowerTransitive
skos:topConceptOf	skos:related
skos:altLabel	skos:semanticRelation
skos:hiddenLabel	skos:Collection
skos:prefLabel	skos:OrderedCollection
skos:notation	skos:member
skos:changeNote	skos:memberList
skos:definition	skos:broadMatch
skos:editorialNote	skos:closeMatch
skos:example	skos:exactMatch
skos:historyNote	skos:mappingRelation
skos:note	skos:narrowMatch
skos:scopeNote	skos:relatedMatch

Çizelge 2.3 – SKOS konseptleri

SKOS konseptleriyle tanımlanmış "Java programlama dili" kaynağı Çizelge 2.4'te ve Şekil 2.4'te gösterilmektedir.

```
ex:Java rdf:type skos:Concept;
 skos:prefLabel "Java Programlama Dili"@tr;
 skos:prefLabel "Java Programming Language"@en;
 skos:altLabel "Java" ;
 skos:narrower ex:JSF ;
 skos:related ex:ProgrammingLanguage .
```

Çizelge 2.4 – SKOS konseptleriyle tanımlanmış "Java" örneği



Şekil 2.4 - SKOS konseptleriyle tanımlanmış "Java" örneği

DBpedia

Vikipedi'nin bağlı veri formatında yapısallaştırılmış versiyonudur. Bizer et al. (2007) tarafından geliştirilmiştir. DBpedia diğer açık veri kümelerine de bağlantılar içerir. Böylece büyük bir bağlı veri kümesi web'de sunulmaktadır.

2.2. Yapısal Olmayan ve Yarı Yapısal Web İçerikleri

Yapısal olmayan veriler sınıflandırılmamış, bir veri modeline sahip olmayan verilerdir. Web sayfaları, fotoğraflar, videolar, word ve pdf dokümanları, e-posta'lar yapısal olmayan veri içeren şeylerden bazılarıdır.

Yarı yapısal veri yapısal veriler ve yapısal olmayan veriler arasında yer alan veri türüdür. Aslında bir yapısal veri tipidir ancak belirli bir veri modeli yapısına sahip değildir. Yarı yapısal veride veri içerisindeki bazı kısımlar etiketlerle veya başka tip belirticilerle tanımlanmaktadır ancak veri belirli bir yapıya sahip değildir.

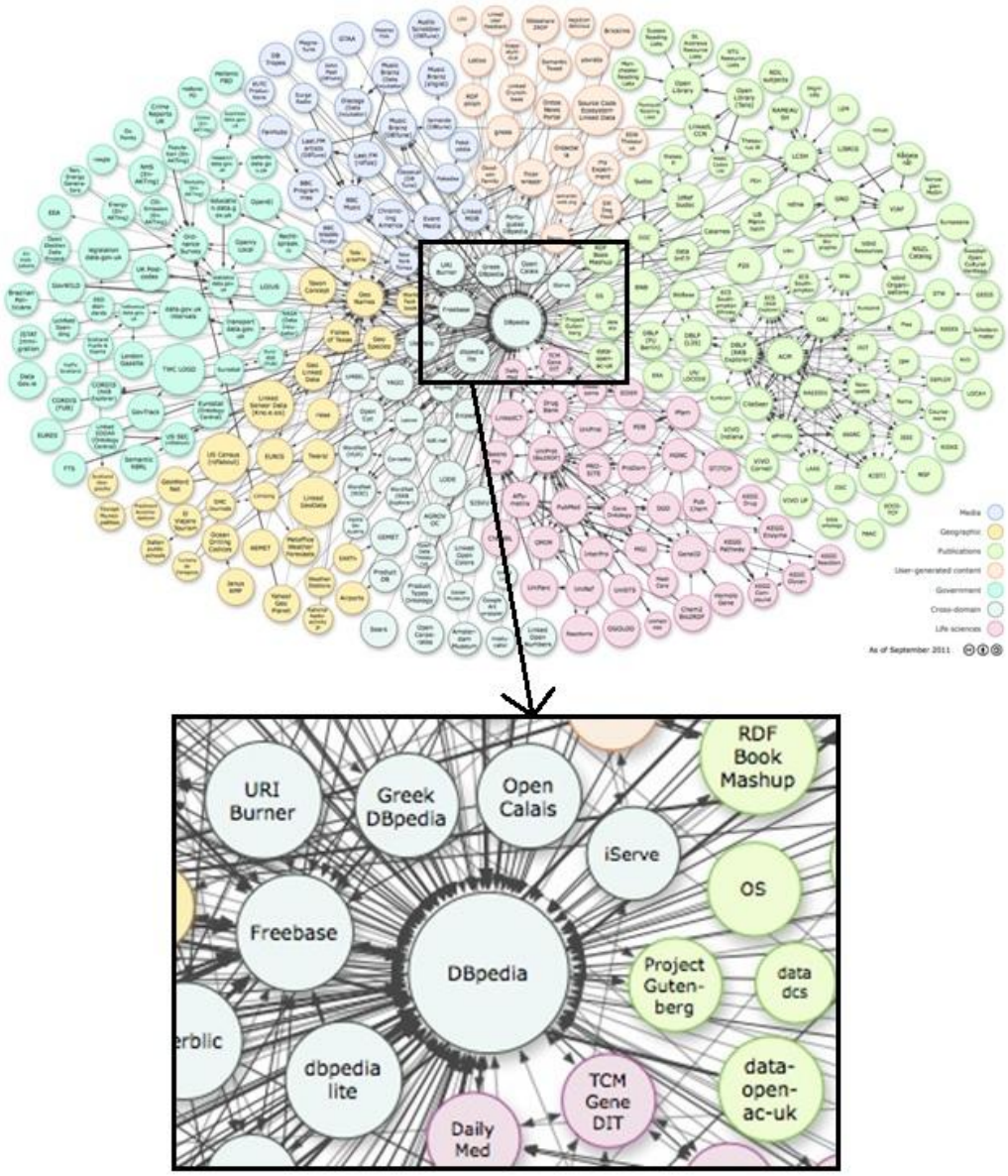
Yarı yapısal verilere örnek olarak, sözcük işlemci programlarda yapısal olmayan metinleri içeren bir dokümanın yazarının adı ve oluşturulduğu tarih gibi üstveri bilgilerini içerebiliyor olması verilebilir. E-postalar gönderen kişi bilgisinin, alıcı bilgisinin, gönderilme tarihinin, gönderilme saatinin ve diğer düzenlenmiş yapısal alanların yapısal olmayan e-posta metnine eklenmesiyle oluşmaktadır. Fotoğraflar ve diğer grafikler oluşturan, tarih, lokasyon bilgisi ve anahtar kelimeler gibi şeylerle etiketlenebilmektedir. Örnek olarak verilen bu yarı yapısal içerikleri yönetmek için XML ve diğer işaretleme dilleri kullanılmaktadır. Tüm bu yapısal olmayan ve yarı yapısal içerikler web içeriklerinin büyük bir kısmını oluşturmaktadır.

3.VERİ DEPOLAMA ALTYAPILARI

3.1. Bağlı Veri

Bağlı veri (linked data), verilerin ve veri kaynaklarının anlamsal olarak birbiriyle bağlanmasıyla oluşan veri kümeleridir. Bu bağlı verilerden oluşan veri kümeleri anlamsal ağın temelini oluşturmaktadır. Bağlı veri kaynaklarını tanımlamak için URI'lar kullanılır. Bağlı veriler RDF üçlülerinden oluşur. N-TRIPLE, Turtle, JSON gibi farklı biçimlerde gösterilebilirler. Bağlı verilerin saklandıkları veritabanlarından sorgulanması SPARQL sorgu diliyle (SPARQL Protocol and RDF Query Language) yapılır.

Bağlı veri yapısında olan açık veri kümelerinin DBpedia merkezli olarak oluşturduğu bağlı veri bulutu Şekil 3.1'de gösterilmektedir.



Şekil 3.1 –Bağlı açık veri bulutu

Bağlı veri üçlülerini saklamak için çeşitli üçlü depolayıcılar (triplestore) bulunmaktadır. Bu üçlü depolayıcıların en çok kullanılanları aşağıda listelenmektedir (parantez içindeki bilgiler üçlü depolayıcıları geliştiren kişi veya firma isimleridir):

- Virtuoso (OpenLink Software)
- Fuseki (Apache)
- TDB (Apache)
- Oracle Spatial 11g (Oracle)
- OWLim (Ontotext)
- BigData (Systap)
- rdfstore-js (Antonio Garrote)
- 4store (Garlik)
- Sesame (Aduna)
- RDFStore (Alberto Reggiori)

Çalışmalarda bu seçenekler arasından Apache'nin Jena kütüphanesi kapsamında sunmakta olduğu TDB RDF üçlüsü depolayıcısının kullanılmıştır. Bağlı veri çalışmaları kapsamında kullanılacak olan Jena kütüphanesi tarafından tamamen desteklenmesi ve yüksek performansı en önemli avantajlarından olmaktadır.

Jena kapsamında yer alıp kullanılacak olan üçlü depolayıcı TDB ile ilgili ayrıntılar şunlardır:

TDB

Geliştirim Dili: Java
Platform: Windows, Linux ve Mac OS X
Lisans: Apache License 2.0

Veri yapısı

Veriler RDF üçlüleri olarak tutulmaktadır. RDF üçlülerinin yapısından önceki bölümlerde bahsedilmektedir.

İndeksleme

RDF üçlülerinin bulunduğu düğüm tablosu her zaman önbellekte tutulmaktadır. Bu sayede veriye hızlı erişim sağlanmaktadır.

Transaction yapısı

TDB'de *transaction* desteği sunulmaktadır. Transaction'lar ACID prensiplerini desteklemektedir.

Sunucu mimarisi

Verilerin tutulduğu mimariyi oluşturan yapılar şunlardır:

- Düğüm (node) tablosu: RDF terimlerinin gösterimlerini tutar. Node ve NodeId yapıları bulunmaktadır. NodeId'ler 8 byte'lık tanımlama verileridir. Node'lar ise 128 bitlik yapılardır ve MD5 ile şifrelenmektedir.

- Üçlü (triples) ve dörtlü (quads) indeksleri: Üçlülerin ve dörtlülerin indeklerini tutar. Üçlüler RDF üçlülerinin indeklerini, dörtlüler ise isimli çizgeler için kullanılmaktadır.

- Önek (prefix) tablosu: RDF verilerinde kullanılan önekleri tutar. Önekleri RDF/XML veya Turtle biçimde tutmaktadır.

Sunucu erişimleri

TDB içerisinde dahili bir HTTP REST arayüzü yer almamaktadır.

Sorgulama Dili

TDB'de sorgulama dili tüm RDF üçlülerinde olduğu gibi SPARQL'dır. Özel bir sorgulama dili sunulmamaktadır.

"Snatch" filminin eklenmesi

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>

INSERT DATA { <http://examplemovie/snatch> dc:title "Snatch" ; dc:director
"Guy Ritchie" . }
```

"Reservoir Dogs" isimli filmin silinmesi

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>

DELETE DATA { <http://examplemovie/reservoirdogs> dc:title "Reservoir
Dogs" ; dc:director " Quentin Tarantino" . }
```

Güncelleme (Önce eski bilgi silinip sonra yeni bilgi eklenir)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

WITH <http://example/contactlist>
DELETE { ?person foaf:givenName 'Enes' }
INSERT { ?person foaf:givenName 'Mehmet' }
WHERE { ?person foaf:givenName 'Enes' }
```

3.2. Melez Veri

Melez veri (hybrid data), farklı veri depolama altyapılarının bir arada kullanılmasıyla oluşan melez veri altyapısına denir. Bu altyapı oluşturulurken, farklı veri altyapılarının kullanım durumuna göre avantajlı yönleri dikkate alınarak seçimlerin yapılması önemlidir. Seçilen veri depolama altyapılarının etkin bir şekilde bir araya getirilmesi ile melez veri altyapısı sağlanmış olmaktadır. Tek bir veri depolama altyapısının kullanılması birden çok veri depolama altyapısının bir arada kullanılmasından basit ve daha az karmaşıktır. Ancak verinin kullanım durumuna göre en uygun veri depolama altyapılarının seçilmesi ve kullanılması etkinliği arttırır.

Melez veri ile ilgili daha eski çalışmalar olsa da en kapsamlısı Fowler and Sadalege (2012) tarafından yapılmıştır. Fowler and Sadalege Polyglot Persistence ismini verdikleri veri depolama yaklaşımında melez veri kullanımı savunmakla birlikte NoSQL veritabanlarının kullanımının yaygınlaşacağını savunmaktadırlar. Bu çalışmada NoSQL veritabanlarının sağlamakta oldukları özelliklerle SQL veritabanlarına birçok alanda üstünlük kurduklarından bahsedilmektedir. Ancak SQL veritabanlarının raporlama gibi halen üstün oldukları alanlar olduğu da kabul edilmektedir. Polyglot Persistence yaklaşımı ile ilgili ayrıntılı bilgi takip eden başlıkta verilmektedir.

3.2.1. Polyglot Persistence yaklaşımı

Fowler and Sadalege (2012), "NoSQL Distelled: A Brief Guide to the Emerging World of Polyglot Persistence" isimli kitaplarında geliştirilecek veya varolan uygulamanın ihtiyaçlarına göre veri depolama altyapılarının seçilmesi gerektiğini vurgulamaktadırlar.

NoSQL veritabanlarını ayrıntılarıyla bilmek hangi veri depolama altyapılarının seçileceğine karar verme aşamasında doğru teknolojilerin seçilmesi açısından önemlidir. NoSQL veritabanı teknolojileri ilerleyen bölümlerde daha ayrıntılı anlatılmaktadır.

NoSQL veritabanları üç ana kategoride incelenebilir. Bunlar:

- Doküman veritabanları
- Çizge veritabanları
- Anahtar-değer veritabanları

Doküman veritabanları:

Belirli bir şemaya sahip olmayan yarı yapısal veritabanlarıdır. Sağlamış oldukları koleksiyon yapılarında farklı veri modellere ait veriler de tutulabilmektedir. En yaygın kullanılanı MongoDB'dir. Sunmakta olduğu ayrıntılı

dokümantasyonu ve desteği, esnek yapısı ve yüksek performansı yüzünden tercih edilmektedir.

En yaygın kullanlan doküman veritabanı olan MongoDB ile ilgili ayrıntılar şöyledir:

MongoDB

Geliştirim Dili: C++
Platform: Windows, Linux ve Mac OS X
Lisans: GPL v3

MongoDB en popüler doküman veritabanlarından biridir. Kullanıcı topluluğu, desteği ve dokümantasyon kaynakları ile diğer rakiplerine göre ön plana çıkmaktadır.

Veri yapısı

MongoDB sadece doküman veri modelini sunmaktadır. Bu yüzden veri yapısı sadece koleksiyonlardan oluşmaktadır. Koleksiyonlar içerisindeki dokümanlar BSON (Binary JSON) formatında saklanmaktadır. Her bir doküman en çok 12K boyutunda olabilmektedir. Daha büyük dokümanlar için GridFS isimli teknolojinin kullanılması gerekmektedir. Bu durumda dokümanlar her biri en çok 255K olabilen parçalara bölünüp saklanabilmektedir.

MongoDB şema bağımsız bir yapı sunmaktadır. Diğer bir deyişle bir koleksiyon içerisinde farklı sahalara sahip kayıtlar yer alabilir.

Tüm dokümanlarda veritabanı tarafından otomatik olarak tanımlanmış _id alanı yer almaktadır. 12 byte uzunluğunda hexadecimal bir sayı içeren bu alan her dokümanın tekilliğini garanti etmektedir. Bu değerin kayıt sırasında kullanıcı tarafından da atanabilmesi mümkündür.

MongoDB içerisindeki dokümanların birbirleri ile ilişkilendirilmesi mümkündür. Bunu sağlayabilmek için iki yaklaşım sunulmuştur:

- Gömülü yaklaşım: Bu yaklaşımda bir doküman başka dokümanları doğrudan içerebilmektedir. Böylelikle işlemler çok daha hızlı gerçekleştirilebilmektedir.

- Referans yaklaşımı: Bu yaklaşımda referans edilen diğer dokümanların _id değerleri saklanmaktadır. Ancak _id değeri olan bir dokümana ulaşabilmek için ayrı bir veritabanı erişimi daha gerçekleştirmek gereklidir. Bu da performans sorunlarına neden olabilmektedir.

İndeksleme

Bir doküman içerisindeki tüm sahalar, hangi seviyede olduğuna bakılmaksızın indekslenebilmektedir.

Transaction yapısı

MongoDB *transaction* için bir destek sunmamaktadır. Ancak bir doküman üzerinde gerçekleşen tüm işlemler atomik yapıdadır.

Sunucu mimarisi

MongoDB, master-slave modunda bir dağıtıklık sunmaktadır. Bir veritabanı birincil (primary) veritabanıdır ve tüm yazma işlemleri bu veritabanı üzerinden gerçekleşir. Birincil veritabanı yazma işlemlerinin kayıt bilgisini oplog içerisinde saklar. İkincil veritabanları buradaki değişiklikleri algılayarak kendi verilerini de güncellemektedir.Birincil veritabanında bir hata oluşması durumda ikincil veritabanlarından biri birincil olarak atanabilir.

MongoDB, yatay olarak dağıtıklık (sharding) da sağlayabilmektedir. Bu durumda aynı koleksiyona ait veriler birden fazla sunucuya dağıtılabilmektedir.

Sunucu erişimleri

MongoDB içerisinde dahili bir HTTP REST arayüzü yer almamaktadır. Bunun yerine MongoDB için geliştirilmiş 3. parti REST arayüzleri bulunabilmektedi. Bazı MongoDB REST sunucuları aşağıda listelenmiştir:

- RESTHeart
- DrowsyDromedary
- Crest

Sorgulama Dili

MongoDB özel olarak isimlendirilmiş bir sorgulama dili sunmamaktadır. Tüm CRUD işlemleri koleksiyonlar üzerinde çalıştırılabilen fonksiyonlar tarafından işletilmektedir. MongoDB map/reduce desteği de sunmaktadır. Aşağıda bu işlemlerle ilgili bazı örneklere yer verilmiştir. Bu örneklerde "skills" isimli bir koleksiyona doküman örneği kaydedilmesi, doküman sorgulanması ve güncellenmesi ile MongoDB map/reduce desteği ile ilgili fonksiyonlar yer almaktadır.

Doküman örneğinin kaydedilmesi

```
db.skills.insert({docId:"Skill1",name:"Test Driven Development",status:1})
```

Dokümanların sorgulanması

```
db.skills.find({name:"Test Driven Development"})
```

Dokümanların güncellenmesi

```
db.skills.update({status:1},{$set:{status:2}})
```

Map/Reduce

```
db.documents.mapReduce(
        function(){ emit(this.status,this.docId); },
        function(key,values) { return {status:key, docIds:values}; }
)
```

Çizge veritabanları:

En yaygın kullanılanı Neo4j'dir. İlişkiler içeren verilerin saklanması için idealdir. Verilerin öneri için kullanıldığı uygulama parçalarında kullanılması uygundur.

En yaygın kullanlan çizge veritabanı olan Neo4j ile ilgili ayrıntılar şöyledir:

Neo4J

Geliştirim Dili: Java
Platform: Windows, Linux ve Mac OS X
Lisans: GPL v3

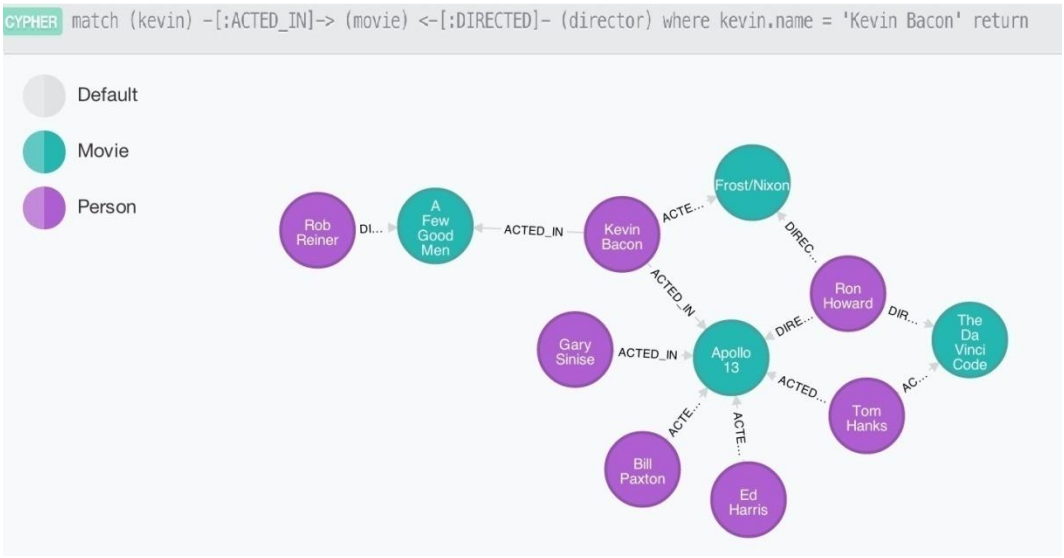
Neo4j en çok bilenen ve en yaygın olarak çizge veritabanıdır. Bu nedenle destek, kullanıcı topluluğu ve dokümantasyon açısından diğer çizge veritabanlarına göre daha önde gözükmektedir.

Veri Yapısı

Neo4j bir çizge veritabanıdır ve özellik tabanlı çizgeleri desteklemektedir. Bu çizgelerde hem tepeler hem de ayrıtlar çeşitli tiplerdeki değerlerin saklanabildiği ek sahalara sahip olabilmektedir.

Neo4j tepelere etiket adı verilen özel indekslerin de atanabilmesi sağlamaktadır. Böylelikle tepeler tiplerine göre ayrılabilmekte ve daha etkin bir filtreleme sağlanabilmektedir. Ayrıca tepeler arasındaki ayrıtlar da kendi tipine

sahip olabilmektedir. Neo4j tarafından saklanan ve gösterilen çizge modelinin bir örneği Şekil 3.2'de gösterilmektedir.



Şekil 3.2 - Neo4j çizge sorgusu ve yapısı

Transaction Yapısı

Neo4J çizge işlemlerinde *transaction* desteği sunulmaktadır. Transaction'lar ACID prensiplerini desteklemektedir.

Tepeler ve ayrıtlar üzerinde yapılan yazma işlemlerinde ilgili tepe ve ayrıtlar kilitlenmekte ve diğer işlemler tarafından erişilememektedir.

İndeksleme

Tüm tepe ve ayrıt sahaları üzerinde indeksleme yapılabilmektedir. Bunun yanı sıra metinsel alanların tam metin indeksleri ile indekslenebilmesi de mümkündür. Tam metin indeskleri için Lucene bileşeni gömülü olarak kullanılmaktadır.

Sunucu Mimarisi

HA (High Availability) adı verilen teknoloji hata toleranslı bir veritabanı mimarisi oluşturulmasını ve yatay olarak ölçekleme yapılabilmesini sağlamaktadır.

Neo4J HA modunda çalışırken her zaman tek bir *master* veritabanı olmaktadır. Bu mimaride birden fazla *slave* veritabanı kullanılabilmektedir. Tüm *slave* veritabanları *master* veritabanı içerisindeki güncel veri ile senkronize olmaktadır. Bunun yanı sıra *master* veritabanının *slave* veritabanlarını hemen güncelleyebileceği bir yapılandırma da seçilebilir. *Master* veritabanı işlem sırasında seçilen *slave* veritabanlarını güncellemeye çalışacaktır. Bu güncelleme

işlemi başarısız olsa dahi *master* veritabanı üzerindeki işlem başarı ile tamamlanacaktır.

Sunucu Yapısı

Neo4J gömülü mod ve sunucu modu olmak üzere iki farklı yapıda kullanılabilmektedir. Gömülü mod Java projelerinde kullanılabilmektedir. Bu durumda Neo4j sunucusu Java sürecinin doğrudan içerisinde çalışmakta ve aynı belleği kullanmaktadır. Sunucu modunda ise Neo4J veritabanının ayrı bir sunucu olarak kullanılması durumu söz konusudur.

Sunucu modu ile çalışırken HTTP REST arayüzü üzerinden erişim sağlanabilmektedir.

Veri Erişim İşlemleri

Neo4j çizge işlemleri için *Cypher* sorgu dili adı verilen özel bir dil sunmaktadır. Bu dil üzerinden tüm CRUD işlemleri gerçekleştirilebilmektedir. Aşağıda bazı *Cypher* sorguları örneklenmiştir:

"Skill" etiketine sahip bir tepenin oluşturulması

```
CREATE (s:Skill {"SkillId":"S1245"})
```

"Skill" ve "Resume" etiketli iki tepe arasında "hasSkill" ayrıtının oluşturulması

```
MATCH (r:Resume {"ResumeId":"R1234"})
MATCH (s:Skill {"SkillId ":"S1245"})
CREATE (r)-[:HASSKILL]->(s)
```

Bir yeteneği (Skill) içeren özgeçmişin (Resume) getirilmesi

```
MATCH (s:Skill {"SkillId ":"S1245"})
MATCH (r:Resume)-[:HASSKILL]->(s)
RETURN r
```

Bir özgeçmişin içerdiği tüm yeteneklerin listelenmesi

```
MATCH (r:Resume {"ResumeId":"R1234"})
MATCH r-[:HASSKILL*]->(s:Skill)
RETURN s
```

Anahtar-değer veritabanları:

Redis en yaygın olarak kullanılan anahtar-değer veritabanlarından biridir. Hızlı erişim gereken verilerin saklanması için ideal bir veritabanı türüdür. Verileri saklamak için genel olarak sistem belleğini kullanırlar. Ayrıca sağlamış oldukları etkin indeksleme ile istenilen veriye hızlıca erişim sağlamaktadırlar.

En yaygın kullanlan anahtar-değer veritabanı olan Redis ile ilgili ayrıntılar şöyledir:

Redis

Geliştirim Dili: C
Platform: Linux, Windows[1], Mac OS X
Lisans: BSD

Veri Yapısı

Redis, bir anahtar-değer veritabanıdır. Dolayısıyla oldukça basit bir veri modeline sahiptir. Ancak bu basitlik çok hızlı olmasını sağlamaktadır.

Tüm kayıtlar bu kaydı tekil olarak tanımlayan bir anahtar ve bu anahtarın bir değeri ile saklanmaktadır. Anahtar alanı sadece *String* değerler içerebilir iken değer alanı için farklı alternatifler bulunmaktadır. Değer alanının saklayabileceği veri tipleri aşağıda özetlenmiştir:

- String: Metin içerikli verilerdir
- Hash: Anahtar ve değer listeleridir.
- List: Değer listeleridir.
- Set: Her bir değerin yalnızca bir kez yer alabildiği listelerdir.
- Sorted Set: Set veri yapısına benzemektedir. Ek olarak her bir değerin "score" adı verilen bir sıralama değeri de saklanmaktadır. Set içindeki değerleri score değerine göre sıralanmaktadır.
- Bitmap: Sadece 1 ve 0 değerlerini içeren listelerdir.

Transaction yapısı

Redis işlemleri çeşitli komutlar üzerinden yürütülmektedir. Birden fazla komutun tek bir *transaction* içerisinde çalıştırılmasını sağlayan basit bir mimari bulunmaktadır.

[1] Windows için bir kurulum paketi sunulmamıştır. Ancak Microsoft Open Technologies tarafından Windows sistemler için derlenmiş ve hazırlanmış bir Redis paketi bulunmaktadır.

İndeksleme

Redis, tüm indekslemeyi anahtar saha üzerinden gerçekleştirmektedir. Bir kayda anahtar değer ile erişmenin maliyeti O(1)'dir. Doğrudan anahtar değer üzerinden olmayan erişimler ise O(n) karmaşıklığına sahip olacaktır.

Sunucu mimarisi

Redis sunucuları *master-slave* mimarisinde çalışabilmektedir. Bu mimaride bir *master* sunucu ve birden fazla *slave* sunucu yer alabilmektedir. Kopyalama işlemleri hem *master* hem de *slave* sunucularda herhangi bir bloklamaya neden olmamaktadır. Bu işlemler sırasında gelen sorgulamalar eski veri seti üzerinden cevaplanabilmektedir.

Veri Erişim İşlemleri

Redis özel bir veri erişim dili sunmamaktadır. Veri yapısı çok basit olduğu için işlemler de basit komutlar üzerinden yürütülmektedir. Aşağıda bazı örnek komutlara yer verilmiştir:

Bir anahtar-değer çiftinin eklenmesi

```
SET key1 value1
```

Anahtarı bilinen bir kayıta ait değerin getirilmesi

```
GET key1
```

Değer alanı liste olan bir anahtar-değer çiftinin eklenmesi

```
RPUSH list1 A
RPUSH list1 B
RPUSH list1 C
```

Değer alanı *hash* olan bir anahtar-değer çiftinin eklenmesi

```
HMSET key key1 value1 key2 value2 key3 value3
```

3.2.2. Melez veri yaklaşımı için gerekli olan veri depolama mimarileri

Kullanılacak veri depolama altyapılarının seçiminde uygulamanın kullanacağı veri türlerine uygun veri depolama altyapılarının seçilmesi ve veriye erişim performansının göz önünde bulundurulması önemlidir.

Çizge veri depolama altyapıları, birçok ilişkiye sahip bir veriyi saklamak için diğer veri depolama altyapılarından daha uygun depolama ve sorgulama yetenekleri sunmaktadır. Öneri sistemlerinde ilişkili verilerin bulunması ve öneri olarak geri dönülmesi açısında da kullanışlıdır.

Tüm NoSQL veri depolama altyapıları sundukları esnek veri saklama yetenekleri ile esneklik gereken durumlarda veriyi en uygun şekilde saklama imkanı sunmaktadır. Bir melez veri depolama altyapısı oluşturmak için gerekli temel NoSQL çözümleri olan MongoDB, Neo4j ve Redis'in özellikleri Çizelge 3.1'de listelenmektedir.

	MongoDB	Neo4J	Redis
Veritabanı Modeli	Doküman	Çizge	Anahtar/ Değer
Geliştirim Dili	C++	Java	C
Platform	Windows, Linux, Mac OS X	Windows, Linux, Mac OS X	Windows, Linux, Mac OS X
Depolama	Doküman	Çizge	Ana Bellek (Anahtar-değer)
Şema bağımsız	Evet	Evet	Hayır
Transaction	Atomik işlemler	ACID	-
Sorgulama	CRUD fonksiyonları	Cypher ve Gremlin	Basit ekleme ve sorgulama işlemleri
Map/Reduce	Var	Yok	Yok
Kopyalama (Replication) yaklaşımı	Master-Slave	Master-Slave	Master-Slave
Dağıtıklık	Var	Yok	Var
HTTP API	Yok	Var	Yok
Lisans	GPL v3	GPL v3	BSD

Çizelge 3.1 – Bir melez veri altyapısı için temel NoSQL çözümleri

Bir melez veri altyapısı oluşturmak için farklı doküman, çizge ve anahtar-değer veritabanları alternatifleri bulunmaktadır. Önde gelen NoSQL veritabanları MongoDB, Neo4J ve Redis harici CouchDB, ArangoDB, OrientDB ve Titan gibi alternatiflerin özellikleri Çizelge 3.2'te gösterilmiştir.

	CouchDB	ArangoDB	OrientDB	Titan
Veritabanı Modeli	Doküman	Doküman+ Çizge+ Anahtar-değer	Doküman+ Çizge	Çizge
Geliştirim Dili	Erlang	C++	Java	Java
Platform	Windows, Linux, Mac OS X	Windows, Linux, Mac OS X	Windows, Linux, Mac OS X	Windows, Linux, Mac OS X
Depolama	Doküman	Doküman	Doküman	Cassandra, HBase, BerkeleyDB
Şema bağımsız	Evet	Evet	Evet	Evet
Transaction	ACID	ACID ve MVVC	ACID	ACID ve Eventual Consistency
Sorgulama	Map/Reduce	AQL	SQL	Gremlin
Map/Reduce	Var	Yok	Yok	Var
Kopyalama (Replication) yaklaşımı	Master-slave	Master-Slave	Master-Master	Kullanılan veritabanına göre değişiklik gösterir
Dağıtıklık	Var	Var	Var	Var
HTTP API	Var	Var	Var	Var
Lisans	Apache License 2.0	Apache License 2.0	Apache License 2.0	Apache License 2.0

Çizelge 3.2 – Diğer NoSQL çözümlerinin özellikleri

4. DOĞAL DİL İŞLEME

Doğal Dil İşleme (Natural Language Processing – NLP), yapay zeka ve dilbiliminin alt kategorisi olan çok disiplinli bir çalışma alanıdır. Bu çalışma alanında, tüm doğal dillerin işlenmesi ve kullanılmasıyla ilgili çalışmalar yürütülmektedir. Bilişim alanındaki önemli gelişmeler, Doğal Dil İşleme'deki araştırma alanlarını popüler hale getirdiğinden zaman içerisinde tek başına değerlendirilmeye başlanan bir disiplin olmuştur.

İnsana özgü bir özellik olan dil yeteneğinin bir anlamda bilgisayarlara da kazandırılması hedeflenmektedir. Bu amaç doğrultusunda Doğal Dil İşleme doğal dillerin bilgisayar tarafından üretilmesi ve anlaşılması problemleriyle ilgilenir. İletişim ve artan devasa boyuttaki verilerin işlenmesi ve anlaşılması için dilin bilgisayar ortamında modeli oluşturulması önemli bir aracın geliştirilmesine aracı olacaktır.

DDİ'nin ana görevi doğal bir dili çözümlemek, anlamak, yorumlamak ve üretmek olan bilgisayar sistemlerini tasarlanmak ve gerçekleştirmektir. DDİ'nin araştırma alanları doğal dillerin yapısının daha iyi anlaşılması, bilgisayarlar ile insanlar arasındaki arabirim olarak doğal dili kullanmak ve bilgisayar ile dil çevirisi yapmaktır. Bu alanda yapılmış tüm çalışmalar bilim ve iş alanında geçerli bir dil olan İngilizce için geliştirilmiştir. İngilizce için kullanılan kurallar ve algoritmaların birebir Türkçe'ye çevrilmesi dillerin biçimbilimsel farklılığından dolayı pek mümkün değildir. Bu yüzden Türkçe için yapılacak olan çalışmalar Türk dilbilimcileri veya Türkçe'yi çok iyi bilen yerli veya yabancı bilgisayar mühendisleri tarafından yapılabilir.

Doğal dil işlemenin uygulama alanlarından biri ise projenin ana amaçlarından biri olan kelime işlemcileridir. Kelime işlemcileri veritabanına SQL vb. sorgu dilleri yerine doğal dil kullanarak sorgu yöneltmek ve bunu çözümleyerek veritabanı üzerinde gerekli sorguyu yaparak sonucu kullanıcıya iletmeyi hedeflemektedir.

Doğal dil işleme çalışmalarının diğer hedef alanları bir dilden başka bir dile yarı otomatik metin çevirisi yapmak, tek veya çok dilli sözcüklere erişmek, doğal dilde cümle ve metin üretmek, metin özetlemek, konuşma tanıma ve konuşma üretmek olarak sıralanmaktadır.

Tez çalışmasında doğal dil işleme çalışmaları kapsamında biçimbirimsel (morfolojik) çözümlemeye ihtiyaç duyulmuştur. Biçimbirimsel çözümleme için Türkçe dilinde biçimbirimsel çözümleme yapan açık kaynak kodlu Zemberek-NLP kütüphanesi kullanılmıştır. Zemberek-NLP kütüphanesi kullanılarak geliştirilen biçimbirimsel çözümleyici araç ile Türkçe metinler işlenmektedir. İşlenen metinler gövdelenerek sonraki aşamalarda ayrıntılı anlatılacak olan etiketleme işlemleri gerçekleştirilmektedir.

Varlık İsmi Tanımlama (Named Entity Recognition – NER)

Varlık İsmi Tanımlama (Named Entity Recognition – NER) daha önceden oluşturulmuş bir veri havuzunda yer alan kavramların metinler içerisinde tanımlanması işlemidir. Örnek olarak, şirket isimlerinden oluşan bir veri havuzu kullanılarak metinler içerisinde geçen şirketlerin tanımlanması işlemi varlık ismi tanımlamadır.

4.1. Doğal Dil İşleme Araçları ve Kütüphaneleri

Doğal dil işleme kütüphanelerinin büyük bir çoğunluğu İngilizce dili için geliştirilmiştir. Farklı yabancı diller için çeşitli kütüphaneler bulunsa da Türkçe dili için açık kaynak gelişmiş bir doğal dil işleme kütüphanesi olarak kısıtlı kütüphaneler bulunmaktadır. Kayda değer Türkçe doğal dil işleme kütüphaneleri şunlardır:

TRmorph

TRmorph kütüphanesi Çöltekin (2010) tarafından C yazılım dili kullanılarak geliştirilmiş. İki katmanlı bir biçimbirimsel çözümleme imkanı sunmaktadır. Zemberek-NLP kütüphanesine göre biçimbirimsel çözümleme açısından farklı bir yöntem izliyor olsa da başarı oranı beklenenden düşük olduğu gözlenmiştir.

Zemberek-NLP

Zemberek kütüphanesi ile ilgili çalışmalar daha eskiye dayansa da kütüphaneyle ilgili makale 2007 yılında yayınlanmıştır (Akın ve Akın). Geliştirilmesi devam edilen Zemberek kütüphanesi 2014 senesi başı itibariyle TRNLTK kütüphanesinden bazı çözümleyici parçalarının da eklenmesiyle Zemberek-NLP adı ile geliştirilmeye başlanmıştır. Önceki sürümlerine göre ciddi değişiklikler içeren Zemberek-NLP kütüphanesi henüz tam stabil olmasa da yapılan testler ve incelemeler sonucunda tez kapsamında kullanılması uygun görülmüştür. Ancak, Akın Mart 2015'te Zemberek-NLP kütüphanesinin geliştiriminin durdurulduğunu açıklamıştır. İlerleyen süreçlerde Zemberek-NLP kütüphanesinin yeni ve stabil sürümleri çıkarsa eğer bu sürümü geliştirilen uygulamaya entegre edilebilecek şekilde tasarım gerçekleştirilmiştir.

Zemberek-NLP kütüphanesi varolan Türkçe dili üzerinde çalışan doğal dil işleme araçları arasında en başarılı biçimbirimsel çözümleme yapan araç olduğu gözlenmiştir. Bu sayede geliştirilmesi hedeflenen doğal dil üzerinden arama yapmaya olanak sağlayacak sistemin başarı oranı artacaktır. Buna rağmen Zemberek-NLP'nin bahsedildiği üzere stabil bir sürümünün henüz yayınlanmamış olması nedeniyle bazı biçimbirimsel analiz ve diğer doğal dil işleme özelliklerinde

bazı eksiklikler mevcuttur. Bu eksiklikler ilk aşamada önemli bir soruna neden olmasa da ileriki aşamalarda geliştirilmesi zorunlu olan kısımlar tarafımızdan proje kapsamında geliştirilecektir.

TRNLTK

Zemberek-NLP kütüphanesine alternatifi olarak Ok (2012) tarafından geliştirilmiş bir kütüphanedir. Sadece biçimbirimsel çözümleyici yapısından oluştuğu için biçimbirimsel çözümleyicisi Zemberek-NLP kütüphanesine göre çok az problem çözmektedir. Ancak Zemberek-NLP kütüphanesine göre iyi yönleri vardır. Python yazılım dili ile geliştirme yapmak isteyen kullanıcıların faydalanması için proje ilk olarak Python dilinde yayınlanmıştır. Ancak daha sonra Zemberek-NLP kütüphanesinden iyi olan yönlerinden Zemberek-NLP kütüphanesinin gelişmiş yapısı kullanılarak faydalanılması için Java yazılım diline çevrilerek Zemberek-NLP kütüphanesi altına taşınmıştır. Bu yüzden yeni bir sürümü çıkmayacağı gibi yapılacak olan geliştirmeler Zemberek-NLP adı altında yayınlanacaktır. Yetersiz olan var olan sürümü üzerinde herhangi bir geliştirme yapılmayacağı için kullanılması tercih edilmemiştir.

Araç	Yazılım Dili	Kelime Analizi	Cümle Analizi	Lisans
TRmorph	C	Var	Yok	GNU LGPL
Zemberek-NLP	Java	Var	Var	Apache V2.0
TRNLTK	Python & Java	Var	Yok	Apache V2.0

Çizelge 4.1 – Doğal dil işleme kütüphaneleri

Türkçe doğal dil işleme kütüphanelerinin sağladıkları imkanlar doğrultusunda karşılaştırması Çizelge 4.1'de gösterilmiştir.

4.2. Türkçe Biçimbirimsel Çözümleme

Biçimbirimsel (morfolojik) çözümleme sürecinde dilin yapı özellikleri incelenir. Biçimbirimsel çözümlemenin üç temel özelliği vardır. Bunlar;

- Kelimelerin kökleri araştırılır ve kelimelerin türleri belirlenir.
- Kelimelerin almış olduğu ekler araştırılır. Ekler tespit edildiğinde kelime grupları belirlenebilir.
- Eklerin türünü araştırır; aynı ek farklı görevlerde kullanılabildiğinden cümleye farklı anlamlar katabilir.

Türkçe dili için biçimbirimsel çözümleme yapılırken önce kelimenin kökü aranır. Bunun için Türkçe dilindeki bütün kelime köklerinin tutulduğu bir sözlük oluşturulur. Oluşturulan kök sözlüğünde aranacak olan kelime bulunduğu haliyle

aranır. Bunun için kelimenin uzunluğu ile aynı uzunluğa sahip olan kelimeler arasında arama yapılır. Kelime bu kökler arasında bulunursa ek almamış olarak belirlenir ve kök halinde olduğu belirtilir. Ancak kelimenin kökü bulunamadıysa kelimenin sağından bir harf atılarak yeniden yöntemle arama yapılır. Kelimenin kökü bulunana kadar aynı işlem devam ettirilir. Kök bulunduğunda ise, kelimenin sağ tarafından atılan kısımlar ekler sözlüğünde aranılarak anlamlı bir ek oluşturup oluşturmadığına bakılır. Eğer atılan kısım anlamlı bir ek oluşturmuyorsa kelimenin kökünü aramaya devam edilir.

Kök ve ekler aranırken Türkçe dil kuralları gereği olabilecek ünlü uyuşması, hece düşmesi gibi dil bilgisi kuralları göz önünde bulundurulmalıdır. Oluşturulan arama algoritmaları bu kurallara dikkat edilerek oluşturulmalıdır.

Türkçe dili sondan eklemeli bir dil olarak her bir ekin kelimeye eklenme sırası ve dizilişi vardır. Ayrıca bazı ekler kelimenin aldığı eklerle bulunduğu duruma göre eklenebilme durumları vardır. Türkçe dilinde biçimbirimsel çözümleme yapılırken bu durumlar göz önünde bulundurulmalıdır.

Biçimbirimsel çözümlemeye ait bazı örnekler Çizelge 4.2'de verilmiştir.

Enstitü	
Kök: enstitü Lemma: enstitü Tür: İsim (Noun)	Çözümleme: [(enstitü:enstitü) (Noun;A3sg+Pnon+Nom)]
Özgeçmişler	
Kök: özgeçmiş Lemma: özgeçmiş Tür: İsim (Noun)	Çözümleme: [(özgeçmiş:özgeçmiş) (Noun;A3pl:ler+Pnon+Nom)]
Gitmek	
Kök: git Lemma: gitmek Tür: İsim (Noun)	Çözümleme: [(gitmek:git) (Verb;Pos) (Noun;Inf1:mek+A3sg+Pnon+Nom)]
Kedi	
Kök: kedi Lemma: kedi Tür: İsim (Noun)	Çözümleme: [(kedi:kedi) (Noun;A3sg+Pnon+Nom)]
Geliyormuş	
Kök: gel Lemma: gelmek Tür: Fiil (Verb)	Çözümleme: [(gelmek:gel) (Verb;Pos+Prog:iyor+Past:du+A3sg)]

Çizelge 4.2 – Biçimbirimsel çözümleme örnekleri

4.3. Türkçe Metin Etiketleyici Araç

Türkçe metin etiketleyici araç geliştirilmeden önce Türkçe dilinde çalışan doğal dil işleme araçları araştırılmıştır. Bu kapsamda Türkçe dilinde doğal dil işleme ve analiz çalışmalarında kullanılabilecek önde gelen 3 kütüphane incelenmiştir. Bu kütüphaneler Zemberek-NLP, TRmorph ve TRNLTK kütüphaneleridir. Bu kütüphanelerden diğerlerine göre daha başarılı biçimbirimsel çözümleme kabiliyeti, daha köklü bir çalışma olması ve güçlü geliştirme ekibine sahip olması ile ileriki süreçler için gelişime açık olması nedenleriyle Zemberek-NLP kütüphanesinin projemiz kapsamındaki doğal dil işleme çalışmalarında kullanılmasına karar verilmiştir.

Doğal dil ayrıştırıcılarına gerekli yazı verisini sağlayabilmek için gerekli olan dokümanların içeriklerini elde etmeye yarayan araçlar araştırılmıştır. Bu araçlar arasında Apache desteği olması ve sistemde yer alan dokümanların tek bir tipte saklanmıyor oluşundan dolayı farklı tipteki dokümanlar üzerinde başarıyla çalışan Apache Tika kütüphanesinin üstünde duruldu. Apache Tika kütüphanesi kullanılarak geliştirilen doküman içeriği çıkarıcı yapı doğal dil işleme kısmıyla birleştirilerek birlikte çalışmaları sağlandı. Böylece özgeçmişlerin içerikleri elde edilerek doğal dil ayrıştırıcı yapı ile analiz edilmesi ve bu analiz sonuçlarının sonraki adımlarda kullanılmasına olanak sağlayan bütünleşik bir yapı geliştirilmiştir.

Tez kapsamındaki doğal dil işleme çalışmalarında Zemberek-NLP kütüphanesinin kullanılması tercih edilmiştir. Yapılan Zemberek-NLP, TRmorph, TRNLTK araçları ile yapılan analiz testleri sonucunda projeye en uygun kütüphanenin Zemberek-NLP olduğuna karar verilmiştir. Zemberek-NLP kütüphanesinin yazım denetleme fonksiyonunun daha güçlü olduğu görülmüştür. Ayrıca şu an için çok verimli çalışmasa da hatalı yazımlarla ilgili düzeltmeler için bir öneriler sistemi Zemberek-NLP kütüphanesinde yer almaktadır.

Zemberek-NLP kütüphanesinin de yetersiz kaldığı yerlerde veya projedeki doğal dil işleme çalışmaları kapsamında gerçekleştirilmesi gereken fonksiyonlar olduğunda gerekli eklemeler ve geliştirilmeler Java yazılım dilinde yapılacaktır.

Doğal dil işleme ile ilgili kısım farklı teknoloji ve kütüphanelerin kullanılmasına olanak veren, kolayca sisteme entegre edilebilen bir şekilde geliştirilmiştir. Bu sayede, tez projesi sonrasında da çalışmalar devam ettiği sürece var olan doğal dil işleme ve biçimbirimsel çözümleme araçlarındaki gelişmeler, yenilikler takip edilecek ve gerektiği zaman farklı kütüphaneler veya var olan kütüphanenin geliştirilmiş versiyonu sisteme entegre edilecektir.

4.3.1. Doğal dil ayrıştırıcısının geliştirilmesi

Zemberek-NLP kütüphanesi kullanılarak geliştirilen doğal dil ayrıştırıcı araç (TextAnalyzer) biçimbirimsel (morfolojik) çözümleme yaparak kelimeleri çözümler. Bir metnin çözümlenme ve analiz süreci Şekil 4.1'de gösterilmiştir. Çözümlenen kelimelerin ilk olarak Türkçe etkisiz kelimeler[2] arasında yer alıp almadığı kontrol edilir. Eğer kelime etkisiz kelimeler arasındaysa dolar işareti ($) ile işaretlenir. Etkisiz kelimeler arasında yer almayan kelimeler için analiz işlemi devam eder.

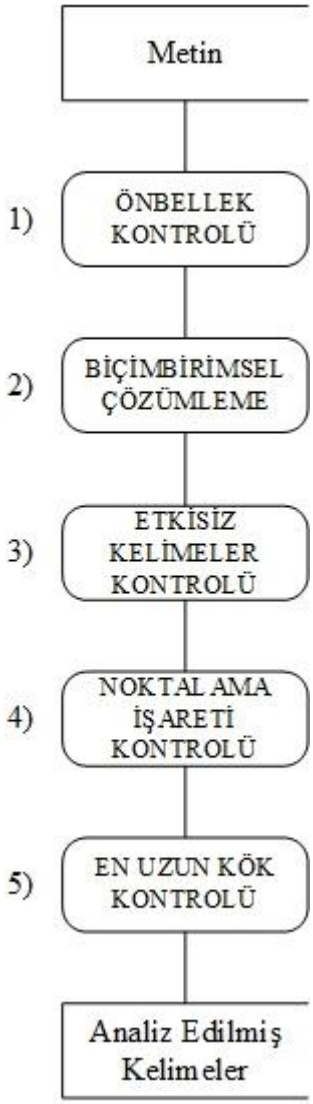
Kelimenin biçimbirimsel çözümlemesi sonuçlarından kelimenin türü kontrol edilir. Eğer bir noktalama işaretiyse kelime dolar işareti ile işaretlenir. Kelime bu kontrolü de geçerse biçimbirimsel çözümleme sonuçlarından kelimenin kökü ve eklerle ilgili analiz sonuçları alınarak bu bilgileri içeren bir kelime nesnesi oluşturulmaktadır. Eğer bir kelimenin çözümleme sonucunda birden fazla kök alternatifi varsa bu alternatifler arasından en uzun olanı, hiç kökü yoksa kelimenin kendisi kök olarak kabul edilmektedir. Kelime nesnesi bir listeye kaydedilmeden önce o metine ait ana kadar kaydedilmiş olan kelimeler arasında aynı kelime olup olmadığı kontrol edilir. Eğer aynı kelime daha önce kaydedildiyse farklı bir kayıt tutulmamaktadır.

Analizi tamamlanan kelimeler bir doküman içerik nesnesinde tutulmaktadır. Bu nesnede her kelimenin metin içerisinde bulunduğu konumu (indeksi) bir anahtar-değer listesinde tutulur. Bu bilgiler daha sonra etiketleme işlemleri sırasında kullanılır.

Analiz sonucunda doğal dil ayrıştırıcı araca girdi olarak gelen metindeki her kelime ile o kelimenin analiz sonuçları önbelleğe kaydedilmektedir. Daha sonra analiz için gelen metinlerdeki kelimeler ilk olarak önbellekten kontrol edilmekte, eğer aynı metnin analizi daha önce yapıldıysa direkt olarak önbellekteki sonuç döndürülmektedir. Eğer önbellekte ilgili kelime mevcut değilse kelimenin analizi yapılmaktadır.

Zemberek-NLP'nin sağlamış olduğu cümle ayrıştırma desteği sayesinde uzun metinler herhangi bir ön cümle ayrıştırma işlemine sokulmadan işlenebilmektedir.

[2] Tek başına bir anlam ifade etmeyen "neden", "çünkü", "şey" gibi bağlaçlar ve kelimeler etkisiz kelime olarak tanımlanmaktadır.



Şekil 4.1 - Metin analiz süreci

İlk aşamada, geliştirilen metin etiketleyici aracın başarı oranı beklenenden düşük çıkmıştır. İlk sürümünde yaklaşık olarak %72 başarı oranına sahip metin etiketleyici aracın başarı oranını arttırmak için bazı iyileştirmeler yapılması gerekmiştir. Metin etiketleyici araç üzerinde yapılan başlıca iyileştirme çalışmaları şunlardır:

Türkçe metin etiketleyici araç üzerinde iyileştirme çalışmaları:

Geliştirilen Türkçe metin etiketleyici aracın başarı oranı beklenenden düşük çıkması üzerine çalışma algoritması üzerinde bazı iyileştirme çalışmaları yapılmıştır. Bu sayede metin etiketleyici aracın hedeflenen başarı yüzdesi aralığında (%80-%92) bir başarı oranıyla çalışması sağlanmıştır.

Yapılan iyileştirmelerden ilki, özgeçmişlerin metin içerikleri işlenirken fiil türündeki kelimeler elenmiştir. Bu sayede herhangi bir fiil içermeyen yetenek veri havuzundaki kaynaklarla eşsesli veya aynı köke sahip olan kelimelerin hatalı

olarak etiketlenmesinin önüne geçilmiştir. Diğer iyileştirme ise, eşsesli veya aynı köke sahip olan kelimeleri içeren birden fazla veri kaynağı mevcut ise, bu kaynaklarla diğer etiketlenmiş veri kaynakları arasında uzaklık ölçümü yapılmakta ve kaynaklar arası uzaklığın daha fazla olduğu veri kaynağı elenmektedir.

4.3.2. Etiketleme işlemi

Metinlerin etiketlenmesi süreci Şekil 4.2'de gösterilmiştir. Doğal dil işleyici araç ile analiz edilen kelimeler doküman içerik nesnesi içinde anahtar-değer listesi halinde etiketleme işlemi için metin etiketleyici araca gönderilir. Her kelime için yetenek verilerinin tutulduğu anahtar-değer veritabanından sorgulama yapılır. İlgili kelimeyi içeren yetenek verileri ayrı bir listeye eklenir.

Bulunan yetenek verilerinin doğru yetenek verileri olduğundan emin olmak için bu veriler metinsel içerik üzerinde kontrol edilir. Doğru etiketlemeler kişiyle veya kişinin özgeçmiş dokümanıyla ilişkilendirilir.

Şekil 4.2 - Metin etiketleme süreci

4.3.3. Doküman içerik çıkartıcının geliştirilmesi

İşlenmek üzere girdi oluşturacak olan veri sadece internet içeriklerinden değil, özgeçmişleri içeren dokümanlardan da elde edilecektir. Bu doğrultuda dokümanlarda yer alan metinsel verilerin çıkartılmasında kullanılacak bir araç geliştirilmesi ihtiyacı doğmuştur. Özgeçmiş dokümanlarının farklı dosya türü uzantılarına sahip olması beklenmektedir. Bu yüzden her türlü dosya uzantısıyla sorunsuz çalışabilen bir doküman içerik çıkartıcı geliştirilmiştir. Doküman içerik çıkartıcının geliştirilmesinde Apache Tika aracı kullanılmıştır.

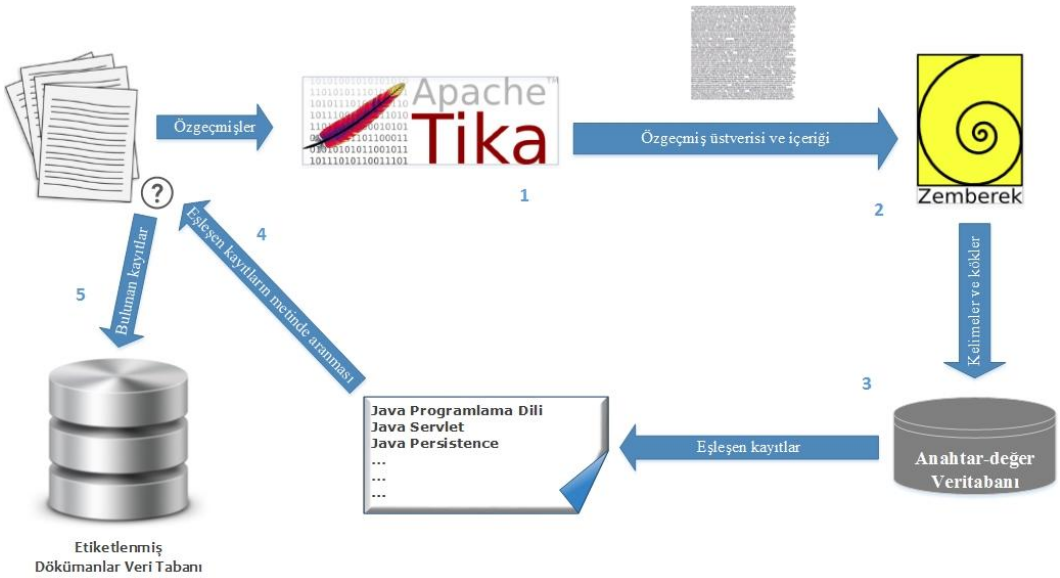
Apache Tika

Tez kapsamında sisteme girdi olacak özgeçmiş dokümanların içeriklerini ve üst verilerini elde etmek için Apache Tika aracı kullanılmaktadır. Apache Tika aracı birçok formattaki doküman içeriğini elde edebilmektedir. Böylece sistemde yer alan farklı tipteki dokümanların içeriklerinin elde edilmesine olanak sağlanmış olmaktadır.

Özgeçmişlerin içeriğini elde etmek amaçlı yapılan geliştirmede bir Java uygulama programlama arayüzü (API) geliştirilmiştir. Bu uygulama programlama arayüzü sayesinde sadece bu sistem parçacığına içeriği ve üst verisi elde edilmek istenen dosyanın dosya yolunun belirtilmesi yeterli olmaktadır. Geliştirilen API geriye dosyanın (özgeçmişin) içeriğini ve üst veri bilgilerini döndürmektedir.

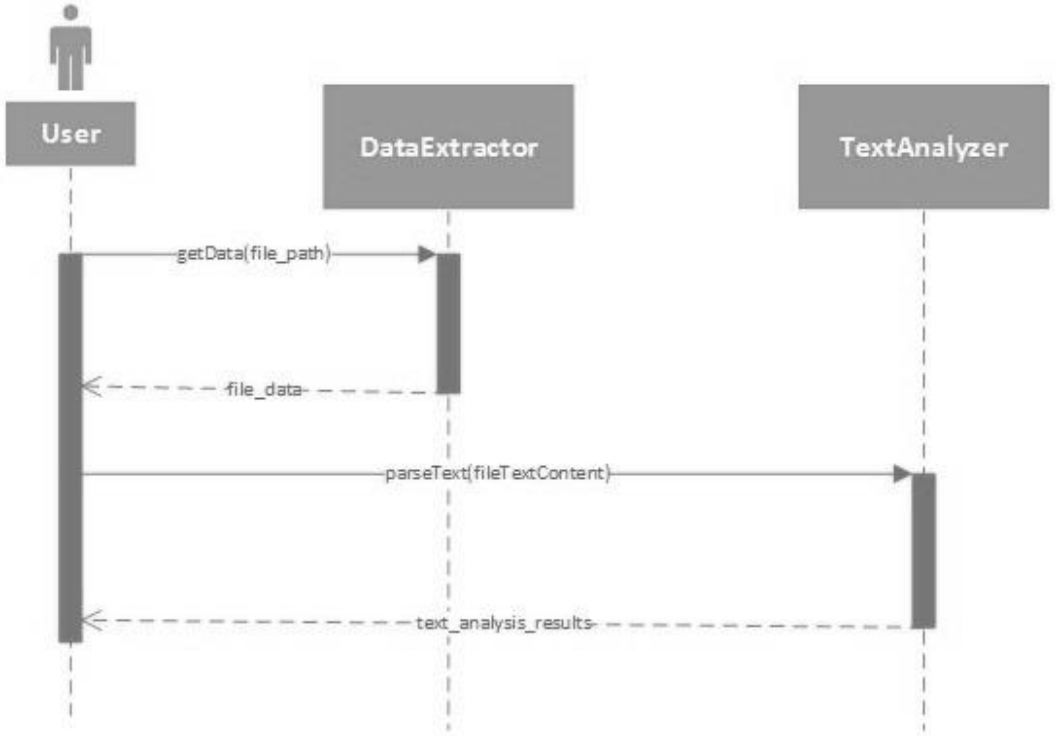
4.3.4. Doğal dil işleme sürecinin bütünleştirilmesi

Özgeçmişlerin niteliklerinin ve ilgili oldukları yeteneklerin belirlenmesi süreci Şekil 4.3'de gösterilmiştir. Özgeçmişlerin ilk olarak Apache Tika aracı ile içeriklerinin çıkarılması gerçekleştirilmektedir. Daha sonra bu çıkarılan içeriklerin analiz edilmesi ile kelime ve kelime köklerine ayrıştırılması Zemberek kütüphanesi kullanılarak geliştirilen biçimbirimsel çözümleme aracıyla gerçekleşmektedir. Elde edilen kelimeler var olan anahtar-değer veritabanında aranmaktadır. Her kelime için anahtar-değer veritabanında eşleşen kayıtlar tekrar özgeçmiş içeriğinde aranıp doküman içerisinde geçen kayıtlar var ise bu kayıtlar özgeçmişlerle ilişkilendirilerek ayrı bir doküman veritabanı koleksiyonunda özgeçmişlerle birlikte saklanır.



Şekil 4.3 – Metinlerin analiz edilme sürecinin bütünleştirilmesi

Özgeçmiş içeriği çıkartıcı araç ve doğal dil ayrıştırıcı araçların geliştiriminde uygulanan mimari Şekil 4.4'de gösterilmiştir. Buna göre, kullanıcının sisteme analiz edilmesi istenen özgeçmişin dosya yolunu belirtmesi yeterli olmaktadır. Belirtilen dosya yolu üzerinden önce geliştirilirken Apache Tika aracının kullanıldığı özgeçmiş içerik çıkarıcı araç (DataExtractor) özgeçmişin metinsel verisi ve üst veri bilgilerini elde etmekte, daha sonra bu metinsel veri geliştiriminde Zemberek-NLP kütüphanesinin kullanıldığı doğal dil ayrıştırıcı araç (TextAnalyzer) sayesinde analiz edilerek kelime ve kelime kökleri bilgileri döndürülmektedir.



Şekil 4.4 – Özgeçmiş İçerik Çıkarıcı ve Doğal Dil Ayrıştırıcı Çalışma Mimarisi

Örnek olarak; bir özgeçmiş içeriğinden elde edilen metnin bir parçasının analizi Şekil 4.5'te gösterilmiştir. Buna göre cümleler öncelikle kelimelere ayrıştırılmaktadır. Kelimeler elde edildikten sonra kelimelerin kökleri ve almış oldukları ekler belirlenmektedir. Kelimelerin köklerinin isim, fiil vb. gibi türleri belirlenmektedir. Ayrıca kelimelerin almış oldukların eklerin türleri de belirlenmektedir.

Şekil 4.5 – Bir metnin analizi

Metin analiz edildikten sonra elde edilen kelimeler ve kelimelere ait kök ve tür bilgileri geri döndürülmektedir. Bu verilere göre özgeçmişin sınıflandırılması ve gerekiyorsa etiketlenmesi gerçekleştirilerek bilgiler saklanmaktadır.

4.3.5. Metin etiketleyici aracın performansı

Geliştirilen metin etiketleyici aracın başarı oranını ölçmek için bazı bilgi geri getirimi ölçüm yöntemleri kullanılmıştır. Kullanılan bilgi geri getirimi ölçüm yöntemleri kesinlik (*presicion*), hatırlama (*recall*) ve f-ölçüsüdür (*f-measure*).

Bu ölçüm yöntemleri genel olarak bir arama sonrasında getirilen doğru veya yanlış sonuçların sayılarıyla hesaplanır. Metin etiketleyici aracın performansının test edilmesinde ise etiketlenen bir metindeki doğru etiketlemelerin ve yanlış etiketlemelerin sayıları bu yöntemler dahilinde kullanılarak ilgili değerler hesaplanmıştır.

Metin etiketleyici aracın performansı ölçmek için bir test verisi hazırlanmıştır. Bu test verisi çeşitli özgeçmiş örneklerinden oluşan bir veri kümesidir. Test veri kümesindeki verilerin etiketlenmesinde kullanılmak üzere anahtar-değer biçimindeki yetenek verileri kullanılmıştır. Bu örnek veri kümesi üzerinde metin etiketleyici aracın testi sonrasında hatırlama, kesinlik ve f-ölçüsü değerleri elde edilmiştir.

Örnek veri kümesi ile ilgili bilgiler şöyledir:

- Toplam doküman sayısı: 228
- Toplam kelime sayısı: 62914
- Toplam ilişkilendirilmesi beklenen etiket sayısı: 347

Hatırlama (*Recall*):

Metin içerisinde doğru olarak etiketlenen etiketlerin sayısının ilişkilendirilmesi beklenen toplam etiket sayısına oranı hatırlama (*recall*) olarak adlandırılmaktadır.

$$Hatırlama = \frac{doğru\ etiketler}{Olması\ gereken\ toplam\ doğru\ etiketler}$$

Test verisi üzerinde metin etiketleyici aracın çalıştırılması sonrasında 319 adet metinlerle ilişkili doğru etiket döndürülmüştür. Metinlerden etiketlenmesi beklenen toplam doğru etiket sayı ise 347'dir. Bu verilerle birlikte elde edilen hatırlama değeri:

$$Hatırlama = \frac{319}{347} \cong 0.92$$

Kesinlik (*Precision*):

Metin içerisinden çıkarılan doğru etiketlerin toplam etiketleme sayısına oranına kesinlik (*precision*) denmektedir.

$$Kesinlik = \frac{doğru\ etiketler}{doğru\ etiketler + yanlış\ etiketler}$$

Test sonucunda toplam etiket sayısı 378 olmuştur. Bu etiketlerden 319 tanesi doğru etikettir. Geriye kalan 59 etiket ise aslında olmaması gereken yanlış etiketlemelerdir. Bu değerler üzerine elde edilen sonuç:

$$Kesinlik = \frac{319}{378} \cong 0.84$$

F-ölçüsü (*F-measure*):

Hatırlama (*recall*) ve kesinliğin (*precision*) harmonik ortalamasına f-ölçüsü (*f-measure*) denmektedir.

$$F - \text{ölçüsü} = \frac{2 \times hatırlama \times kesinlik}{hatırlama + kesinlik}$$

Hatırlama ve kesinlik değerleri üzerinden alınan harmonik ortalama sonrasında elde edilen f-ölçüsü sonucu:

$$F - \text{ölçüsü} = \frac{2 \times 0.92 \times 0.84}{0.92 + 0.84} \cong 0.88$$

5. “ANLAMSAL LINKEDIN”

LinkedIn profesyonel sosyal ağ sitesi kişilerin profillerine girdikleri bilgileri ile birer özgeçmiş değeri taşımaktadır. Bu yönüyle iş hayatında önemi büyüktür. Hatta LinkedIn şirketlerin iş başvurularını LinkedIn üzerinden alması için özel bir uygulama geliştirme arayüzü (API) desteği sunmaktadır. Bu uygulama geliştirme arayüzünün adı ise “Apply with LinkedIn” (LinkedIn ile başvur) (Şekil 5.1).

Şekil 5.1 - Apply with LinkedIn uygulaması butonu

“Apply with LinkedIn” ile başvuruda bulunan kullanıcıların profillerinde bulunan gerekli tüm bilgiler kullanıcının izni doğrultusunda erişilebilmektedir. Ayrıca bir özgeçmiş ile başvuruda bulunmasına gerek kalmamaktadır.

Şirketler kendi mevcut çalışanlarının LinkedIn hesaplarını kullanıcıların izni dahilinde “Apply with LinkedIn” arayüzüyle uygulamaya bağlayarak mevcut çalışanlar hakkında özgeçmişlerinde bulunan bilgileri LinkedIn üzerinden alabilirler. Geliştirilen yetenek veri modeli ve Anlamsal LinkedIn uygulaması ile çalışanlara ait yetenek profili çıkartılmaktadır. Bu çalışanlar ait yetenek verisini zenginleştirmek için çalışanların şirketlere vermiş oldukları özgeçmişler de analiz edilmektedir. Yapısal olmayan metinsel veri içeren bu özgeçmişler de analiz ederek elde edilen veri ilgili kişinin yetenek profiline yetenek veri modeli dahilinde eklenmektedir.

5.1. LinkedIn’den Yarı Yapısal Kullanıcı Verilerini Alma

LinkedIn kullanıcılarına ait verileri almak için LinkedIn tarafından sağlanan çeşitli uygulama geliştirme arayüzleri mevcuttur. Bunlardan herkese açık olan API ile kullanıcılara ait kısıtlı verilere erişilebilmektedir. Ancak kurumların başvurup kullanım hakkı alabildikleri “Apply with LinkedIn” uygulama geliştirme arayüzü ile geliştirilen veri transfer uygulamasıyla kullanıcıların yetenek ve diğer tüm verileri alınabilmektedir. LinkedIn veri transfer uygulamasının geliştirilmesinde LinkedIn’in REST API desteği kullanılmıştır.

Erişilebilecek kullanıcı profil alanları

Üye profili alanları:

Aşağıdaki üye profil alanları tüm LinkedIn geliştiricilerine açıktır.

- Temel profil alanları

- Lokasyon alanları
- Pozisyon alanları

Sadece “Apply with LinkedIn” geliştiricilerine açık olan üye profili alanları:

Aşağıdaki üye profil alanları sadece başvuran ve onaylanmış “Apply with LinkedIn” uygulamalarına açıktır.

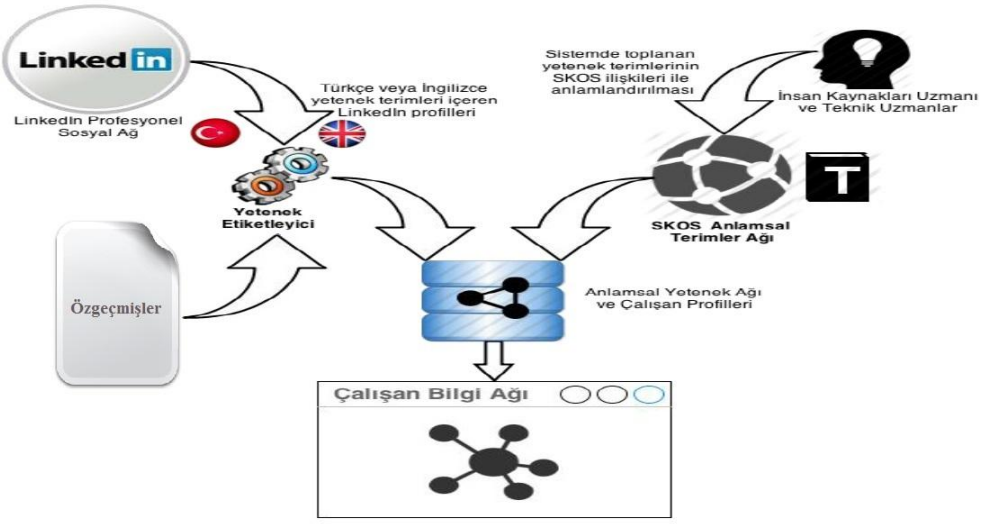
- Tüm profil alanları
- İletişim bilgileri alanları
- Kurum bilgileri alanları
- Yayın bilgileri alanları
- Patent bilgileri alanları
- Dil alanları
- Yetenek alanları
- Sertifika alanları
- Kurs alanları
- Eğitim alanları
- Askerlik alanları
- Referans alanları

Geliştirilen uygulama kapsamında kullanıcının LinkedIn profilini bağlarken tüm profil bilgileri alanları dışında izin alınması gereken, uygulama için temel olarak yeterli olan alanlar ise şunlardır:

- Eğitim alanları
- Kurum bilgileri alanları
- Kurs alanları
- Yetenek alanları
- Sertifika alanları
- Yayın bilgileri alanları
- Patent bilgileri alanları
- Dil alanları

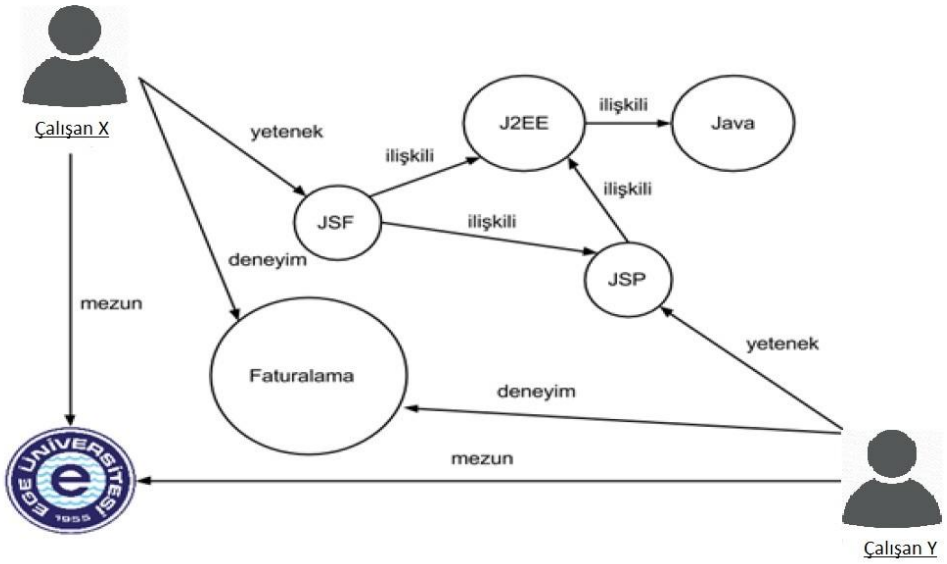
5.2. LinkedIn Verilerinin Anlamsallaştırılması

“Apply with LinkedIn” API’si kullanılarak geliştirilen veri transfer uygulaması aracılığıyla elde edilen kullanıcı yetenek verilerinin işlenmesi ve anlamlandırılması gerekmektedir. Bunun için elde edilen metinsel veriler önceki bölümde ayrıntılı anlatılan geliştirilmiş metin etiketleme aracından geçirilmektedir. Bu işlem sonucunda anlamsallaştırılmış LinkedIn verileri elde edilir. Bu süreç Şekil 5.2'de gösterilmiştir.



Şekil 5.2 - LinkedIn verilerini anlamsallaştırma süreci

Elde edilen verinin tutulduğu model örneği ise Şekil 5.3'te gösterilmektedir. Buna göre Çalışan X ve Çalışan Y çalışanları arasındaki benzerlikler ve ilişkiler çıkarılabilmektedir.



Şekil 5.3 - Anlamsal LinkedIn verisi

5.3. Anlamsal Verinin Saklanması

Yapısal olmayan web içeriklerinin ve özgeçmiş metinlerinin etiketlenmesiyle elde edilen anlamsal veriyi kullanım durumlarına göre en etkin

şekilde saklamak gerekmektedir. Bu doğrultuda bir melez veri altyapısı geliştirilmesine karar verilmiştir.

Geliştirilen melez veri altyapısında yer alacak olan veri depolama teknolojilerinin seçilmesi önemli bir konudur. Verilerin kullanılacakları amaçlara en uygun veri depolama teknolojisinin seçilmesi uygulamanın etkinliğini ve çalışma hızını doğrudan etkilemektedir.

Yetenek veri modeli

Yetenek veri modeli ve bu modele ait yetenek verileri bağlı veri biçimde RDF üçlüleriyle SKOS konseptleri kullanılarak tutulmaktadır.

Yetenek veri modelinde düğümler arasındaki ilişkiler için kullanılan SKOS konseptleri şunlardır:

- skos:broader
- skos:narrower
- skos:related

Bu SKOS konseptlerinden *skos:broader* ilgili veri kaynağının daha kapsamlı olan bir veri kaynağıyla ilişkili olduğunu tanımlamak için kullanılır. *skos:narrower* ise ilgili veri kaynağının daha dar bir kapsama sahip başka bir veri kaynağıyla ilişkili olduğunu tanımlamak için kullanılır. Diğer SKOS konsepti *skos:related* ise veri kaynakları arasındaki kapsam durumunu belirtmeden ilişkili olduklarını belirtir.

Yetenek veri modeli dışında da SKOS konseptleri kullanılmaktadır. Bir bağlı veri kaynağının SKOS konsepti olduğunu belirtmek için o kaynağın *rdf:type* özelliği *skos:Concept* sınıfıyla tanımlanır.

Yetenek modeline ait verilerde bazı diğer SKOS konseptleri kullanılmaktadır. Bu SKOS konseptleri:

- skos:prefLabel
- skos:altLabel

Bu konseptlerden *skos:prefLabel* ilgili kaynağın tercih edilen ismini belirtir. Farklı dillerdeki isim tanımlamaları için dil etiketi ile birlikte bir kaynak için birden fazla *skos:prefLabel* konsepti tanımlanabilir. *skos:altLabel* ise alternatif isim tanımlamaları için kullanılır. *skos:prefLabel* ile aynı şekilde dil etiketleriyle birden fazla *skos:altLabel* konsepti tanımlanabilir. SKOS konseptleriyle tanımlanmış yetenek verisi üçlüleri örneği Şekil 5.4'te gösterilmektedir.

```
<http://dbpedia.org/resource/Category:Programming_languages><http://www.w3.
org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2004/02/skos/core#Concept> .

<http://dbpedia.org/resource/Category:Object-oriented_programming_languages>
<http://www.w3.org/2004/02/skos/core#broader>
<http://dbpedia.org/resource/Category:Programming_languages> .

<http://dbpedia.org/resource/Category:Object-oriented_programming_languages>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2004/02/skos/core#Concept> .

<http://dbpedia.org/resource/Category:Java_(programming_language)>
<http://www.w3.org/2004/02/skos/core#broader>
<http://dbpedia.org/resource/Category:Object-oriented_programming_languages>
.

<http://dbpedia.org/resource/Category:Java_(programming_language)>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2004/02/skos/core#Concept> .

<http://dbpedia.org/resource/Category:Java_(programming_language)>
<http://www.w3.org/2004/02/skos/core#prefLabel> "Java (programming
language)"@en .

<http://dbpedia.org/resource/Category:Java_(programming_language)>
<http://www.w3.org/2004/02/skos/core# prefLabel > "Java programlama dili"@tr
.

<http://dbpedia.org/resource/Category:Java_(programming_language)>
<http://www.w3.org/2004/02/skos/core#altLabel> "Java" .

<http://dbpedia.org/resource/Category:Objective-C>
<http://www.w3.org/2004/02/skos/core#broader>
<http://dbpedia.org/resource/Category:Object-oriented_programming_languages>
.

<http://dbpedia.org/resource/Category:Objective-C>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2004/02/skos/core#Concept> .

<http://dbpedia.org/resource/Category:Objective-C>
<http://www.w3.org/2004/02/skos/core#prefLabel> "Objective-C" .
```

Şekil 5.4 – SKOS Yetenek verileri

Yetenek anahtar-değer verileri, çalışanların temel verileri ve çalışanlarla ilişkilendirilmiş yetenek verileri

Yetenek verilerinin ve özgeçmişlerin etiketlenebilmesi için gerekli olan anahtar-değer veritabanı için MongoDB'nin kullanılması tercih edilmiştir. MongoDB'nin tercih edilme sebebi klasik bir anahtar-değer yapısı değil de özel tasarlanmış bir anahtar-değer yapısının kullanılacak olmasıdır. Özel tasarlanmış olan bu anahtar-değer yapısı Şekil 5.5'te gösterilmiştir.

```
{
  "_id": "ObjectId("55954e8c46d6d5ccb55cbb8f")",
  "display":"Java Programlama Dili",
  "code":"Skill$19832",
  "language":"tr",
  "words":[
      {
        "word":"java",
        "root":"java"
      },
      {
        "word":"programlama",
        "root":"program"
      },
      {
        "word":"dili",
        "root":"dil"
      }
   ]
}
```

Şekil 5.5 - MongoDB yetenek anahtar-değer veri yapısı

Ayrıca etiketlenen özgeçmiş dokümanları ve çalışanların hangi yeteneklere sahip oldukları da MongoDB doküman veritabanında tutulmuştur. TDB bağlı veri veritabanındaki yetenek verileri ile MongoDB doküman veritabanındaki veriler ortak benzersiz kodlarla eşlenmektedir.

Önbelleklemeler

Önbellekleme işlemleri için Redis anahtar-değer veritabanı kullanılmıştır. Özgeçmiş metinlerinde geçen kavramlar etiketlenirken MongoDB anahtar-değer koleksiyonundan etiketlenmiş olan kaynaklar önbellekte saklanmaktadır. Böylece daha sonra aynı kavramlar etiketlenmek istendiği tekrar sorgu yapılmadan önbellekten getirilmektedir.

Doğal dil işleme aracında analiz edilen metinlerdeki her kelime ve analiz sonuçları önbellekte saklanmaktadır. Böylece aynı kelimelerin analizi sadece tek sefer yapılmakta, aynı kelime analiz için doğal dil işleme aracına geldiğinde yeniden analiz gerçekleştirilmeden önbellekten analiz sonucu döndürülmektedir.

5.4. Anlamsal Verinin Kullanılması

Oluşturulan yetenek veri modeli üzerinden herhangi bir yetenek verisinin kendisine, daha kapsamlı yeteneklere, daha dar kapsamlı yeteneklere ve bu yetenekle ilişkili diğer yeteneklerden istenilen özelliğe sahip olanlarına erişebilmek mümkündür. Bunun için bir öneri sistemi kurulmalı ve yetenekler üzerinden ister ilişkili yeteneklere, ister ilişkili ancak daha dar kapsamlı yeteneklere, ister ilişkili ancak daha kapsamlı yeteneklere sahip olan çalışanlara rahatça erişim sağlanabilir. Bunun sayesinde herhangi bir ekip için hangi yetenekler gerekiyorsa o yetenekler üzerinden en uygun çalışanlar ilgili ekibe dahil edilebilir.

Bu model sayesinde kuruluşlar bir proje kapsamında kullanılması gereken teknolojilerle ilgili bilgisi olan bir çalışana sahip olmasalar bile, hangi çalışanların henüz sahip olmadıkları bu yeteneğe en yakın, en ilişkili yeteneklere sahip oldukları çıkarılabilir ve bu çalışanlardan oluşan bir ekip kurulabilir. Bu ekip ilgili yeni teknolojiye diğer çalışanlardan daha kolay adapte olacaktır.

Bu çalışma ile kuruluşların çalışanlarının yeteneklerinden en etkin şekilde yararlanması, kurulacak çalışma ve proje ekiplerinde ihtiyacı uygun ve en uyumlu çalışanlardan oluşan takımlar kurulması ve yeni bir teknoloji öğrenilmesi gerektiğinde buna en uygun çalışanlarını seçebilmesi hedeflenmiştir. Bu sayede kuruluşta çalışanlardan alınan verim artacaktır. İş gücünün etkin bir şekilde kullanılması sağlanmış olacaktır.

6. SONUÇ

Anlamsal ağa geçişin devam ettiği günümüzde Türkçe verilerin de bilgisayar tarafından anlaşılabilir hale getirilmesi büyük önem taşımaktadır. Bu kapsamda Türkçe doğal dilde bulunan metinlerin analiz edilmesi ve bu doğal dil metinlerinin anlamsallaştırılması gerekmektedir. Bu kapsamda Türkçe metin etiketleyici araç ve çalışma kapsamında ele alınan yetenek verileri ile ilgili bağlı veri yapıtaşlarını içeren bir veri modeli geliştirilmiştir.

Kuruluşların en önemli sorunlarından biri olan yetenek yönetimini etkin bir şekilde çözebilmeleri için insan kaynakları bölümünün kullanabileceği bir uygulama için yöntem önerilmiştir.

LinkedIn profesyonel sosyal ağ sitesinden alınan yarı yapısal kullanıcı yetenek verilerinin etiketlenmesinde %100 tam başarı sağlanmaktadır. Başarı oranın %100 olmasında LinkedIn'den alınan verilerin yarı yapısal olması ve anahtar-değer yetenek veri havuzunda ilgili tüm yetenek verilerinin bulunması rol oynamaktadır.

Özgeçmişlerden elde edilen yapısal olmayan metinlerin analizinde ise %88 oranında başarı sağlanmıştır. Hedeflenen başarı oranı (%80-%92) böylece tutturulmuş olmuştur.

Yetenek veri modelinde SKOS konseptleri kullanılarak elde bulunan verinin anlamsallaştırılması ve web teknolojisinin geleceği olan anlamsal ağa (Semantic Web - Web 3.0) uyumlu olması sağlanmıştır. Ayrıca verinin sadece insanlar tarafından değil makineler (bilgisayarlar) tarafından da anlaşılması, işlenmesi ve öneriler sunulması sağlanmıştır.

KAYNAKLAR DİZİNİ

Berners-Lee, T., Hendler, J., and Lassila, O., The Semantic Web, Scientific American, May 2001, p. 29-37.

Shadbolt, N., Hall, W. and Berners-Lee, T., 2006, The Semantic Web Revisited, IEEE Intelligent Systems Journal, May/June 2006:96-101p.

Sadalage, P. J. and Fowler, M., 2012, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Indiana, USA, 232p.

Akın, A. A. ve Akın, M. D., 2007, Zemberek, an open source NLP framework for Turkic Languages, 8p (unpublished)

Çöltekin, Ç., 2010, A Freely Available Morphological Analyzer for Turkish, LREC - European Language Resources Association, Malta

Çelik, D. and Elçi, A., 2012, An Ontology-Based Information Extraction Approach for Résumés, ICPCA/SWS 2012, 165-179p

OpenCalais, 2014, Calais: Connect. Everything., http://www.opencalais.com/about (Erişim tarihi: 6 Ağustos 2015)

Schandl, T. and Blumauer, A., 2010, PoolParty: SKOS Thesaurus Management Utilizing Linked Data, The Semantic Web: Research and Applications Lecture Notes in Computer Science Vol. 6089, 421-425p

Cunningham, H., Maynard, D., Bontcheva, K. and Tablan, V., 2002, GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications, 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, USA

Mendes, P., Jakob, M., García-Silva, A. and Bizer, C., 2011, DBpedia Spotlight: Shedding Light on the Web of Documents, 7th International Conference on Semantic Systems (I-Semantics), Graz, Austria

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. G., 2007, DBpedia: A Nucleus for a Web of Open Data, ISWC/ASWC 2007, 722-735p

Schulz, A., Ristoski, P. and Paulheim, H., 2013, I See a Car Crash: Real-time Detection of Small Scale Incidents in Microblogs, Social Media and Linked Data for Emergency Response (SMILE) Workshop, ESWC 2013, Montpellier, France

Horowitz, E., 2015, MongoDB Architecture Guide, New York, USA, 16p (unpublished)

KAYNAKLAR DİZİNİ (devam)

Ok, A., 2012, Turkish Natural Language Toolkit, https://github.com/aliok/trnltk (Erişim tarihi: 6 Ağustos 2015)

The Apache Software Foundation, 2007, Apache Tika - a content analysis toolkit, http://tika.apache.org/ (Erişim tarihi: 6 Ağustos 2015)

W3C, 2015, Semantic Web, http://www.w3.org/standards/semanticweb/ (Erişim tarihi: 6 Ağustos 2015)

W3C, 2013, W3C DATA ACTIVITY Building the Web of Data, http://www.w3.org/2013/data (Erişim tarihi: 6 Ağustos 2015)

W3C, 2014, Latest "RDF Primer" versions, http://www.w3.org/TR/rdf-primer/ (Erişim tarihi: 6 Ağustos 2015)

LinkedIn Corporation, 2015, Apply with LinkedIn, https://developer.linkedin.com/docs/apply-with-linkedin (Erişim tarihi: 6 Ağustos 2015)

ÖZGEÇMİŞ

Enes Bulut, 1990 yılında Polatlı'da doğdu. Türkiye Cumhuriyeti vatandaşıdır. İlk ve orta öğretim eğitimini Polatlı'da, lise eğitimini Ankara'da tamamladı. Lisans derecesini 2012 yılında Ege Üniversitesi Bilgisayar Mühendisliği bölümünde aldı. Aynı bölümde yüksek lisans eğitimini tamamladıktan sonra doktora derecesi için yine aynı üniversite ve bölümde eğitimine devam edecektir. Lisansüstü eğitimine devam ederken bir yandan da özel bir şirkette anlamsal ağ, doğal dil işleme ve büyük veri konularındaki araştırma & geliştirme projelerinde görev almaktadır.