BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TÜRKÇE METİNLERDEN ANLAMSAL BİLGİ ÇIKARIMI İÇİN BİR VERİ MADENCİLİĞİ UYGULAMASI

A. ALPHAN ARSLAN

YÜKSEK LİSANS TEZİ

2011

TÜRKÇE METİNLERDEN ANLAMSAL BİLGİ ÇIKARIMI İÇİN BİR VERİ MADENCİLİĞİ UYGULAMASI

A DATA MINING APPLICATION FOR EXTRACTING SEMANTIC INFORMATION FROM TURKISH TEXTS

A. ALPHAN ARSLAN

Başkent Üniversitesi
Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin
BİLGİSAYAR Mühendisliği Anabilim Dalı İçin Öngördüğü
YÜKSEK LİSANS TEZİ
olarak hazırlanmıştır.

2011

“Türkçe Metinlerden Anlamsal Bilgi Çıkarımı İçin Bir Veri Madenciliği Uygulaması” başlıklı bu çalışma, jürimiz tarafından, 23/02/2012 tarihinde, **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan : Prof. Dr. Ziya AKTAŞ

Üye (Danışman) : Doç. Dr. Hasan OĞUL

Üye : Doç. Dr. Hamit ERDEM

ONAY

../../2012

Prof. Dr. Emin AKATA

Fen Bilimleri Enstitüsü Müdürü

TEŞEKKÜR

Eşime.

ÖZ

Günümüzde genel ağın yaygınlaşması ile beraber kaynakların fazlalığı bilgiye erişmede yeni bir sorun olarak ortaya çıkmaktadır. Bilgiye erişimde şu anki durumuyla çeşitli arama motorları anlamsal bağlantılar olmaksızın arama yapmamıza izin vermekte fakat doğru veriye erişmeyi garanti edememektedir. Anlamsal ağ genel ağ üzerinde işlenmiş veriye ulaşmak üzere öngörülen bir yapıdır, fakat beslenmesi için mevcut verilerin işlenmesi gerekmektedir. Bu bağlamda bilgi çıkarımı, doğal dildeki, yapısal olmayan, metinlerin çözümlenmesi ve bu metinlerin içerdiği gerekli bilginin yapısal olarak belirlenmesi işlemidir. Bu noktada veri madenciliği süreçleri bu kez yapısal olmayan veri üzerinde çalışacak şekilde evrilebilir. Bu amaçla belirlenen süreç OİÇ(Otomatik İçerik Çıkarımı) ile tanımlanmıştır. Bu süreç temel olarak 3 adımda oluşur: varlık(ad) çıkarımı, ilişki çıkarımı, olay çıkarımı. Varlık (ad) çıkarımı serbest metinlerde gecen varlık isimlerinin bulunması, ilişki çıkarımı ise metinde belirlenen bu varlıklar arasındaki ilişkinin belirlenmesi ve olay-eylem çıkarımı ise belirlenen bu varlıkların içinde bulunduğu olayların belirlenmesidir.

Bu çalışmanın amacı Türkçe metinlerden bilgi çıkarımı sürecinde ilişkilerin tanımlanması için yapılabileceklerin incelenmesidir. Bu bağlamda Türkçe ve İngilizce için yapılan çalışmalar incelenmiş ve bilgi çıkarım sürecindeki "varlık" kavramının Türkçe bir ada eşit olduğu varsayılarak bir sistem tasarlanmıştır. Gazete haber metinlerinden seçilen tümcelerden bir veri kümesi oluşturulmuş ve oluşturulan bu veri kümesinde her bir tümcenin içinde geçen her iki sözcük , ele alınan tümcelerin yüklem olabilecek sözcüğü göz önüne alınarak , karşılaşma sıklığı, aralarında bulunan diğer sözcüklerin sayısı, tümcede ilk sözcük oluşları gibi bazı özellikleri kullanılarak ilişkili olup olmadıkları hakkında inceleme yapılmıştır. İncelemede kullanılan bu özellikler ifadede bulunduğu konuma göre belirlenen özelliklerin yanı sıra ilk sözcüğün kaç farklı sözcükle birlikte oluşu ,sözcük çiftinin kaç farklı yüklem ile birlikte olduğu gibi sıklık verileri de eklenmiştir. Yapılan inceleme ele alınan iki sözcüğün ilişkili olmalarına karar vermede farklı özelliklerine çeşitli eşik değerler uygulanarak karar verilmiş ve sınıflandırılmış veri

destek vektör makinesi algoritması kullanılarak belirlenen eşik değerleri ve veri kümesinin doğruluğuna dair sonuçlar elde edilmeye çalışılmıştır. Hazırlanan bu sistem Türkçe hazırlanmış bir genel ağ sayfasının içeriğinin belirlenen standartlara uygun hale getirilmesi ve sunulması için bir ön çalışma niteliğindedir.

Anahtar Sözcükler: *Anlamsal Ağ, Metin analizi,Otomatik İçerik Çıkarımı, Bilgi Çıkarımı,Destek Vektör Makinesi Algoritması*

Danışman: Doç. Dr. Hasan OĞUL, Başkent Üniversitesi, Bilgisayar Mühendisliği Bölümü

ABSTRACT

Today, with spreading of internet, information word has a new problem about to reach the right information from the large amounts of data. Existing search engines can only fetch the data even no relation with the search subject. Semantic web technology is created to reach the related information from the web. For this technology, Information Extraction is extracting related and structured information from natural language raw texts. ACE determines whether this processes as a result of conferences and workshops. With ACE the problem deals with three main tasks; Entity Detection and Tracking , Relation Detection and Characterization, and Event Detection and Characterization.

The purpose of this study was to examine Turkish texts can be possible to identify relationships in the process of information extraction. In this study we used Automated Content Extraction(ACE) as a guide to detect relation of entities from Turkish texts. In this context Turkish and English studies examined and assuming that if an entity is equal to a name of a system designed. With this system, news page contents collected and some sentences ,words and verb of sentence abstracted and every couple words in a sentece with the verb of the sentence recorded with some other specifications about the locations in the phrese. After building this frequency data base various views created to determine if the word couple is related. With this views classification made by word couple and verb rate and tested over support vector machine alghoritm. A web page content prepared in Turkish can be brought into line with the standards specified with this system for the submission of a preliminary study.

Keywords: *Semantic Web, Text analysis, Information Extraction, Automated Content Extraction,Support Vector Machine Algorithm*

Advisor: Assoc. Prof. Dr. Hasan OĞUL, Başkent University, Department of Computer Engineering

İÇİNDEKİLER

Sayfa

ÇİZELGE DİZİNİ

ŞEKİL DİZİNİ

1. GİRİŞ

Veriyi depolamak ve sunmak için gerekli altyapının tamamlandığı günümüzde bu verinin anlamlı hale gelmesi yani bilgiye dönüşmesi ve bu bilgiyi insanlar yerine bilgisayarların değerlendirebilir hale gelmesi asıl hedeftir. Makinelerin derleyebildiği bir bilgi sistemi için ilk olarak var olan bilgiyi sunmak ikinci olarak da buna doğru şekilde ulaşılmasını sağlamak gereklidir. Mevcut altyapı (genel ağ) bu bilginin depolanması ve erişilmesi için gayet uygundur. XML (Genişletilebilir Biçimlendirme Dili) üst veri ve ilişkili veriyi saklamak için gayet uygundur ve mevcut sayfalar üzerinde uzlaşılan şekillere uygun olarak hazırlandığı durumda sunum büyük ölçüde tamamlanmış olacaktır. Standartların belirlendiği günümüzde yeni sayfalar bu yapılara uygun olarak geliştirilmeye başlansa da şu ana kadar olanın bu şekle uygun hale getirilmesi yeni bir sorundur. Genel ağda bulunan verinin uygun şekle getirilmesi daha çok doğal dil işleme yöntemleri ile çözülebileceği düşünülmektedir. Metin madenciliğinin anlamına bakacak olursak yapısal olmayan veri kümesinin yapısal bir hale dönüştürülmesi ve bunun üzerine veri madenciliği sürecini geçirdiğini görürüz.

Yapılan çalışma ile bu güne kadar anlamsal ağ için oluşturulan standartlar ve teknolojiler, bunların nasıl kullanılabileceği ve bu çalışmaların Türk dili için nasıl gerçekleştirebileceği sorularına yanıt aranmıştır. İçerik çıkarımı için yapılan çalıştaylar ile elde edilen standartlar ve teknolojiler, geliştirilme süreçleri ve yapılan yayınlar ile Türk dili hakkındaki çalışmalar ve bunların bir araya getirilmesi hakkında yapılan çalışmalar incelenmiştir.

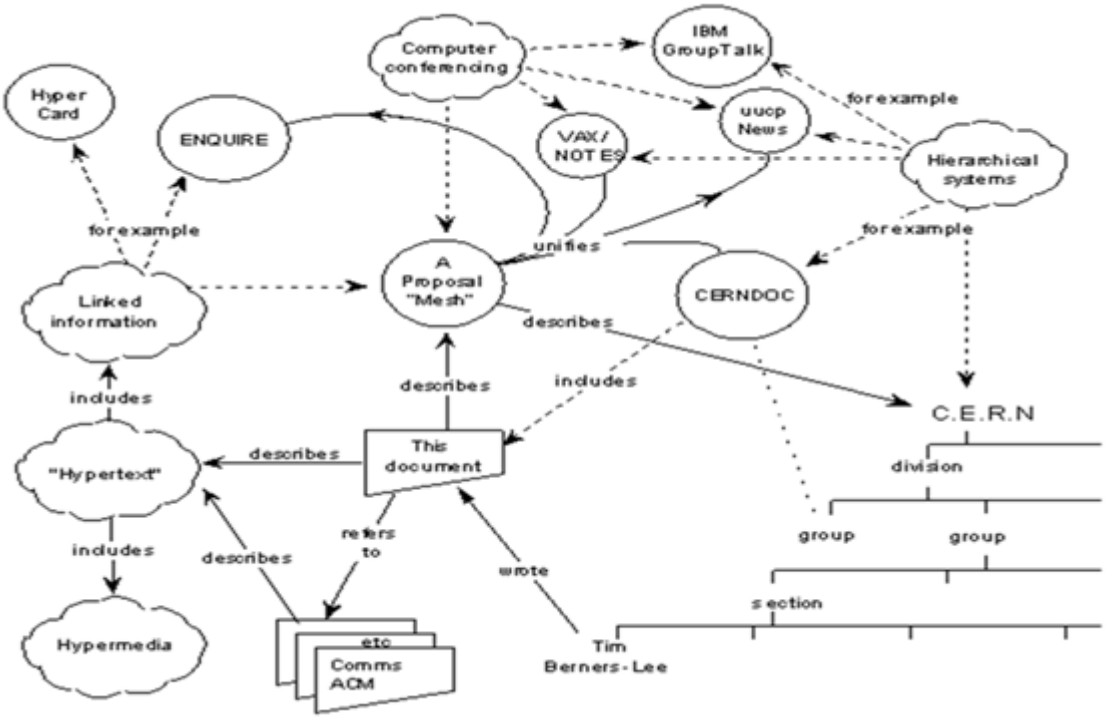
Asıl motivasyonumuz Türkçe sayfalardaki verinin dünya standartlarına uygun ve bilgiye dönüşmüş halde sunulmasını sağlamaktır. Bu amaçla belirlenen bilgi çıkarım görevlerinin ikinci aşaması olan İlişki Çıkarımı süreci incelenmiştir.Sınıflandırmada kullanılan teknolojiler arasında son yıllarda destek vektör makinesi algoritmasının popularitesi nedeniyle sınama işleminde bu algoritma tercih edilmiştir.

2. GENEL BİLGİLER

2.1. Anlamsal Ağ Nedir?

Anlamsal Ağ kavramı, bugünkü Genel Ağ (internet)'ın temel yapılarını (URI, HTTP ve HTML gibi) tasarlayan ve bulan kişi olan Tim Berners-Lee tarafından öne sürülmüş ve mevcut web ortamının geliştirilerek tam potansiyel kullanımı için genel ağın bir sonraki adımı olarak görülmektedir.

Aşağıda Berners-Lee tarafından tasarlanan asıl yapıyı gösteren bir şekil bulunmaktadır.

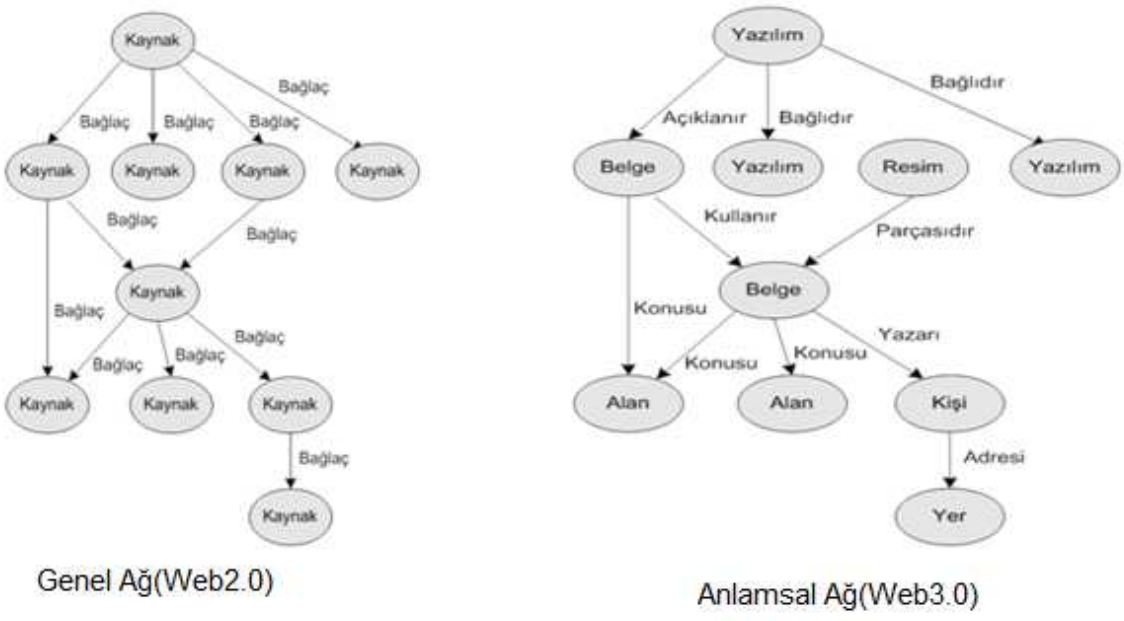


Şekil 2.1 - Berners-Lee tarafından tasarlanan asıl Anlamsal Ağ

Anlamsal Ağ'da temel amaç veriyi, iyi tanımlanmış ve ilişkilendirilmiş olarak bilgi haline getirecek ve servislerin genel ağ ortamında kolay bir şekilde makineler tarafından okunabilir ve anlaşılabilir olmasını sağlayacak standartların ve teknolojilerin geliştirilmesidir.

Genel ağın geçmişine bakarsak Web 1.0 teknolojisi, durağan genel ağ sayfalarının nasıl olduğu, ziyaretçilere bu genel ağ sayfalarının kaynak koduna erişme ve yorum ya da değişiklik yapma izninin verilmediği bir dönemdir. Şuan Web 2.0 teknolojisi ile iç içe yaşıyoruz. Genel ağ sayfalarının içeriğinde değişiklik yapabilme (örnek: vikipedia), insanlarla bağlantı kurabilme (örnek: facebook), verilerimizi paylaşabilme (örnek: youtube) yeteneklerini kullanabilmekteyiz. Özetlemek gerekirse; Kısacası, Web 1.0'ı bir kütüphaneye, Web 2.0'ı bizim de konuşmaya katılarak bilgi alış verişine katkı sağladığımız bir ortama benzetebiliriz.

Biz bu şekilde devam ederken, Web 3.0 (Anlamsal Ağ) teknolojisi bir çok şeyi bilen bir "asistan" rolüne giriyor. Halen bir arama motoru üzerinden arama yaptığımızda, aradığımız kelimeyi içeren genel ağ siteleri sıralanıyor. Bu sitelerin içerisindeki ayırımı biz yapmak zorundayız. Anlamsal Ağ etkin olarak kullanıldığında ise, makineler web sayfalarını yorumlayabilecekleri için doğru sonuçlara ulaşmamızı sağlayacaktır. Aşağıdaki şekilde genel ağ ve tasarlanan anlamsal ağ arasında veriye ve bilgiye ulaşmada ilişkilerin ayırımı yapılmıştır



Şekil 2.2 - Genel Ağ ve Anlamsal Ağ

2.2. Ontoloji Nedir?

Anlamsal Ağ her aradığımızı bulma olasılığımızı arttıracak, oldukça kapsamlı bir veritabanı oluşturmayı hedefliyor. Üst veri (meta data) olarak adlandırılan bu “veri hakkında veri” sistemi, bilgisayarlar tarafından okunabilir bir yapıya sahip olması gerekiyor. Bu noktada anahtar teknoloji olarak karşımıza ontoloji çıkıyor.

Ontoloji terimi felsefede varlık bilim olarak tanımlanmaktadır. Ontoloji doksanlı yıllarda yapay zeka alanında popüler bir terim olarak belirli bir alandaki bilgilerin paylaşımını ve yeniden kullanımını sağlayacak “kavramlaştırmaların biçimsel ve açık belirtimi“ olarak tanımlanmış ve kullanılmıştır. Son dönemde ise ontolojilerin kullanımı zeki bilgi entegrasyonu, doğal dil işleme ve bilgi yönetimi konularında artmaktadır.

Ontoloji bir alandaki bilgilerin “paylaşılan ve genel bir anlamının” oluşmasına imkan verir. Ontolojiler herhangi bir alanda standart olarak kullanılacak ortak ve paylaşılan sözcük kümelerini (vocabulary) veya terminolojiyi belirler.

Ontolojinin bilgisayar bilimindeki en çok kabul gören tanımlarından biri; “kavramsallaştırmanın açıkça belirtilmesidir” (Gruber 1993). Kavramsallaştırma, belli bir tasarım aşamasında soyut model oluşturma anlamına gelir. Bu modelin, tasarım aşamasında bilinen tüm bilgilerinin geride hiç bir soru bırakmayacak şekilde açıkça tanımlanması gerekir (Maedche and Staab 2001).

Ontolojinin başka bir tanımı da ontolojinin neler içermesi gerektiğini ve neleri içeren modellerin ontoloji sayılması gerektiğini açıklar (Maedche and Staab 2001). Bu tanımın diğerlerinden farkı; burada sözü edilen ontolojilerin web ontolojileri olmasıdır. Her terimin tanımlayıcısı bulunur ve ontolojide bulunan terimler sonlu sayıdadır. Bunlar; terimlerin anlamlarını, terimler arasındaki ilişkileri, terimler arasındaki eşanlamlılıkları ve sıradüzen ilişkilerini içerir. Sonuç olarak, bir kavram kümesinin ontolojisinin olabilmesi için en azından aşağıdaki niteliklere sahip olması gerekir:

Terim dağarcığı sonlu ve genişletilebilir olmalı. Sınıflar ve terimler arasındaki ilişkilerin etmenler tarafından tek bir şekilde anlaşılır olması; OWL[1]'nin sınıf yapısını kullanması gerekir.

Üç temel niteliğin yanında ontolojilerin anlatım gücünü arttırabilmek ve nitelikli sorgulama sonuçları almak için, örnek içerme, sınıf niteliklere sahip olma, değer karşılaştırma kıstasları, mantıksal karşılaştırma kıstasları, özellikleri de yukarıda verilen ontoloji olabilme niteliklerine eklenebilir. Bu mantıksal yapıların da dâhil edilmesiyle; ontolojiler, karmaşık bilgileri modelleyebilen ve bu bilgiler üzerinde sonuç üretebilen bir seviyeye gelirler. Ontolojiler ontoloji dilleri (RDFS[2], DAML+OIL[3], OWL, ..) ile tanımlanır. Bir çok alan için değişik ontoloji dillerinde ontolojiler geliştirilmektedir.

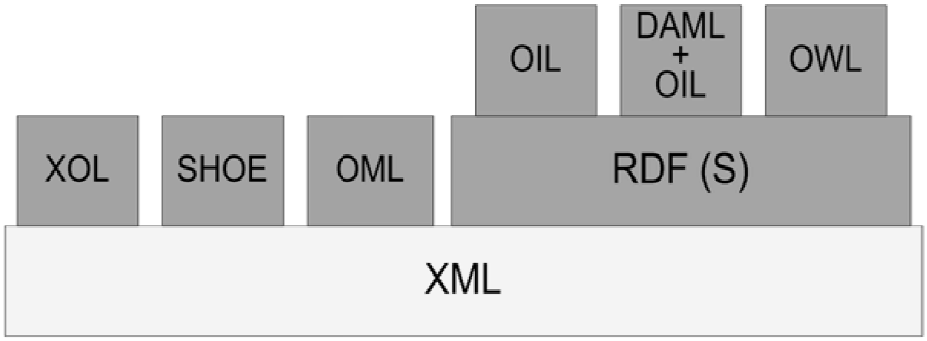
[1] Web Ontology Language (OWL):Ontolojileri tanımlamak ve çeşitlendirmek için kullanılan bir dildir.

[2] Resource Description Framework Schema (RDFS):Kaynak tanımlama çatısının şablonudur.

[3] DAML+OIL:DARPA etmen biçimleme dili ile ontoloji çıkarım katmanı olarak OIL in özelliklerinin birleştirildiği çatıdır.

2.3. Anlamsal Ağ Dilleri

Anlamsal web dilleri ontolojilerin ve ontolojilerle web ortamındaki nesnelerin (kaynakların) tanımlanmasını sağlar.



Şekil 2.3 - RDF'in altyapıdaki yeri

Aşağıda şekil 2.3 ile belirtilmiş olan RDF ve diğer diller sıra ile anlatılmıştır ve sonrasında karşılaştırılmışlardır.

2.3.1. RDF (Kaynak tanımlama çatısı (Resource description framework))

XML dili verilerin kodlanması ve taşınması için söz dizimi yapısını belirler. RDF (Resource Description Framework Kaynak Tanım Çatısı) bir veri modelidir. Bu model web ortamındaki nesnelerin (kaynakların), kaynak özelliklerinin ve özellik değerlerinin tanımlanmasını baz alır. RDF ifadelerinde yer alan nesne, özellik, değer üçlüleri RDF'in temelini oluşturur.

- Kaynaklar (Resources): Üzerinde konuşulan her tür varlık bir kaynak olarak ele alınır.
- Özellikler (Properties) : Özel türde kaynaklardır.
- Değerler (Values): Kaynakların özelliklerinin aldığı değerlerdir. Basit veri türünde olabilecekleri gibi başka URI'lerde değer olarak kullanılabilir.

```
<?xml version=“1.0” encoding=“UTF-16”?>


<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
      <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">
            <dc:title>Tony Benn</dc:title>
            <dc:publisher>Wikipedia</dc:publisher>
      </rdf:Description>
</rdf:RDF>
```

Şekil 2.4(RDF kullanım örneği)

Nesne , özellik , değer terimleri RDF'te özne(subject) , yüklem(predicate) , nesne(object) olarak adlandırılır.RDF ifadelerinin özne ,yüklem ,nesne üçlüsü olarak gözterilir.İfadeler oluşturulurken ml standartları kullanılır.

2.3.2. RDFS (RDF schema)

RDF veri modeli genel ağ ortamındaki kaynaklar, isimlendirilmiş kaynak, özellikleri ve değerleri üçlülerini temel alan basit bir gösterim yöntemidir.

RDFS gösterimi RDF veri modelini genişleten bir tip sistemidir. Bu tip sistemi bir alanda kullanılacak olan sözcük kümesini tanımlar.

Çizelge 2.1 - RDF şeması anahtar sözcükler

Nesne ve ya özellik	**Açıklama**
rdf:type	Özne bir nesne örneği ve ya tipindedir
rdfs:Class	Nesne tanımlar
rdfs:subClassOf	Nesne bir nesnenin alt nesnesidir
rdf:Property	Özellik tanımlar
rdfs:subPropertyOf	Özne bir özelliğin alt özelliğidir
rdfs:domain	Özne özelliğinin alanı
rdfs:range	Özne özelliğinin alabileceği alan değerleri sahası
rdfs:label	Öznenin açıklayıcı adı
rdfs:comment	Özne kaynağı hakkında açıklama
rdfs:Literal	Karakter değerler sınıfı(tam sayılar,karakter dizileri)

RDSF nesnelerin üst küme / alt küme ilişkilerinin tanımlanmasına imkan verir.

2.3.3. DAML(DARPA agent markup language)+OIL(Ontology interface layer)

Anlamsal ağın temelini oluşturan ontolojileri tanımlamak için RDFS şema dilinin yeteneklerini genişleten üst seviye dillere gereksinim duyulmaktadır. RDF(S)' in bir üst seviye katmanı olarak DAML (DARPA Agent Markup Language), OIL

(Ontology Interface Layer), DAML+OIL ve OWL (Web Ontology Language) ontoloji dilleri tanımlanmıştır.DAML+OIL şu aşamada en gelişmiş ve olgunlaşmış dil olarak gözükmektedir.DAML dili Amerikan hükümetinin desteklediği bir çalışma sonucunda Ağustos 2000'de yayınlanmıştır.OIL (Ontotoloji Interface Layer) Avrupa Birliği IST programı çerçevesinde geliştirilmiş bir dildir.

Bu iki dilin yapılarını birleştirmek için Amerika ve Avrupa Birliği'nce oluşturulan ortak komite DAML+OIL dilini geliştirerek Aralık 2000'de yayınlamıştır.

DAML+OIL'in en son versiyonu Mart 2001'de yayınlanmıştır. İlk yayın tarihinden itibaren DAML+OIL bir çok anlamsal web araştırmacısının ilgisini çekmiş ve yaygın bir kullanımı oldukça yaygınlaşmıştır.

Şu anda değişik alanlar için DAML+OIL ile geliştirilmiş yaklaşık 250 adet ontoloji ve 60 tane bu dile özel geliştirme aracı bulunmaktadır.

Anlamsal web servisi yeteneklerinin anlamsal web ortamında temsil edilmesi ve dinamik olarak bulunup kullanılması için geliştirilen bir web servisi ontolojisi bulunmaktadır. Bu ontoloji, kendisinin geliştirilmesinde kullanılan ontoloji diline balı olarak DAML-S veya OWL-S adını almaktadır.

Aşağıda teknolojilerin bazı yönlerini karşılaştırmak üzere (Çizelge 2.2) bir tablo hazırlanmıştır.

Çizelge 2.2 - Anlamsal Ağ teknolojilerinin karşılaştırılması

Özellik	XML Sema	RDF(S)	DAML OIL	OWL
Sınırlı listeler (bounded lists)			✓	✓
Çokluk kısıtlamaları cardinality constraints)	✓		✓	✓
Sınıf ifadeleri (class expressions)			✓	✓
Veri tipleri (data types)	✓		✓	✓
Tanımlı sınıflar (defined classes)			✓	✓
Değer kümesi (enumarations)	✓		✓	✓
Eşitlik (equvalance)			✓	✓
Genişleyebilirlik (extensibility)		✓	✓	✓
Biçimsel anlam bilimi (formal semantics)			✓	✓
Kalıtım (inheritance)		✓	✓	✓
Çıkarım (inference)			✓	✓
Yerel sınıflar (local restrictions)			✓	✓
Nitelikli kısıtlamalar (qualified constraints)			✓	
Somutlaştırma (reification)		✓	✓	✓

3. BİLGİ ÇIKARIMI

Genel ağ sayesinde veriye ulaşma şeklimizde değişti. Çok fazla veri arasında arama motorları sayesinde arama yapabiliyoruz ve erişme sıklıkları ve eriştiklerimizi yorumlayarak habere ulaşabiliyoruz. Fakat arama motorları anlamlı habere yani bilgiye ulaşmamızı garanti edemiyorlar. 1999 da yapılan araştırmaya göre 1.5 exabyte (1.5 milyar gigabyte) kaynak internet ağında yer alıyor. Bu kadar fazla kaynağa erişebiliyor olmamız içeriğindeki bilgiye ulaşmamız anlamına gelmiyor. Bilgiye erişmek bu haliyle yeni bir sorun olmuş oluyor.

Çoğu sorunun çözümünde faydalandığımız gibi bu sorununda çözümünde bilgisayar sistemlerinden yardım alabiliriz. Bilgi elde etme (Information retrival) geniş ölçekli dijital kütüphanelerden verinin sınıflandırılıp işlenebilir hale getirilmesini amaçlar.

Çoğu bilginin veritabanları yerine doğal dil içinde olması doğal dil işlemeyi çözümün bir parçası haline getirmiştir. Doğal dil işleme, yapay zeka kullanımı ve dilbilim yaklaşımları bu süreçte analiz etme ve anlamada en önemli araçlardır.Metin içindeki anlamın çıkartılması ve bilgisayar tarafından anlaşılabilir hale getirilmesi amaçlanmaktadır.

3.1. Bilgi Çıkarımı Nedir (Information Extraction IE) ?

Basit olarak Bilgi Çıkarımı(IE),doğal dil ile oluşturulmuş bir metinden, varlıkların tanımlanması , tanımlanan varlıklar arası ilişkilerin belirlenmesi ve olayların belirlenmesi sürecidir.Yapısal olmayan bir metinden yapılan analiz ile bilgiye erişilmesi ve yapısal halde sunulması süreci de denebilir.

Bir önceki bölümde de belirtildiği gibi bilgi çıkarımı doğal dil işlemede önemli bir görevdir.Ancak Bilgi Çıkarımı diğer birçok DDİ sürecinden daha kolay yönetilebilirdir.Öncelikle bilgi çıkarımı yazarın niyeti yerine metin hakkında genel soruları yanıtlamaya çalışır.Amaç tanımlanan şablonda yuvaları doldurmaktır.Bu nedenle bilgi çıkarımı için bir belgenin özeti yeterli olabilir.Bu yüzden ne aradığımızı biliyorsak ve nasıl değerlendirileceğini biliyorsak Bilgi Çıkarımı bizim için çok iyi tanımlanmış bir süreçtir.

Bir kaç basit örnek vermek gerekirse; ekonomi sayfasından elde ettiğimiz metini inceleyebiliriz.

<HaberMetin>

<Baslik>” <Varlik>HUGO</Varlik> ” geri döndü!</Baslik>

<Oz><Varlik>Hugo</Varlik>, orijinal ortamında, <Varlik> iPhone</Varlik>,<Varlik> iPad</Varlik>,<Varlik> iPod</Varlik>, <Varlik>Android</Varlik>, <Varlik>PlayStation</Varlik>, <Varlik>PSP</Varlik>, <Varlik>Minis</Varlik> ve PC eğlence oyunu olarak yeniden başlıyor </Oz>

<Icerik> O günlerde <Varlik>Hugo</Varlik> ve fantastik TV oyununun tüm ülkeyi nasıl alt üst ettiğini hatırlıyor musunuz? Şimdi <Varlik>Hugo</Varlik> ve madenine unutulmaz ziyaret yeniden yaklaşıyor, hazır olun. <Varlik>Hugo</Varlik> Retro Mania adı verilen yeni oyun, kaynağını doksanların televizyon eğlencesine göre geliştirilen oyun oynama deneyimimizden alıyor. Ancak şimdi, macera arayışına girmek için sabit hatlı telefon tuşları gerektiren geçmiş deneyiminizden farklı olarak, <Varlik>Hugo</Varlik> Retro Mania’yı, dokunmatik ekranlı telefonunuzda, bilgisayarınızda veya tabletinizde oynayabilirsiniz.

“<Varlik>Hugo</Varlik> ’yu aldığımızdan beri ilk defa, <Varlik>Hugo</Varlik> ’ya ve özellikle de <Varlik>Hugo</Varlik> ’nun dünya çapındaki şöhretine uygun hale getirilmiş bir proje geliştirdik. Çoğu insan <Varlik>Hugo</Varlik> ’nun Güney Amerika, Almanya, İspanya, Fransa, Rusya ve Türkiye gibi ülkelerdeki şöhretini duyduğunda şaşırıyor” şeklinde konuşan KREA Medie genel müdürü Henrik Kølle, sözlerini şöyle sürdürdü: “Ancak biz tecrübesiz değiliz ve bugün rekabetin her zamankinden daha şiddetli olduğunun farkındayız. Bu nedenle <Varlik>Hugo</Varlik> Retro Mania’dan beklentimiz, gerçekten de <Varlik>Hugo</Varlik> ’nun küresel bir marka olarak yeniden yaratılmasını sağlayacak pek çok heyecan verici adımın ilki olması yönünde”.

Şekil 3.1(Varlıkların örnek metinde gösterimi)

<Varlik>Hugo</Varlik> Retro Mania, 15 Aralık 2011 tarihinde on yediden fazla

dilde <Varlik>iPad</Varlik>, <Varlik>iPhone</Varlik>, <Varlik>iPod</Varlik>,<Varlik> Android</Varlik>, <Varlik>PlayStation</Varlik><Varlik> PSP</Varlik><Varlik> Minis</Varlik> ve PC için piyasaya sürülüyor.

<Varlik>Hugo</Varlik> ’nun:

Dünyanın ilk interaktif telefon oyunlarından biri olduğunu

Facebook’da 1.2 milyonun üzerinde hayranı olduğunu

Sekizden fazla ülkede “En iyi TV eğlencesi” ödülünü aldığını biliyor muydunuz?

<Varlik>Hugo</Varlik> hakkında

<Varlik>Hugo</Varlik> , interaktif eğlence sunma gibi özel bir amaçla 1989 yılında dünyaya getirildi. <Varlik>Hugo</Varlik> , bir TV spikeri tarafından yönlendirilen oyuncunun sabit hatlı telefonunun tuşlarını kullanarak <Varlik>Hugo</Varlik> ’yu yönetebildiği bir televizyon oyunuydu. Oyun, büyük bir başarıya imza atılarak, kırkın üzerinde farklı ülkeye dağıtıldı. 2009 yılında, Danimarka Kraliyet Kütüphanesi, <Varlik>Hugo</Varlik> ’nun ulusal dijital kültür mirasının ana temsilcilerinden biri olduğunu duyurdu.

</Icerik>

</HaberMetin>

Şekil 3.1 Devamı (Varlıkların örnek metinde gösterimi)

Verilen örnekte HaberMetin tagı metin başlangıcı ve sonunu belirtmektedir.Oz tagı haberin özünü, içerik tagıda haberin içeriğini belirtmektedir.Varlık işaretçileri ile belirlenen sözcük yada sözcük öbeklerinin bazıları ilişkiler yada olayların belirlenmesi amacıyla belirtilmiştir.

Varlık{isim="Hugo" , tanım="Oyun adı" , tip="Oyun"}

Varlık{isim="iPad" , tanım=" dokunmatik telefonların büyütülmüş ve donanım olarak geliştirilmiş hali " , tip="Elektronik cihaz"}

Varlık{isim=" iPhone " , tanım ABD kökenli Apple şirketinin piyasaya sürdüğü ilk cep telefonu modeli " , tip="Cep telefonu"}

Varlık{isim=" iPod " , tanım=" Apple firması tarafından çıkartılan bir sabit disk tabanlı mp3 çalar " , tip="Elektronik cihaz"}

Varlık{isim=" Android " , tanım=" Linux işletim sistemi tabanlı mobil ve Pda'lar için geliştirilmiş açık kaynak kodlu bir işletim sistemi " , tip="Yazılım"}

Varlık{isim=" PlayStation " , tanım="Oyun konsolu" , tip="Elektronik cihaz"}

Varlık{isim=" PSP " , tanım=" Taşınabilir oyun makinesi " , tip="Elektronik cihaz"}

Varlık{isim=" Minis " , tanım=" Güney Kaliforniya'daki San Francisco vadisinin bir parçası olan Teknoloji merkezi" , tip="Yer"}

Bu varlıkların tanımlanmasında insanlar yerine dogrudan makinaları kullanmak istersek dogrusal bir yaklaşım izlendiğinde önceden tanımlanmış varlıkları içeren bir veri kümesinden yararlanmak gerekebilir. Google gibi bir kaynaktan faydalanarak Hugo olarak tanımladığımız varlık ile ilgili olarak araştırma yaparsak Hugo Oyunları olarak çıkacaktır.

Tanımlanan bu varlıkların arasındaki bazı ilişkileri tanımlamak istersek

İlişki{varlık1="iPod" , varlık2="Apple" , tip="Elektronik cihaz"}

İlişki{varlık1="iPad" , varlık2="Apple" , tip="Elektronik cihaz"}

İlişki{varlık1="Android",varlık2=" Open Handset Allience ve Google",tip="Yazılım"}

Aynen varlıkların çıkarımında olduğu gibi bu ilişkiler sadece metin içeriği vasıtasıyla değil daha önceden analiz edilmiş ve yapısal olarak depolanmış bir veri

kümesinden çekilebilir.iPad örneğini ele alacak olursak Apple firmasının ürettiği elektronik cihaz olduğu bilgisine erişilebilir.

Örnekleri çoğaltmak gerekirse bir diğer haber detayını[4] da inceleyebiliriz.

<HaberMetin>

<Baslik> Yılın son konuşması <Varlik> Draghi</Varlik> 'den <Baslik><Oz><Varlik>AB Merkez Bankası</Varlik> (<Varlik>ECB</Varlik>) Başkanı <Varlik>Mario Draghi</Varlik>'nin bugün yapacağı konuşmada yatırımcıları rahatlatmaya yönelik açıklamalarda bulunması bekleniyor </Oz><Icerik><Varlik> AB Merkez Bankası</Varlik> (<Varlik>ECB</Varlik>) Başkanı <Varlik>Mario Draghi<Varlik/>'nin bugün yapacağı konuşma merakla beklenirklen piyasalar da <Varlik>ECB</Varlik>'nin 3 yıllık ucuz likidite planının sonuçlarını izliyor olacak.Küresel piyasalar Noel öncesi son dönemece girdi. Ancak özellikle kredi derecelendirme şirketlerinin <Varlik>Euro Bölgesi</Varlik> ülkelerine yönelik baskıları aralıksız sürüyor. Piyasalar yılın son günlerinde <Varlik>Fransa</Varlik>'nın da notu kırılanlar kervanına katılmasından endişe ediyor. <Varlik>AB Merkez Bankası</Varlik> (<Varlik>ECB</Varlik>) Başkanı <Varlik> MarioDraghi</Varlik>'nin bugün yapacağı konuşmada yatırımcıları rahatlatmaya yönelik açıklamalarda bulunması bekleniyor. Diğer yandan<Varlik>ECB<Varlik>'nin piyasalara sunacağı üç yıllık ucuz likidite imkânının sağlayacağı rahatlamanın da etkileri yakından izlenecek. AB'li yetkililer bu adımla<Varlik> Euro Bölgesi<Varlik>'nde giderek daha büyük bir sorun haline gelen bankacılık sorununu kontrol altına almayı umuyor. Gözler kredi notlarında Piyasalar <Varlik>Belçika<Varlik>'nın ardından yılın son günlerinde <Varlik>Fransa<Varlik>'ya yönelik bir not sürprizinden endişe ediyor.</Icerik>

</HaberMetin>

Şekil 3.2 (Varlıkların örnek metinde gösterimi)

[4] http://ekonomi.haberturk.com/finans-borsa/haber/698219-yilin-son-konusmasi-draghiden sayfasından elde edilmiştir

Verilen örnekte HaberMetin tagı metin başlangıcı ve sonunu belirtmektedir.Oz tagı haberin özünü Icerik tagıda haberin içeriğini belirtmektedir.Varlık işaretçileri ile belirlenen sözcük yada sözcük öbeklerinin bazıları ilişkiler yada olayların belirlenmesi amacıyla belirtilmiştir.

Varlık{isim=" Draghi " , tanım="AB Merkez Bankası başkanının soyadı" , tip="İnsan"}

Varlık{isim=" AB Merkez Bankası " , tanım=" Avrupa Birliği Merkez Bankası" , tip="Banka"}

Varlık{isim=" ECB " , tanım= "Avrupa Birliği Merkez Bankası'nın kısaltılmış adı" , tip="Banka"}

Varlık{isim=" Mario Draghi " , tanım=" AB Merkez Bankası başkanının adı " , tip="İnsan"}

Varlık{isim=" Euro Bölgesi " , tanım=" Avrupa Birliği'nde, para birimlerini Euro olarak değiştiren ülkeler." , tip="Yer"}

Varlık{isim=" Fransa " , tanım="Euro Bölgesindeki ülkelerden biri" , tip="Ülke"}

Varlık{isim=" Belçika" , tanım=" Euro Bölgesindeki ülkelerden biri " , tip="Ülke"}

Bu varlıkların tanımlanmasında insanlar yerine doğrudan makineleri kullanmak istersek doğrusal bir yaklaşım izlendiğinde önceden tanımlanmış varlıkları içeren bir veri kümesinden yararlanmak gerekebilir.Google gibi bir kaynaktan faydalanarak Draghi olarak tanımladığımız varlık ile ilgili olarak araştırma yaparsak Mario Draghi olarak çıkacaktır (Vikipedi 2011).

Tanımlanan bu varlıkların arasındaki bazı ilişkileri tanımlamak istersek

İlişki{varlık1="ECB" , varlık2="Avrupa Birliği" , tip="Banka"}

İlişki{varlık1="AB Merkez Bankası" , varlık2="Avrupa Birliği" , tip="Banka"}

İlişki{varlık1="Fransa ",varlık2="Euro Bölgesi",tip="Ülke"}

3.2. Biçimsel Tanım

Bilgi çıkarım süreci yukarıdaki örneklerde de görüldüğü gibi verinin anlamlı hale getirilmesi yaklaşımlarındandır. Bu noktada veri madenciliği ve metin madenciliği kavramlarından da bahsetmemiz gerekir.

Veri madenciliği mevcut veriden anlamlı bilgileri, ilişkileri çıkarmada kullanılan tekniklere verilen genel isimdir (Dolgun, Özdemir ve Oğuz 2009). Veri madenciliği ile yapısal veri analiz edilebilir. Ancak metin ve web madenciliği ile yapısal olmayan verinin veri madenciliğinde kullanılabilir hale getirilmesi yani yapısal hale getirtebilmesi mümkündür.

Veri madenciliği büyük veri yığınlarında gizli olan örüntüleri ve ilişkileri ortaya çıkarmak için istatistik ve yapay zeka kökenli çok sayıda ileri veri çözümleme yönteminin kullanıldığı bir süreçtir. Veri madenciliği algoritmaları; istatistiksel algoritmalar, matematiksel algoritmalar ve yapay zeka algoritmalarını (sinir ağları, karar ağaçları, kohonen ağlar, birliktelik kuralları vb.) bir arada içerir.

Veri madenciliği çözümleri ve algoritmaları metin veya web verisindeki kalıpları bulmadan veya model oluşturmadan önce metin veya web verisinin yapısal olması gerekmektedir. Metin ve Web madenciliği işlemleri, veri madenciliğinde kullanılacak yapısal veriye ulaşmak için kullanılan araçlar olarak tanımlanabilir. Metin madenciliği, çok büyük belgelerin analizi ve metin tabanlı verinin içerisindeki gizli kalıpların elde edilmesidir. Web madenciliği ise, web içerikleri, sayfa yapıları ve web bağlantı istatistiklerinin de içinde olduğu web ile ilişkili olan verinin analizini içermektedir

Klasik veri analiz yöntemleri verinin değişken ve kayıt bazlı düzenlendiği varsayımı ile işlem yapmaktadır. Eğer veri metin formatında ise, verinin anlamının ortaya çıkarılması için kullanılan yöntem metin madenciliğidir.

Bu yöntemler, hem sorgudaki hem de metindeki kelimelerin karakterlerini karşılaştıran bir temele dayanır. Bundan dolayı metnin içeriğine ait anlamsal veya

açıklayıcı sonuçlar elde etmek mümkün değildir. Dili anlamanın temeli dilbilimsel yollara dayanır ve çoğunlukla doğal dil işleme (DDİ) olarak ifade edilir. DDİ 'yi içeren bir sistemde, karmaşık yapıların bulunduğu ifadeler (örneğin; soğuk davranış ile havanın soğukluğu arasındaki fark gibi) akıllı olarak çıkarabilmekte ve terimleri sınıflayarak; ürünler, organizasyonlar veya kişiler gibi sınıflara atamaktadır.

Metin madenciliği süreci iki aşamada gerçekleşir;

a) Anahtar içerik/ifadeler metinden elde edilir,
b) Elde edilen içerik/ifadeler, yüksek dereceden ilişkili olduğu kategorilere atanır.

Bu aşamaları basit bir örnek üzerinden açıklamak gerekirse;

1. Aşama: "Kur" ve "Faiz oranı" ifadeleri metinden elde edilir,

2. Aşama: Bu iki ifade, otomatik olarak "Finans" etiketli kategoriye aktarılır.

Metin madenciliği uygulamaları iki ana sınıfta ayrılabilir:

Metnin anlaşılması/özetlenmesi: Metin madenciliğinin amaçlarından bir tanesi metinden anlamlı nitelikli bilginin çıkarılmasıdır. Böylece metnin içerdiği anahtar içerik anlaşılabilecektir. Örneğin,Euro bölgesindeki son durumu öğrenmek isteyebiliriz.

Metin ile modelleme: Daha yaygın olarak, metin madenciliği terk etme veya ürün alma gibi müşteri davranışlarının tahmin edildiği bir modelin geliştirilmesi aşamasının bir bölümünü oluşturmaktadır. Metinden elde edilen içerik girdi değişkeni olarak kullanılır ve diğer bilgiler ile beraber öngörüsel model geliştirilir.

Veri madenciliği sadece yapısal veriyi kullanabildiği için veri madenciliği çözümleri ve algoritmalarından faydalanılarak metin içerisindeki kalıplar bulunarak, model oluşturulmadan önce bilginin yapısal hale getirilmesi gereklidir. Metin madenciliğinin sonucu veri madenciliği için girdi oluşturmaktadır.

Metin ve veri madenciliği arasında etkileşimli bir ilişki vardır. Metin madenciliği sonucunda elde edilen yapısal veri, veri madenciliği modellerinde kullanılmakta ve elde edilen sonuçlar daha sonra metnin yapısının incelenmesinde kullanılmaktadır.

3.3. İleti Anlama Görüşmeleri(Message Understanding Conference MUC)

Bilgi çıkarımı sistemleri geliştirmek üzere bir çok proje hayata geçirilmiştir.İleti Anlama Görüşmeleri (Message Understanding Conferences (MUCs)), 1987 den itibaren 1998 e kadar yapılan bilgi çıkarımı konusunda gelişmelerin görüşüldüğü bir toplantıdır.İleti Anlama Görüşmeleri sayesinde bilgi çıkarım yaklaşımlarının, süreçlerinin ve teknolojilerinin karşılaştırılarak geliştirilmelerinde faydalı olmaktadır.

Yapılan görüşmelerde geliştiriciler ve araştırmacılar buluşlarını paylaştılar ve çeşitli gereksinimleri belirlediler. Aşağıda çizelge 3.3 ile geçmiş yıllarda yapılan projeler ve hangi alanda yapıldıklarına dair yılları ile belirtilen bulunmakta.

Çizelge 3.3(Yıllara göre İAG'deki gelişmeler)

Proje	Yıl	Alan
MUC-1	1987	Deniz Muharebe Harekatları
MUC-2	1989	Deniz Muharebe Harekatları
MUC-3	1991	Latin Amerika Ülkelerinde Terörizm
MUC-4	1992	Latin Amerika Ülkelerinde Terörizm
MUC-5	1993	Kurumsal Ortak Girişim ve Mikroelektronik
MUC-6	1995	Yönetim değişiklikleri haber makaleleri
MUC-7	1998	Uçak Kazaları / Roket Atışları

3.4. Otomatik İçerik Çıkarımı(Automated Content Extraction, ACE)

Otomatik İçerik Çıkarımı geliştirme çalıştayı, İleti Anlama Görüşmelerinin (MUC) sonucu ve devamı olarak 1999 dan itibaren otomatik içerik çıkarma teknolojilerini geliştirme ve destekleme amacıyla bir araya gelmiştir.

Genel olarak ileti anlama görüşmelerindeki şablon benimsenmiştir. Geliştirme süreçleri görevleri, geliştirme eğitim ve test verileri tartışılmakta ve çeşitli adımlar belirlenmektedir. Günümüze kadar belirlenen adımlar;

Varlık Saptamam ve Yakalama: Kaynak metinde bahsedilen varlıkların saptanması ve söz konusu oldukları yerlerin yakalanması
İlişki Saptama ve Nitelendirme: Yakalanan varlıkların aralarındaki ilişkilerin saptanması ve bu ilişkideki niteliklerinin belirlenmesi
Varlık Saptama ve Tanımlama: Kaynak metinde varlıkların saptanması ve niteliklerinin çıkarımı
İlişki Saptama ve Tanımlama: İlişki saptanması ve ilişkinin niteliklerinin belirlenmesi
Zaman Saptama ve Tanımlama: Metinde geçen geçici ifadelerin saptanması ve tanımlanması
Değer Saptama ve Tanımlama: Metinde geçen sözcüklerin metindeki anlamının değerinin saptanması(sayı değerinin para olduğunun belirlenmesi)
Olay Saptama ve Tanımlama: Olayların saptanması ve tanımlanması
Yerel varlık Saptama ve Tanımlama: Varlıkların saptanması ve tanımlanmasında metine özgü değerlerin göz önüne alınarak sürecin işleyişi
Yerel ilişki Saptama ve Tanımlama: Varlıklar arasındaki ilişkinin bulundukları metinin sınıfına göre saptanması ve tanımlanması
Genel Varlık Saptama ve Tanımlama: Varlıkların saptanmasında kaynak metinin niteliklerinin göz ardı edilmesi
Genel İlişki Saptama ve Tanımlama: Varlıklar arası ilişkileri saptarken metin özeliklerinin göz ardı edilmesi

İleti Anlama Görüşmelerinden Otomatik İçerik Çıkarımına geçerken ilgi alanına çoklu kaynak ve çoklu dilbilim sorunlarında girmiştir. Çalıştayda 1999 dan 2008 e

kadar kat edilen yolda gelişmeleri göstermek amacıyla çizelge 3.4 hazırlanmıştır (Tatar 2011).

Çizelge 3.4(Yıllara göre OİÇ gelişmeler)

Yıl	**Diller**	**Tanımlanan Görevler**
2000	İngilizce	VSY (Pilot)
2001	İngilizce	VSY, İSN
2002	İngilizce	VSY, İSN
2003	İngilizce, Çince, Arapça	VSY, İSN
2004	İngilizce, Çince, Arapça	VST, İST, ZST
2005	İngilizce, Çince, Arapça	VST, İST, ZST, DST, OST
2007	İngilizce, Çince, Arapça, İspanyolca	VST, İST, ZST, DST, OST
2008	İngilizce, Arapça	YVST, YİST, GVST, GİST

İçerik çıkarımı ile ilgili ilk sunuma 1950 de rastlıyoruz (Harris 1958). Raporlanmış olarak yapılan ilk çalışmaların 1950 sağlık alanında yapıldığı bilinmekte. Bilgi çıkarımı alanında süregelen gelişmeler,çalışmaları doğal dildeki metinler üzerine kaydırmıştır.

4. TÜRK DİLİ

Türkçe Avrupa ve Asya'da geniş bir coğrafyaya yayılmış, dünyada en çok konuşulan altıncı dildir. Ural-Altay dilleri ailesine, Altay kolunun bir üyesi olan Türkçe bitişken bir dil olup, sözcüklerin sonlarına ard arda çekim ve türetim ekleri eklenerek yüzlerce farklı sözcük oluşturmak mümkündür (Oflazer 1995) . Bu yapısı nedeniyle Türkçenin biçimi kullanılarak analizi diğer dillerle karşılaştırıldığında çok daha karmaşık olabilmektedir. Örneğin “bence” tek sözcüğü ile ifade edebildiğimizi İngilizcede "in my opinion" sözcük öbeği ile ifade edebiliriz.

İçerik çıkartmak için Otomatik içerik çıkarımında belirlenen aşamalar üzerinden doğrusal bir yaklaşımla, Türkçe metinlerde yer alan sözleri kullanarak metin içerinde bulunuşlarına dair istatistiksel elde edebiliriz. Bunu yapmak için Türkçe bir metinden çıkartmaya çalıştığımız anlam için sözlerin ve içlerindeki sözcüklerin ilişkilerinden faydalanabiliriz. Türkçede genel olarak sözü oluşturan sözcüklerin dizilişleri göz önüne alındığında çoğunlukla özne, nesne, yüklem kalıbına uyduğunu görebiliriz. Fakat bu kalıba her zaman uygun olmayabilmekte ve öğeler söz içerisinde serbestçe yer değiştirebilmektedir. Amacımız söz ün öğelerine ayrışmasını sağlamak ise klasik yöntemden faydalanabiliriz: yüklemi bulmak ve yükleme kim, ne, nerede, nasıl, niçin gibi sorular sorarak diğer öğelere ulaşmak.

Türkçede söz içerisinde öğeler özne, yüklem, nesne, tümleç ve zarf şeklinde temel öğeler olarak ya da ünlem ve bağlaç gibi yardımcı öğeler yada söz dışı öğeler olarak bulunabilirler.

Aşağıdaki insan eliyle bir öğelere ayırma işlemi örneklenmiştir.

‘Ben sporcunun zeki, çevik ve ahlaklısını severim.’ M.Kemal Atatürk

1.Yüklem : severim [Yüklem]

2.1.Kim sever : Ben [Özne]

2.2.Ne sever : sporcunun zeki çevik ve ahlaklısını [Nesne]

Bu sonuçları bilgi çıkarım konusunda da belirtildiği üzere belirlenen varlıkları ve sözdeki yükleme sorulan sorular ile aralarındaki ilişkileri ve olayları tanımlada kullanabiliriz.Sözün öğelerini belirleme sürecini doğrusal bir yaklaşımla tasarlamamız gerekirse

1.Yüklemi bul

2.Diğer temel öğeleri bul

Böylece her bir öğe bizim için yüklemle olan ilişkiyi tanımlamış olsun.Ve her bir söz için yapılan bu çıkarımlardan faydalanarak tam metin üzerinde çeşitli çıkarımlarda bulunabilelim.

Bir sözcüğün yada sözcük öbeğinin söz içerisinde yüklem olduğunu anlamak için sözdeki konumu bizim karar vermemizde yeterli değildir. Yüklem olabilme kurallarını incelediğimizde, yüklem bir eylem sözcükten oluşabildiği gibi bir tamlamanın eylemleştirici ekler almasıyla da oluşabilir.Bu konuyla alakalı yapılmış çalışmalardan elde edilebilecek bilgiyi bize gerektiği perspektif ile soyutladığımızda aşağıdaki tablo oluşturulabilir.

Çizelge 4.1 - Türkçe'de ekler

Çekim ekleri	Yapım Ekleri
Çokluk eki	
Hal ekleri	İsimden isim yapan yapım ekleri
İyelik ekleri	İsimden Eylem yapan yapım ekleri
Eşitlik eki	Eylemden isim yapan yapım ekleri
Ek eylem ekleri	

Bu tabloda bizim için ilk aşamada Ek eylem ekleri ve İsimden eylem yapan ekler önemlidir. Tüm metinde yer alan bütün sözcükler için teker teker köklerine ayırarak

bakmaya çalıştığımızda bu bölümlemenin bizim için yetersiz olduğunu ve atomik olarak ele almak üzere sözcükleri teker teker ele alınması yerine eğer bir tamlama söz konusu ise en küçük birim olarak tamlamayı oluşturan sözcük öbeğinin bir arada alınması gerektiğini görürüz.

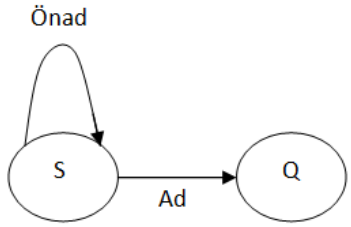
Bu durumda örneğimize dönecek olursak; yüklem olarak belirlediğimiz “severim “ sözcüğü eklerine ayrıldığında sev-er-im bir eylem kök, bir geniş zamanı belirten ek ve birde kişiyi belirten ek içermektedir.’Ben ’ sözcüğü olmasaydı bile yüklemin eklerinden kim sorusunun, yanıtını önceden tanımlamış eklerden bilebilirdik.

Türkçede tamlamaları inceleyecek olursak; yapısal olarak iki öbeğe ayrıldığını görürüz

- Ad Tamlamaları : ad önüne başka adlar gelerek oluşturulur
- Belirtili Ad Tamlaması :Tamlayan ve tamlanan ek almıştır
- Belirtisiz Ad Tamlaması : Tamlayan ad ek almaz
- Takısız Ad Tamlaması : Hem tamlayan hem tamlanan ek almaz
- Zincirleme Ad Tamlaması : iki yada daha çok ad tamlamasının iç içe bulunması ile oluşur
- Önad Tamlamaları : ad önüne önadlar getirilerek oluşturulurlar
- Niteleme Önad Tamlamaları : Önad nitelik belirtmektedir
- Belirtme Önad Tamlamaları :
- İşaret : Adların yerini gösterir ,çoğulları yoktur
- Sayı : Adları sayısal olarak belirtir
- Belgisiz : Adları kesinlik kazanmadan belirtir
- Soru: Adlardan önce gelerek adı soru yönünden belirtir
- Unvan: Kişi adlarından önce bulunarak saygı yada tanıtma yönünden betimler.

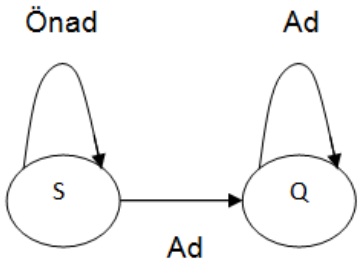
Yukarıda belirtilen özellikleri soyutlayarak bir tamlamayı tanımlayabilirsek eğer söz üzerinde ele alacağımız en küçük parçaları da kesin olarak saptayabilmiş

oluruz.Önad tamlamaları çeşitli araştırmalarda ATN[5] (genişletilmiş geçiş ağı) modeli şekil 4.1 deki gibi belirtilmiştir.



Şekil 4.1 - Önad Tamlaması

Şekil 4.1 ve şekil 4.2 de belirtilen S başlangıç durumunu Q son durumunu ifade eder. Şekil 4.1 deki bu belirtime göre en az 0 adet önad bir adın önüne gelerek bir önad tamlaması oluşturur. Oluşan öbeğin yani tamlamanın bir ad olduğu varsayılırsa bu öbeğin bir ad önüne gelmesi ile bir ad tamlaması oluşturduğu görülür. İfade etmede genişletilmiş geçiş ağlarından faydalanabiliriz.



Şekil 4.2 - Türkçede Tamlama

Tamlamaları bilgisayar tarafından tanımlayabilmek için kuralları belirlemeye çalışırsak yukarıdaki başlıkların soyutlama düzeyinde özeliklere bakacak olursak;

[5] ATN Genişletilmiş Geçiş Ağları (Augmented Transition Network),dili matematik model olarak ifade etmek üzere ,Woods tarafından 1970 ve 1973 yılları arasında geliştirilmiş bir yaklaşımdır.

En az iki adın, aralarında anlam bağlantısı kurarak oluşturduğu, bir nesnenin parçası olduğunu ya da bir nesnenin başka bir nesneyle tamamlandığını gösteren ad takımıdır.

Ünlüyle biten sözcüklere tamlayan eki getirildiğinde araya “n”; tamlanan eki getirildiğinde araya “s” kaynaştırma ünsüzü gelir. Örnek: elmanın yarısı

Bazı durumlarda “-ın, -in, -un, -ün” tamlayan eki araya “n” değil “y” alarak kaynaşır. Örnek: suyun sesi, neyin nesi

- Tamlayanla tamlananın yeri değişebilir
- Tamlayan ya da tamlanan birden çok kullanılabilir
- Tamlayan ve tamlana sözcük öbekleri olabilirler
- Tamlanan önad tamlaması olabilir
- Kimi durumlarda tamlayan ekinin yerini “-den, (-dan)”durum ekleri alabilir
- Tamlayan eki ya da ilgi hâl eki -(n) in , Tamlanan eki, daha doğrusu iyelik ekleri -(s/y) i ad tamlamalarına özgü eklerdir.

Eylem Olabilecek sözcükler ve sözcük öbekleri

Söz içerisinde varlıkların yaptıkları ve ya etkilendikleri işleri, hareketleri, oluşları, kılışları, durumları zamana ve kişiye bağlı olarak anlatmada kullanılan sözcük ya da sözcük öbeğidir. Yapısına göre iki başlıkla inceleyebiliriz;

- Basit eylemler: Yapım eki almamış, bir tek sözcükten oluşan eylemlerdir. Eylem kökten oluşları tanımlamayı kolaylaştırır
- Türemiş Eylemler: Ad ve ya eylem kökleri ile yansımalardan, yapım ekleri kullanılarak türetilmiş eylemlerdir.

Eylemin tanımının yapılması varlıklara dair olayların tanımlanmasında yardımcı olabileceği gibi metindeki ‘kim’ ve ‘ne’ sorularının yanıtını almada yardımcı olacaktır. Eylem olabilmenin belirlenmiş kurallarına bakacak olursak;

- Yapım eki almamış eylem sözcük kökünden oluşmuş olabilirler
- Ad soylu köklerden aşağıdaki ekleri alarak oluşmuş olabilirler
- -e,-i,-a(l) eki alarak : az-al-mak, düz-el-mek, kör-el-mek, doğru-l-mak

- -la,-le eki alarak : ot-la-mak, yem-le-mek, baş-la-mak, yavru-la-mak
- -laş,-leş eki alarak : haber-leş-mek, mektup-laş-mak, güzel-leş-mek
- -ar,-er,-r eki alarak : baş-ar-mak, mor-ar-mak, kara-r-mak
- -a,-e eki alarak : yaş-a-mak, kan-a-mak, tün-e-mek
- -sa,-se eki alarak : benim-se-mek, su-sa-mak
- -da,-de eki alarak : gümbür-de-mek, takır-da-mak, hırıl-da-mak, inil-de-mek, şırıl-da-mak
- -kir,-kır,-kur,-kür eki alarak : püs-kür-mek, hay-kır-mak, fış-kır-mak, hıç-kır-mak
- Eylem soylu köklerden aşağıdaki ekleri alarak oluşmuş olabilirler
- -(a)la,-(e)le eki alarak : eş-ele-mek, kov-ala-mak
- -(i) eki alarak :sür-ü-mek, kaz-ı-mak
- -(i)l eki alarak : dik-il-mek, yak-ıl-mak, üz-ül-mak
- -(i)n eki alarak : sil-in-mek, kaç-ın-mak, gör-ün-mek
- -(i)ş eki alarak : gir-iş-mek, kız-ış-mak, böl-üş-mek
- -(i)t eki alarak : eri-t-mek, oyna-t-mak, yürü-t-mek
- -d(i)r eki alarak : çiz-dir-mek, yaz-dır-mak, ör-dür-mek, aç-tır-mak, kes-tir-mek
- Eylem soylu sözcükten aşağıdaki ekleri alarak adlaşmış olabilirler
- -me eki alarak : karma, umursama
- -mek eki alarak : gülmek, yemek, çakmak
- -iş eki alarak : özleyiş, duyuş, bakış
- -(i)m eki alarak : uçurum, geçim, ölçüm
- -i eki alarak : yazı, sayı, doğu
- -k eki alarak : yanık, açık, tarak
- -gi eki alarak : sevgi, uy{u)ku, örgü
- -ek eki alarak : erek, bıçak, korkak
- -gin eki alarak : kırgın, yetişkin, solgun
- -(i)n eki alarak : ekin, yığtn, akm
- -gen eki alarak : unutkan, çahşkan, sıkılgan
- -geç eki alarak : yüzgeç, süzgeç, utangaç
- -ti eki alarak : bulantı, çalkantı, yaşantı
- -inti eki alarak : esinti, sıkıntı, üzüntü
- -(i)t eki alarak : yakıt, geçit, umut

- -enek eki alarak : gelenek, olanak, tutanak
- -ici eki alarak : boğucu, yakıcı, yıkıcı

Bu kurallara dayanarak metin içerisindeki bir sözcüğün ya da bir tamlamanın eylem olup olmadığı anlaşılabilir.

5. İÇERİK ÇIKARIMI

5.1. Varlık Çıkarımı

Varlık çıkarımı Bilgi Çıkarım Sürecinde temel bir alt görevdir. Belirlenen metin içerisinde bulunan kişi adları, kurum adları, yer adları, zaman, para ve kısaltmalarını adlandırma ve sınıflandırma işlemleridir. Bu işlem sonucunda metin içerisinde bulunan varlıklar saptanır ve nitelikleri elde edilir.

Yapılan çalışmalar sonucu Otomatik İçerik Çıkarımı sürecinde varlık tanıma görevi varlıkları 5 temel sınıfa ayrılmıştır (Zhou, et al. 2009).Bunlar;

- İnsanlar
- Kurumlar
- Yerler
- Araçlar(Hizmetler)
- Coğrafi konumlar(Ülkeler, şehirler gibi)

Belirlenen bu sınıflar ile beraber varlıkların bazı nitelikleri de çıkarılmaya çalışılmaktadır. Bu alanda yapılmış çalışmalar ile dilden bağımsız olarak varlıkları tanıyabilen sistemler geliştirilmeye çalışılmıştır(Cucerzan ve Yarowsky 2009). Sözlük kullanılmadan varlığın çıkarımı için bazı çalışmalar mevcuttur(Mikheev, Moens ve Grover 1999).İngilizce için yapılmış çalışmaların Türkçeye uygulanmasıyla elde edilen sonuçları içeren çeşitli araştırmalarda mevcuttur(Tür, Hakkani-Tür ve Oflazer 2003).

Dalkılıç, Gelisli ve Diri (2010) tarafından yapılan bu çalışmaların bir çoğu incelenmiş ve bazı çıkarımlar yapılmış ve basit bir model geliştirilmiştir. İşlenen Türkçe metin üzerinde varlıkların çıkarımı için sözcüğün büyük harfle başlamasından yararlanılır. Her büyük harfle başlayan sözcük varlık olma olasılığı taşır ve geçici olarak elde tutulur. Metin içerisinde tekrar karşılaşma sıklığına göre bu sözcüğün ya da sözcük öbeğinin varlık oluşuna karar verilir. Varlık oluşa karar vermede metin içinde bulunmanın yanı sıra çeşitli bir sonraki aşamada varlığın sınıflandırılması adına bazı işlemler gerçekleştirilir. Bu çalışma ile varlığın

sınıflandırılmasında 3 sınıf temel olarak alınmıştır, Yer Adları, Kişi Adları, Kurum Adları. Koyulan diğer kurallara aşağıda değinilmiştir

Yer adlarının tanımlanmasında sözlüklerden faydalanılır. Coğrafi isimleri belirlemede kullanılacak sözlüklerin çok küçük olması sözlüklerin kullanılmasında temel etmendir.
Kurum Kuruluş adları bulmada yine sözlükler oluşturulmuştur. Bunun yanı sıra kuruluşların sonlarına aldıkları A.Ş ve LTD. gibi kısaltmalardan faydalanılmıştır. Bunların dışında kısaltmaların kurum ismi olma olasılıklarından faydalanılmıştır
Kişi adları tanımlamada sözlük yardımı alınmamış bunun yerine saptamalar 2 öneride bulunulmuştur. Bunlardan birincisi Türkçede unvanların kişi adlarından önce gelebildiği görülmektedir. Gazeteci Yazar Abdi İpekçi, Prof. Dr. Mehmet Haberal gibi örnekler verebiliriz. Bunun yanı sıra unvanlar kişi adlarından sonrada gelebilir. Ahmet Bey, Selda Hanım gibi. İkinci yöntem ise sözde yüklemin belirttiği kişiden faydalanılmasıdır.

Yapılan çalışma sonucunda birtakım kısıtlar saptanmış ve aynı metinde yer verilmiştir.

Varlık çıkarımı için yapılan bir başka çalışmada eylemleri tanıma ve onların üzerinden bazı kurallar ile varlıkları çıkarma yaklaşımıdır(Adalı ve Sönmez Haziran 2011).Bu çalışmada Türk Diline dair belirttiğimiz çıkarımlarda yer alan kuralların bazılarından faydalanarak biçimsel analiz yapılmış ve eylem olabilecek sözlük bulunmaya çalışılmıştır. Ontoloji tabanlı bilgi çıkarımı ve belge yapı analizi teknikleri bir arada kullanılmıştır.

5.1.1. WordNet nedir?

Varlık çıkarımında önemli çalışmalardan birisi de WordNet'tir. Her dilde kullanılan sözcüklerin biçimleri yerine anlamlarının bir araya getirilmesini sağlamak üzere tasarlanmış bir veri yapısıdır.Kavramsal bir sözlük gibi düşünebileceğimiz wordnet'in diğer sözlüklerden farkı eş anlamlı sözcükleri, karşıt anlamlı sözcükleri barındırmasının yanı sıra hiyerarşik olarak alt-kavram ve üst-kavram belirtmesiyle ilişkilerini tanımlamasıdır.WordNet'in bir diğer özelliğide bu yapıyı dilden bağımsız olarak gerçekleştiriyor olmasıdır.

WordNet ilk olarak Princeton Üniversitesinde İngilizce için tasarlanmış WordNet Projesi ile ortaya çıkmıştır. Projede bahsedildiği üzere Çizelge 5.1 ile belirtilen veri yapısı tasarlanmıştır (Miller 1990).

Çizelge 5.1 - WordNet kavramları

Üst Kavram(Hyperonym)	Bütünün Parçasi(Mero Portion)
Alt Kavram(Hyponym)	Alt Olay(Subevent)
Bölümün Bütünü(Holo Part)	Olayidir(Is Event Of)
Bütünün Bölümü(Mero Part)	Sebep Olur(Causes)
Üyenin Bütünü(Holo Member)	Sonucudur(Is Caused By)
Bütünün Üyesi(Mero Member)	Durumundadir(Be In State)
Parçanın Bütünü(Holo Portion)	Durumudur(State Of)
Yaklaşık Zıtanlamlı(Near Antonym)	Yaklasik Esanlamli(Near Synonym)

Anlam tiplerinin yanında sözcüklerin önad, nesne, zamir, eylem ilişkilerine de yer verir. Wornet Ontolojisi ise kısaca sözcüklerin bağları ile belirlenen ilişkilerden oluşturulan yapıdır. WordNet'in ontoloji olarak tanımlanması Anlamsal Ağ En İyi Çözüm Geliştirme Grubu[6] tarafından 2004 yılında tanımlanmıştır(The World Wide Web Consortium 2007).Bu veri yapısı ile terimin aile bağlarını sorgulanabilinir. Bunun anlamı sözcüğün kökenbilimi (etimoloji) incelemenin yanı sıra bağlarını sorgulamaktır.

okul{bina, oda, duvar, ev, yapı, öğrenci, öğretmen}

Şekil 5.1 (Varlık bağları)

[6] Semantic Web Best Practices and Deployment Working Group:Anlamsal Ağ teknolojilerini geliştirmek ve desteklemek amacıyla w3c tarafından toplanan bir çalıştaydır. http://www.w3.org/2001/sw/BestPractices/

WordNet ile elde edilen veri tabanının Türkçeye çevirtilerek Türkçe WordNet'in oluşturulması ve ya aynı veri yapısında Türkçe bir veritabanı oluşturulması üzerine çeşitli araştırma ve geliştirmeler yapılmış ve halen devam etmektedir.

Bu çalışma ile ilgilenilen ilişki tanımlama ve saptama işlemi için varlıkların çıkarımı, ilişkilerin çıkarımına bir ön çalışma olarak yapılırsa belirlenen varlıkların arasındaki ilişkiyi 3 düzeyi temel almıştır (Zhou, et al. 2009).

- Adlar
- Nominal ifadeler
- Zamirler

5.2. İlişki Çıkarımı

Bizim asıl ilgimiz Otomatik içerik çıkarım ın temel görevlerinden İçerik çıkarımıdır. İlişki çıkarımı tanımlanmış varlıklar arasında çeşitli yönlerden ilişkilerin bulunması işlemleridir. İlişki çeşitli düzeylerde incelemelerle yapılıyor olabilir. Türkçe bir metin içerisinde bulunan sözcüklerin ele alınabileceği gibi, belirli bir alanda olduğu bilinen metinler içerisinde de ilişkiler tanımlanabilir. Bizim çalışmamızda metinler içinde bulunan sözlerden, öğelerine yarıştırma yoluyla ve metin içindeki sözcüklerin tiplerinden faydalanarak sözcük çiftlerinin benzer alanda bir arada bulunuşlarından faydalanarak ilişki tanımlama olmak üzere iki araştırma yapılmıştır.

Otomatik İçerik Çıkarımı İlişki tanıma ve niteleme çalışmamızda kılavuz olarak kullandığımız çalışmadır. Varlık tanıma işlemi sonrası 5 ana sınıfta toplandığından bahsetmiştik, kişiler, kurumlar, yerler, araçlar, coğrafi konumlar. İlişki tanıma bu varlıklar arasında açık ve ya saklı bir ilişki varsa bu ilişkiyi tanımlama işlemidir.

Yapılan çalışmalar arasında Zhou, et al. (2009) 'ın yaptığı Özellikler Yardımı ve Destek Vektör Makinesi Algoritması kullanılarak içerik çıkarımı İngilizce üzerine yapılmış bir örnektir. Çalışmaları baz ifade parçalama yöntemi üzerine kurgulanmıştır. Aynı zamanda WordNet gibi araçların anlamsal ilişkilerin

çıkarımında kullanımına bir örnek olmuştur. Yaklaşımları aynı zamanda bizimde Türkçe için uyguladığımız ikinci yaklaşıma kılavuz olmuştur. Sözcüklerin yerlerine göre çeşitli özellikler belirlenmiştir. Metin içerisinde varlık olduğu saptanmış sözcük yada sözcük öbeğinin ilişkilendirilecek olan ile metin içerisindeki konumları ile ilişkili olup olmadıklarına dair sınıflandırma yapılmaya çalışılmıştır.

Tatar (2011) ise yakın zamanda Türkçe bilgi çıkarımının tüm süreçlerine dair yapılmış önemli bir diğer çalışmadır. Çalışmada kurumlar, kişiler ve yerler olarak tanımlanmış varlıklar arasında 3 sınıfta ilişki tanımlanmıştır. Bunlar;

- İçinde Bulunma : Bir kurumun bulunduğu yeri ilişki olarak tanımlar
- İle Bağlantılı Olma : Bir kişinin bir kurum ile bağlantılı olması ilişkisini tanımlar
- Tarafından Saldırılma : Bir kişinin yada kurumun başka bir kişiye yada kuruma saldırma ilişkisini tanımlar

Bu ilişkileri tanımlamada söz düzeyinde araştırma yapılmıştır.Tarafından saldırma alan bağımlı bir ilişki diğerleri genel kullanılabilecek ilişkilerdir.

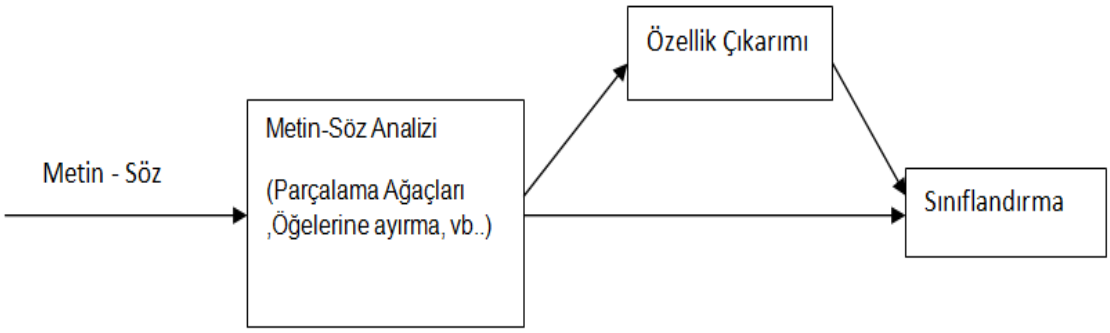
Asıl hedef ilişkileri bir öbek varlık üzerinde tanımlamaktır. Yakında yapılmış çalışmalar varlıkları ikili öbekler olarak ele alarak ilişkiyi tanımlamak daha sonra tanımlanmış ilişkiye dair bütün varlık öbeğini değerlendirmek üzerine kurgulanmaktadır.

Yapılan çalışmaları 3 öbeğe ayırarak incelemek mümkündür.

- Gözetmenli yaklaşımlar
- Özellik Tabanlı
- Çekirdek Tabanlı
- Geçişli
- Yarı Gözetmenli yaklaşımlar
- Önyüklemeli yaklaşımlar
- Dipre, Kartopu, KnowItAll, TextRunner
- Yüksek seviyeli ilişki tanımlama(Tümleşik Kullanım)

Gözetmenli yaklaşımlar sorunu bir sınıflandırma sorunu olarak ele alırlar. Verilmiş ilişkili ve ilişkisiz örnek verileri üzerinden iki varlığın ilişkili oluşuna dair sonuç elde edilmeye çalışılır. Destek vektör makineleri, perceptron ysa, logaritmik doğrusal model gibi algoritmalar kullanılarak ikili sınıflandırma yapılır.

Özellik tabanlı yaklaşımlarda, algoritmalarda kullanılmak üzere sözün ve sözcüğün birtakım özellikleri kullanılabildiği gibi sözün yapısal (sözün öğeleri,ağaç yapısı) sunumlarıda kullanılabilir.Genellikle akışın tepeden görünüşü Şekil 5.2 deki gibidir.



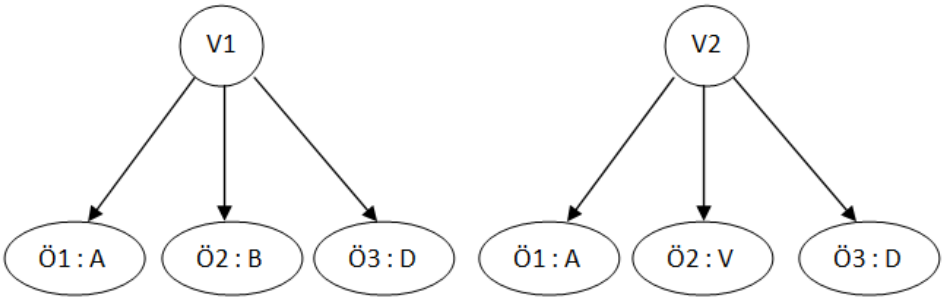
Şekil 5.2 - Metin analizi ve sınıflandırma

Zhao (2005) 'da da bahsedilen şekliyle

- Söz üzerinde gerekli analizler yapılır(Çözümleme ağaçları)
- Özellik çıkarımı yapılır
- Varlıklar arasında kalan sözcükler
- Varlıkların sınıfları
- İki varlık arasında bulunan sözcükler sayısı
- İki varlığı bölen sözcükler
- Çözümleme ağacında iki varlık arasındaki yol

Bunun yerine sözcüklerin kök bilgisinden de yararlanılabilinir. Sözcük bağlamı da ilişkiyi tanımlamada güçlü bir etmendir.

Çekirdek tabanlı yaklaşımlar da ise ağaçlar kullanılır. Nesneler arasındaki benzerlik kullanılır. Sözcüklerin genişletilmesinde kullanılan kurallar ve kök bilgisi kullanılarak sınıflandırma yapılmaya çalışılır. Kurulan ağaç yapılarında kök sözcüğün kendisi yapraklarda öznitelikleri ile oluşur. İki varlığın karşılaştırılmasında uyuşan öznitelikler sayısından faydalanılır eşleşen her öznitelik için sayaç bir arttırılır, eşleşmeyen her öznitelik için sayaç bir azaltılır.



Şekil 5.3 - Çözümleme ağacı ve Nitelik eşleştirme

Bunescu (2005)'de uygulanan şekli ise bağımlılık ağacında en kısa yol algoritması yeterli görülmüştür.En kısa yol geçişindeki her sözcük sayılarak sonuca varılmaya çalışılmıştır .

İki yordam Bach ve Badaskar (2007) aşağıdaki Çizelge 5.2 de karşılaştırılmıştır.

Çizelge 5.2 - Özellik tabanlı yöntem ve Çekirdek tabanlı yöntem karşılaştırma

	Özellik-Öznitelik tanımlama	Hesaplama Karmaşıklığı
Özellik Tabanlı yöntem	Metinin analiziyle birtakım özeliklerin çıkartılması gerekliliği	Daha Düşük
Çekirdek Tabanlı yöntem	Metin analiziyle önceden bir özellik çıkarım işlemi gerektirmez. Benzerlikler hesaplanır	Daha Yüksek

Bizim çalışmamızda 2 temel yaklaşım ortaya konmuştur.

1. Sözün öğelerinden faydalanarak belirlenmiş varlıklar arasında ilişkinin tespit edilmesi
2. Sözcüklerin kök ve eklerinden faydalanarak birbirleri arasında bir ilişkinin var olup olmadığının tespit edilmesi

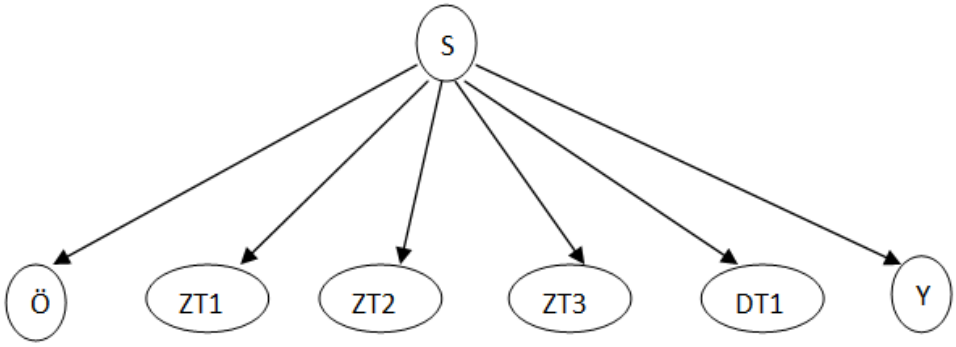
Her iki yaklaşımda da metinde yer alan sözler incelenecek düzey olarak belirlenmiştir.

5.2.1. Sözlerin öğeleri ile ilişki tanımlama yaklaşımı

İlk yaklaşımımız basit olarak metinde bulunan sözlerin önce öğelerine göre çözümleme ağacının oluşturulması ve oluşturulan ağaç üzerinde bilinen varlıkların arasındaki ilişkiyi bulmak üzere en kısa yol algoritması kullanılarak çıkarımlarda bulunmaya dayanmaktadır. Yapılan bu çıkarımlar metin içerisinde diğer sözler için aynı işlemin yapılması ile aynı varlık ikilisi arasında ilişki olup olmadığına dair bir sonuç bulmaktadır.

Sözün öğelerine ayrıştırılarak oluşturulacak bir çözümleme ağacı bölüm 2 de verdiğimiz gazete haberinden alınmış bir söz ile örneklenecek olursa aşağıdaki gibi olacaktır.

- Sözcük (S) : Hugo Retro Mania, 15 Aralık 2011 tarihinde on yediden fazla dilde iPad, iPhone, iPod, Android, PlayStation PSP Minis ve PC için piyasaya sürülüyor.
- Yüklem (Y) : sürülüyor
- Özne (kim,ne) (Ö) : Hugo Retro Mania
- Dolaylı tümleç (nereye) (DT1) : piyasaya
- Zarf tümleci (ne için) (ZT1) : iPad, iPhone, iPod, Android, PlayStation PSP Minis ve PC
- Zarf tümleci (ne zaman) (ZT2) : 15 Aralık 2011 tarihinde
- Zarf tümleci (nasıl) (ZT3) : on yediden fazla dilde



Şekil 5.4 - Örnek çözümleme ağacı

Oluşan ağaçta ilişkilerin saptanması amacıyla “Hugo” ve “iPhone” varlıklarını ele alacak olursak özne ile ilk zarf tümleci arasındaki en kısa yol 2 birim olduğu görülür. Örnekte verilen söz basit bir söz olup iç içe sözler veya tamlama gibi bir yapıda değildir.Aynı metinden alınmış daha karmaşık olan bir söz ile örnekleme aşağıda verilmiştir.

S0: “Çoğu insan Hugo ’nun Güney Amerika, Almanya, İspanya, Fransa, Rusya ve Türkiye gibi ülkelerdeki şöhretini duyduğunda şaşırıyor” şeklinde konuşan KREA Medie genel müdürü Henrik Kølle, sözlerini şöyle sürdürdü: “Ancak biz tecrübesiz değiliz ve bugün rekabetin her zamankinden daha şiddetli olduğunun farkındayız. Bu nedenle Hugo Retro Mania’dan beklentimiz, gerçekten de küresel bir marka olarak yeniden yaratılmasını sağlayacak pek çok heyecan verici adımın ilki olması yönünde”.

S1: Çoğu insan Hugo ’nun Güney Amerika, Almanya, İspanya, Fransa, Rusya ve Türkiye gibi ülkelerdeki şöhretini duyduğunda şaşırıyor

S2: Ancak biz tecrübesiz değiliz ve bugün rekabetin her zamankinden daha şiddetli olduğunun farkındayız. Bu nedenle Hugo Retro Mania’dan beklentimiz, gerçekten de küresel bir marka olarak yeniden yaratılmasını sağlayacak pek çok heyecan verici adımın ilki olması yönünde

S3: Ancak biz tecrübesiz değiliz ve bugün rekabetin her zamankinden daha şiddetli olduğunun farkındayız

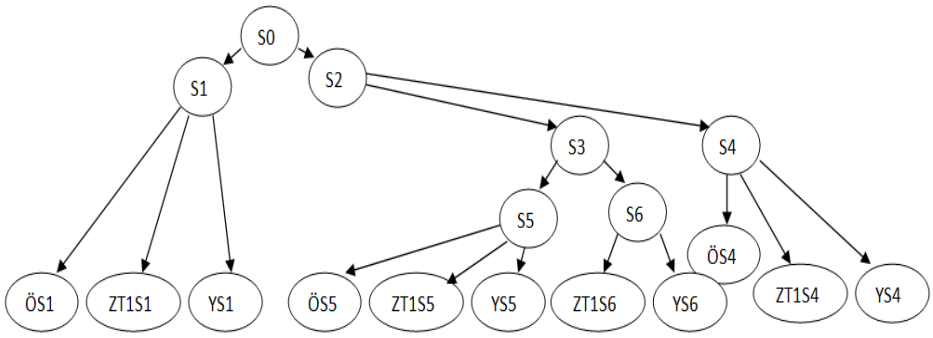
S4: Bu nedenle Hugo Retro Mania'dan beklentimiz, gerçekten de Hugo 'nun küresel bir marka olarak yeniden yaratılmasını sağlayacak pek çok heyecan verici adımın ilki olması yönünde

S5: Ancak biz tecrübesiz değiliz

S6: bugün rekabetin her zamankinden daha şiddetli olduğunun farkındayız

- Temel sözde yüklem (YS0) : sürdürdü
- Temel sözde özne (ÖS0) : "Çoğu insan Hugo 'nun Güney Amerika, Almanya, İspanya, Fransa, Rusya ve Türkiye gibi ülkelerdeki şöhretini duyduğunda şaşırıyor" şeklinde konuşan KREA Medie genel müdürü Henrik Kølle
- Temel sözde zarf tümleci (S2): "Ancak biz tecrübesiz değiliz ve bugün rekabetin her zamankinden daha şiddetli olduğunun farkındayız. Bu nedenle Hugo Retro Mania'dan beklentimiz, gerçekten de Hugo 'nun küresel bir marka olarak yeniden yaratılmasını sağlayacak pek çok heyecan verici adımın ilki olması yönünde"
- İlk sözde yüklemi (YS1) : şaşırıyor
- İlk sözde öznesi (ÖS1) : Çoğu insane
- İlk sözde zarf tümleci (ZT1S1) : Hugo 'nun Güney Amerika, Almanya, İspanya, Fransa, Rusya ve Türkiye gibi ülkelerdeki şöhretini duyduğunda
- Dördüncü sözde yüklem (YS4) : yönünde
- Dördüncü sözde özne (ÖS4) : Hugo Retro Mania'dan beklentimiz
- Dördüncü sözde zarf tümleci (ZT1S4) : küresel bir marka olarak yeniden yaratılmasını sağlayacak pek çok heyecan verici adımın ilki olması
- Beşinci sözde yüklem (YS5) : değiliz
- Beşinci sözde özne (ÖS5) : biz
- Beşinci sözde zarf tümleci (ZT1S5) : tecrübesiz
- Altıncı sözde yüklem (YS6) : farkındayız
- Altıncı sözde zarf tümleci (ZT1S6) : bugün rekabetin her zamankinden daha şiddetli olduğunun

Sözleri yukarıdaki şekilde öğelerine ayırmış olursak çözüm ağacı aşağıdaki gibi olacaktır.



Şekil 5.5 - Örnek çözümleme ağacı

Oluşturduğumuz çözümleme ağacı üzerinden “Hugo” varlığını ve “marka” varlığını saptamış olsak ikisi arasındaki en kısa yolu 5 olarak puanlayabiliriz.

Yaklaşımımızı gerçekleştiren bir sistem kurulabilmesi için öncelikle ilk adımı gerçekleştirmede kullanılmak üzere Türkçe sözleri öğelerine ayırabilen bir kütüphaneye ihtiyacımız vardır. Türkçe sözlerin öğelerine ayrılması işlemi doğal dil işlemede başlı başına bir sorun olarak ele alınmaktadır . Yapılan taramada bu işi yapma üzerine halen devam eden birçok projeye rastlanmıştır. Bunlardan bazıları Özyurt ve Köse (2006) da bulunduğu gibi çözümleme ağaçları oluşturmada tamlamaların ve sözcük öbeklerinin kullanımı ya da Çabuk, Yüksel, Mocan, Diri ve Amasyalı(2003) da yer aldığı şekliyle sözleri ayrıştırabilen kütüphanelerdir.

Öğelere ayırarak çözümleme ağacı oluşturulması yöntemini uygulayamadığımızdan diğer yöntem geliştirilmiştir

5.2.2. Sözcüğün yapısal (morfolojik) incelemesi ile ilişki tanımlama yaklaşımı

İkinci yaklaşımımız metin içerisindeki sözcüklerin kök ve ekleri göz önüne alınarak bazı özelliklerinin çıkartılması ve kök ek birimlerini ikili olarak karşılaştırma ile destek vektör makinelerinden yardım alarak ilişkinin var olduğuna dair tahminde bulunmayı hedefler . Bu amaçla Türk dilinde sözcüklere dair yapılan araştırma sonucunda en küçük birim olarak sözcük ve tamlama olmuş sözcük öbekleri alınmıştır. Yapılan çalışmada metin içerinde bulunan en küçük birimlerin saptanmasında Türk Dil Kurumu genel ağ sisteminde kullandığı sorgu sayfalarından ve zemberek adındaki açık kaynak kodlu geliştirilmiş kütüphaneden faydalanılmıştır. Metinlerin elde edilmesinde Genel ağda yer alan dizin yapısı saptanmış gazetelerden ve bunların finansal, kültürel, gündem ve ya teknoloji gibi sınıflandırılmış olmalarından yararlanılmıştır. Oluşturulması tasarlanan veri yapısına Yıldız Teknik Üniversitesi Kemik öbeği tarafından Türk Dil Kurumu ve Vikipedi den yararlanarak oluşturdukları, sözcüklerin eş anlamlılık zıt anlamlılık gibi ilişkilerinin tanımlandığı veri kümesi de birimlerimiz üzerinde yapılan bazı analiz safhalarında kullanılmak üzere eklenmiştir.

Sözü parçalara ayırarak incelemek istediğimizde ya bir önceki yöntemde olduğu gibi öğelerine ayrıştırarak tanımlamalarından faydalanırız yada en küçük birim olarak sözcükleri alırız. Türkçede sözcük tiplerini incelediğimizde sözcüklerin ancak ad oldukları durumda ve açık olarak belirtilmesi durumunda varlık olarak saptanabilecekleri çıkarımında bulunabiliriz. Önadlar bizim için bir ad önüne geldiklerinde onun niteliğini belirleyerek (tamlama olarak) yine bir ad olurlar. Adıllar kişi adları yerine kullanılırlar ve söz içerisinde örtülü olarak belirtimde bulunmuş olurlar. Belirteçler bir eylem sözcüğün önüne gelerek, ilgeçler ise tek başlarına anlamı olmayan sözcüklerdir. Bağlaçlar sözleri bağlamada ve ünlemlerde sesleniş veya duygu belirtimleri oldukları için varlık olarak tanımlanmayacaklardır. Türkçe de sözcüklerin bu bilgilerinden yola çıkılarak metinlerden sözler ve sözlerden sözcükler elde edilebilinir. Elde edilen parçalar ilk olarak aşağıdaki gibidir.

- Adlar: Varlık olabilecek birimler
- Önadlar: Bir adın veya bir tamlamanın önüne gelerek niteleyici olabilecek sözcükler

- Adıllar: sözün metin içerisinde bulunan bir varlığa dair bir ifadede bulunabileceği ihtimaline rağmen hesaplamalar dışı bırakılmış sözcükler
- Belirteçler: Bir eylemi niteleyebilecek sözcükler
- İlgeçler: hesaplama dışı sözcükler
- Bağlaçlar: sözü ikiye ayırmada kullanılan sözcükler
- Ünlemler: hesaplama dışı sözcükler

Sözün elde edilmesinden sonra uygulanan işlem yukarıdaki sonuçları üretmek üzere tasarlanmıştır. Türkçe WordNet uygulaması halen geliştirilmekte olduğundan ve varlıkların hali hazırda belirlendiği bir veri tabanına sahip olmadığımızdan elde edilen her bir ad bizim için potansiyel varlık olarak değerlendirilmiştir. Elde edilen her ad aynı söz içerisinde bulunan bir diğer ad ile bir arada bulunuşlar, bulundukları anda Zhou, Su, Zhang ve Zhang (2009) da uygulandığı gibi önce bulunma, aralarında başka sözcüklerin bulunması, aralarında başka adların bulunması alınan metinin alındığı yere göre sınıfı gibi özelliklerinin birarada bulunduğu ve metinlerden bağımsız bir sayaç olarak arttırdığımız frekansların belirlendiği veri kümemize işlenir. Frekansların tutulduğu bu yapıdan çeşitli perspektifler elde edilebilinir. Bunlardan bazıları sonuçlarda gösterilmektedir. Oluşturulan bu veri kümesinden destek vektör makinesine eğitimde kullanmak üzere sonuçlar türetilir ve bu adlar arasında eğer bir varlık olarak tanımlansaydılar bir ilişki olup olmayacağına dair sonuç elde etmesi beklenir.

6. UYGULAMA

6.1. Genel Yapı

Saptanan varlıkların birbirleriyle ilişkili oluşlarını ve ilişki özelliklerini belirlemek üzere Türkçe metinlerden faydalanılması planlanmıştır. Bu maksatla üzerinde çalışılacak veri kümesi hazırda bulunamamıştır ve göreceli olarak düzgün kurulmuş ve yazım kuralları düzgün uygulanmış metinlere ulaşmak amacıyla genel ağda yayınlanan gazete metinleri kullanılmıştır. Burada metinleri elde etmek üzere gazeteler yerine vikipedi gibi genel ağda yayınlanan bir ansiklopedi de kullanılabilirdi.

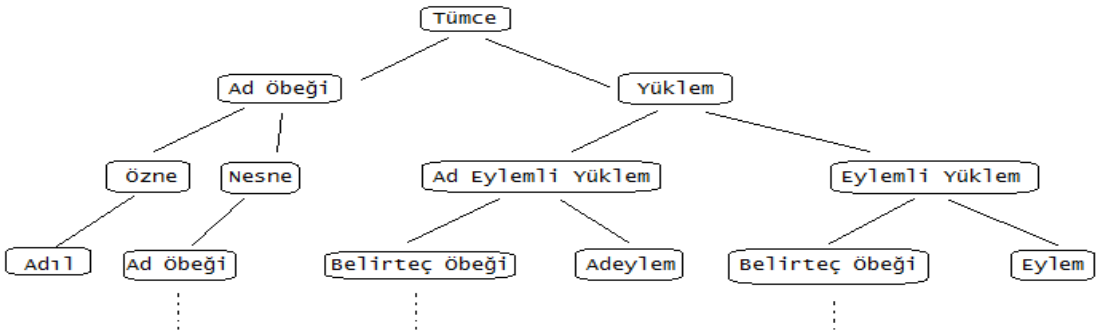
Metinlerin elde edilmesi yeterli değildir. Üzerinde çalışılacak sözcüklerin o anda tümcede hangi öğe olduğunu elde edilen ilişkide bir özellik olarak yer alması tasarlanmıştır. Bir sözcük öbeği eğer bir tümce ise bir anlamı var demektir. Öğelerine ayırma işlemi aslında tümceyi anlamak için zihnimizde çıkarttığımız çözümleme ağacıdır. Bir diğer deyişle eğer bir tümce öğelerine ayrılabiliyorsa o zaman anlaşılabilir demektir. Bu amaçla ifadenin tümce olduğuna karar verirken öğelerine ayrılabilen ifadeler seçilmeye çalışılmıştır. Çalışmada, tümcelerin sözdizimsel çözümlemesi diğer pek çok dilde olduğu gibi Chomsky sıradüzenindeki bağlam bağımsız dilbilgisine göre gerçeklenmeye çalışılmıştır. Tümcenin öğelerine ayrılması işlemi tek başına ele alınması gereken bir konu olduğu için sözdizimsel çözümlemesi görece olarak daha basit tümce türlerinin çözülmesi hedeflenmiştir. Seçilen tümce türleri aşağıda belirtildiği gibidir.

- Basit tümceler
- Girişik(Geçişmiş) bileşik tümceler
- Bağlı bileşik tümceler

6.1.1. Tümcenin çözümlenmesi

Tümcenin belirlenen şablonlara uygunluğu sınanmak üzere hazırlanan yordam prolog dili belirli tümce bilgisi sözdiziminden faydalanılarak hazırlanmıştır. Türk dili kuralları işletilmek üzere tümce bir ad öbeği ve bir yüklem olarak ikiye

ayrılmıştır.Ad öbeği sadece özne sadece nesne yada her ikisinden de oluşmaktadır.Her bir öbeğin atomik parça olan sözcüğe kadar parçalanabilir olması asıl hedeftir.Şekil 6.1 basit tümceyi çözümlerken kullanılan mantığın bir bölümünü göstermektedir.



Şekil 6.1 - Basit tümce çözümleme ağacı

Yordam çalıştırılmadan önce ulaşılması hedeflenen atomik parçalar yordama bildirilir. Bu işlem ifade içerisinde yer alan sözcükleri o tümcede bulundukları tiplerinin saptanması ve yordama bildirilmesi ile gerçekleştirilir. Yordamda kullanılmak üzere belirlenen tipler aşağıda yer almaktadırlar.

- Adlar ve Özeladlar olarak iki tipte
- Eylemler ve Adeylemler olarak iki tipte
- Belirteçler
- Adıllar
- Önadlar
- Bağlaçlar

Bu tiplerin yordama ilgili sözcük için tanıtılması ile yordamda ulaşılacak atomik parça tanımlanmış olur. Sözcükler ifadede doğru tipte ve doğru yerde yer alıyorlarsa yordam sonucu doğru olarak döner.

Tümce olduğu belirlenmiş şablonlara uygunluğu ile tespit edilmiş ifadede yükleme ulaşmak için Türk Dilinde yüklemin özelliklerinden faydalanılır.

- Elde edilen ifade basit tümce ise bir adet yüklemi olacaktır ve oda en sonda bulunan eylem yada adeylem sözcüktür.
- Eğer bağlaçlarla bağlanmış basit tümceler ise bağlaçlardan önce yer alan eylem yada adeylem o alt tümcenin yüklemi olacaktır.
- Eğer geçişmiş bir tümce ise ardı ardına gelen iki eylem yada adeylemden ilki alt tümcenin ikinciside tümcenin yüklem olan sözcüğü olarak kabul edilmiştir

Burada elde edilecek tümcenin öğelerine ayrılabilir olması yanı sıra yüklemine doğrudan ulaşılabilir olması hedeflenmiştir. Belirlenen şablonlar yüklemine daha kolay ulaşılabilmesi amacıyla ve göreceli olarak kolay çözümlenebildikleri için seçilmiştir.

Yüklemin elde edilmesi amacıyla çözümlenebilir olan tümcede bulunan adeylem ve eylem sözcükler değerlendirilmiştir. Bir tümcede yüklemin hangi sözcük olacağını belirlemede sözcüğün durumuna dair bölüm 4'de belirlenen kurallar uygulanmıştır.

6.1.2. Sıklıkların kaydedilmesi

Yapılan çalışmanın amacı sözcükleri tümce içinde seçilen diğer sözcük ile bulunuş özelliklerini kayıt altına almak , kayıt altına alınırken yüklem olarak saptanan sözcüğünde bu kayda eklenmesi ve elde edilen bu sıklıklarından çıkarımlar yapmak demiştik. Kaydedilen bu özellikler kayıt isimleri ve açıklamaları ile Çizelge 6.1 de yer almaktadır. Belirlediğimiz şablonlara göre çözümlenebilir olduğu saptanan tümcenin yüklemi dışında kalan her sözcük çiftinin aşağıdaki özellikleri yeni bir kayıt olarak eklenir.

Çizelge 6.1 - Sıklık Kayıtları

SOZCUK 1	Sözcük 1 in tanımlayıcısı
SOZCUK 2	Sözcük 2 nin tanımlayıcısı
SOZCUK1_ILK	Sözcük 1 tümcede ilk sözcük mü
SOZCUK2_SON	Sözcük 2 tümcede yüklemden bir önceki sözcük mü
ILK_IKI	Sözcük çifti tümcede ilk iki sözcük mü
SON_IKI	Sözcük çifti yüklem dışında tümcede son iki sözcük mü
ORT_SOZCUK_SAYISI	İki sözcük arasında kalan diğer sözcüklerin sayısı
ORT_AD_SAYISI	İki sözcük arasında kalan ad tipindeki sözcüklerin sayısı
KATEGORI_ID	Sözcüğün elde edildiği metnin belirlenmiş kategorisi
SOZCUK1_EK_SAYISI	İlk sözcüğün ek sayısı
SOZCUK2_EK_SAYISI	İkinci sözcüğün ek sayısı
YUKLEM_ID	Tümcenin yüklemi

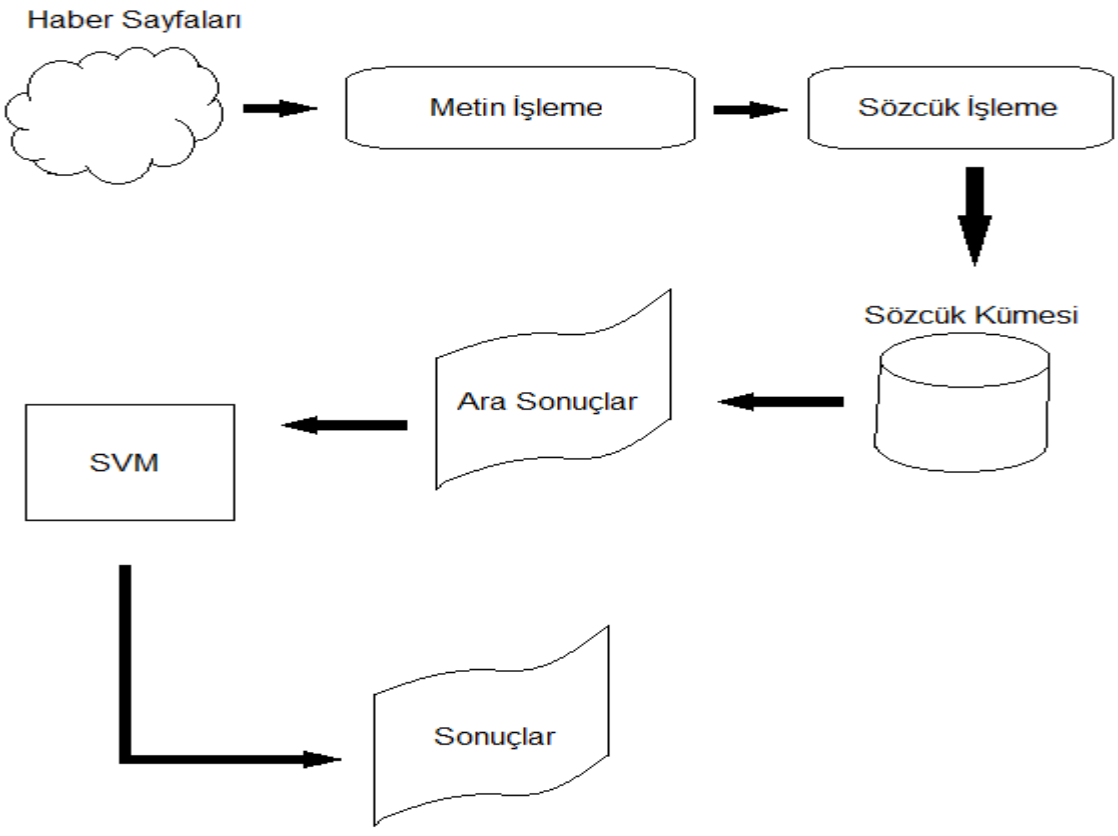
6.2. Mimari Yapı

Oluşturulan sistem farklı zamanlı çalışan 2 ayrı yordamdan meydana gelmektedir. İlk sistem Genel ağda bulunan gazetelerden metinleri toplayan ve basit bir ön işlem sürecinden geçirerek tümcelerin elde edilmesini sağlayan sistemdir. Gazetelerin genel ağ altında tanımladıkları 5 sınıflandırmadan faydalanarak ilk

sayfalarından haber detayına erişilmiştir. Zamanlanmış görev olarak çalışan yordam için akış şeması aşağıda Şekil 6.2 ile belirtilmiştir.

Oluşturulan yapıda gazetelerin detay haber sayfaları farklı farklı şekillerde olduğu için xml tagı olarak değerlendirildiğinde metin kısmı belirlenen eşik değerden yüksek olan tagların değerleri arasında en çok sözcük içereni haber detayı olarak değerlendirilmiştir.

Elde edilen metinler öncelikle metinin bütününü içeren metinler kümesine eklenmiştir. Metinlerden sözleri elde etmede öncelikle boşluklara göre elde edilen sözcükler zemberek kütüphanesi ile denetlenmiştir. Bu işlemin amacı genel ağ site isimleri ve ya para gibi nokta içeren sözcüklerin, sözleri elde etmede noktanın kullanılamaz hale getirmesinden ötürüdür.



Şekil 6.2 - Önerilen sistemin akış şeması

Bu işlem ile bu tip sözcükler saptanır ve boş bir değerle yer değiştirilir ve sonrasında nokta baz alınarak sözcüklere ulaşılır. Elde edilen her söz hangi metine ait olduğuna dair bilgiyle beraber sözler kümesine eklenir.

Burada elde edilen sözler boşluklara göre tekrar parçalanarak sözcükler elde edilir. Elde edilmiş bu sözcüklerden zemberek kütüphanesi yardımıyla kökleri elde edilir ve bu kökler Türk Dil Kurumunun genel ağ sayfalarından sorgulama sayfası kullanılarak ad, önad oluşları araştırılır. Bu aşamada ad olduğu saptanan sözcük veri yapımızda varlık sözcükler ve ekler kümelerini beslemektedir.

Yapılan ön çalışmadan sonra bölüm 6.1 de belirlediğimiz yapıya göre veri kümemizi besleyebilir duruma gelmiş oluruz . Varlık olabilecek sözcükler kök ve ek birimleri olarak söz içerisinde ikililer olarak sıklık tablomuza kaydedilir.

Farklı zamanda çalışan diğer sistemimiz ise bu frekans tablosunun bir görüntüsünü eğitim amaçlı kullanır. Elde ettiğimiz farklı görüntüler istatistiksel amaçlı olarak kullanılabilir. Sıklık kümemizin bir bölümü destek vektör makinesinin eğitiminde kalanını da sınama için kullanılır. Sıklık tablosunda eğitim ve sınama verilerinin ayrı tutulmasında yardımcı olması için tanımlayıcı yerleştirilmiştir. Sürekli gelişen sıklık kümemizin oluşturulmasında ilk kayıtların girişi için belirlenen kök ekler ikililerinin ilişkili oldukları varsayılmıştır. Sıklık tablomuzdan elde edilen verinin şekli Çizelge 6.2 de belirtilmiştir.

Çizelge 6.2 - Oranlar görüntüsü

s1_oran	Sözcük 1 in
s1s2_oran	Sözcük 1 in Sözcük 2 ile karşılaşma oranı
s1s2y_oran	Sözcük 1 in Sözcük 2 ile aynı yüklemde karşılaşma oranı
s1s2cesitsayisi	Sözcük 1 in kaç farklı sözcükle karşılaştığı
s1s2ycesitsayisi	Sözcük 1 in Sözcük 2 ile kaç farklı yuklem ile karşılaştığı
s1ilk	Sözcük 1 in ilk olma oranı
s2son	Sözcük 2 nin son olma oranı
İlkiki	Sözcük çiftinin ilk iki sözcük olma ortalaması
soniki	Sözcük çiftinin yüklem dışında son iki sözcük olma ortalaması
aras_ort	Sözcük çiftinin aralarında bulunan adların ortalaması
araa_ort	Sözcük çiftinin aralarında bulunan sözcüklerin ortalaması

6.3. Gerçekleştirim ve Uygulama Sonuçları

Bağlam görünümü Şekil 6.2 ile belirtilen sistemin incelenmesini kolaylaştırmak için genel veri madenciliği aşamaları ile ayrıştırılmıştır.

6.3.1. Veri kümesi oluşturma

Sistemde veri insanlar tarafından hali hazırda sınıflandırılmış metinlerden yararlanarak elde edilir. Gazete metinleri hali hazırda sınıflandırılmıştır. İlk program parçası java genel kütüphaneleri kullanarak genel ağda bulunan sınıflandırılmış haber sayfalarını elde eder. Haber sayfalarında genel yapı

incelendiğinde önce sınıflandırılmış haberlerin başlıklarının bulunduğu bir sayfa haber detayına bağlanmayı sağladığı görülür. Haber sayfalarında giriş sayfası, bu sınıflandırılmış haber başlıklarına erişimi sağlayan ilk ara yüzdür. Sınıflandırılmış haber sayfasına erişimde iki yöntem kullanılabilir.

1. Giriş sayfası HTML i taranarak düzenli ifadeler yardımıyla
    - sınıflandırma.habersayfası.com
    - www.habersayfası.com/sınıflandırma
2. Ele alınacak gazetelerin sınıflandırılmış sayfalarının önceden tanımlanması

ilk yöntem; giriş sayfası, sınıflandırılmış giriş sayfası, haber detayı notasyonuna uygun her gazete için kullanılabilir.

Ulaşılan haber detayı sayfasında HTML tagları arasında kalan sözcük sayısı 10 sözcükten fazla olduğu durumlarda bu tagların detay haberi taşıdığı varsayılarak metin soyutlanır.

6.3.2. Veri ayıklama ve veri önişleme

Elde edilen metinler henüz hala temiz değildir. Tekrarlamalar içerebilmektedir. Aynı metinin birden fazla kaynakta aynı şekilde bulunmaları olasıdır. İlk aşamada hedefimiz olan sözlere ulaşmada nokta kullanılabilmesi için genel ağ sayfa isimleri, para ifadeleri ve kısaltmalar gibi nokta içeren sözcükler etiketleri ile yer değiştirilir. Bu işlem sonrasında ifadenin nokta ile ayrılmış tümcelerden oluştuğu varsayılarak elde edilen bu ifade tümcelere ayrılması amacıyla nokta işaretine göre bölünür. Elde edilen alt ifadelerin hedef şablonumuza uyan bir tümce olup olmadığı parçalama ağacı oluşturularak kontrol edilir. Bu aşamada oluşturulan çözümleme ağacı eldeki sözcük öbeğinin

- Basit tümce
- Bileşik tümce
- Bağlaçlı tümce

örüntülerinden birisine uygun olup olmadığını kontrol eder ve uygun olduğu durumlarda veri tabanına kaydedilir. Belirlenen şablonlara uygunluğun sınanması ve çözümleme ağacının oluşturulmasında prolog dili belirli tümce bilgisi sözdiziminden faydalanılarak kullanılır. Yordamın yazar tarafından hazırlanmış olan prolog kodları Ek2 de bulunmaktadır. Yordam ifadeyi Chomsky nin kuramına uygun olarak , sözcük en küçük birimine ulaşana kadar parçalar. Çalışma zamanında yordam çalıştırılmadan önce ifadede bulunan sözcüklerin tipleri yordama belirtilir. Bu sayede Türk diline kurallarına uygun olarak tümcede ki görevi elde edilmeye çalışılır. Yordamın sonucu , tümce belirlenmiş olan yapısal tümce türlerinden birisine uygun ise ifade tümce olarak kabul edilir ve daha önceden kaydedilmemiş ise veri kümesine eklenir.

6.3.3. Veri azaltma ve veri dönüşümü

Elde edilen tümceler sözcüklere, sözcükler de kök ve eklerine ayrıştırılır ve sıklık veri kümesinde kullanılmak üzere önceden belirtilmiş özellikleri belirlenen özellikleri çıkartılır.

Sözcükler elde edilirken Zemberek kütüphanesinden yardım alınarak kök ve ek haline getirilirler. Her bir sözcük için sırasıyla

- elde edilen kök ad veya eylem soylu bir kök değilse kayıt dışı kalır
- elde edilmiş sözcüklerin belirteç oluşu ,eylem , adeylem önünde bulunması yada ad önünde bulunmasına göre ayrıştırılarak kaydedilir

Sözcükler, ekler ve kök ek karşılaşılma sıklığı yukarıdaki işlemler sonucunda beslenen veri kümeleridir.

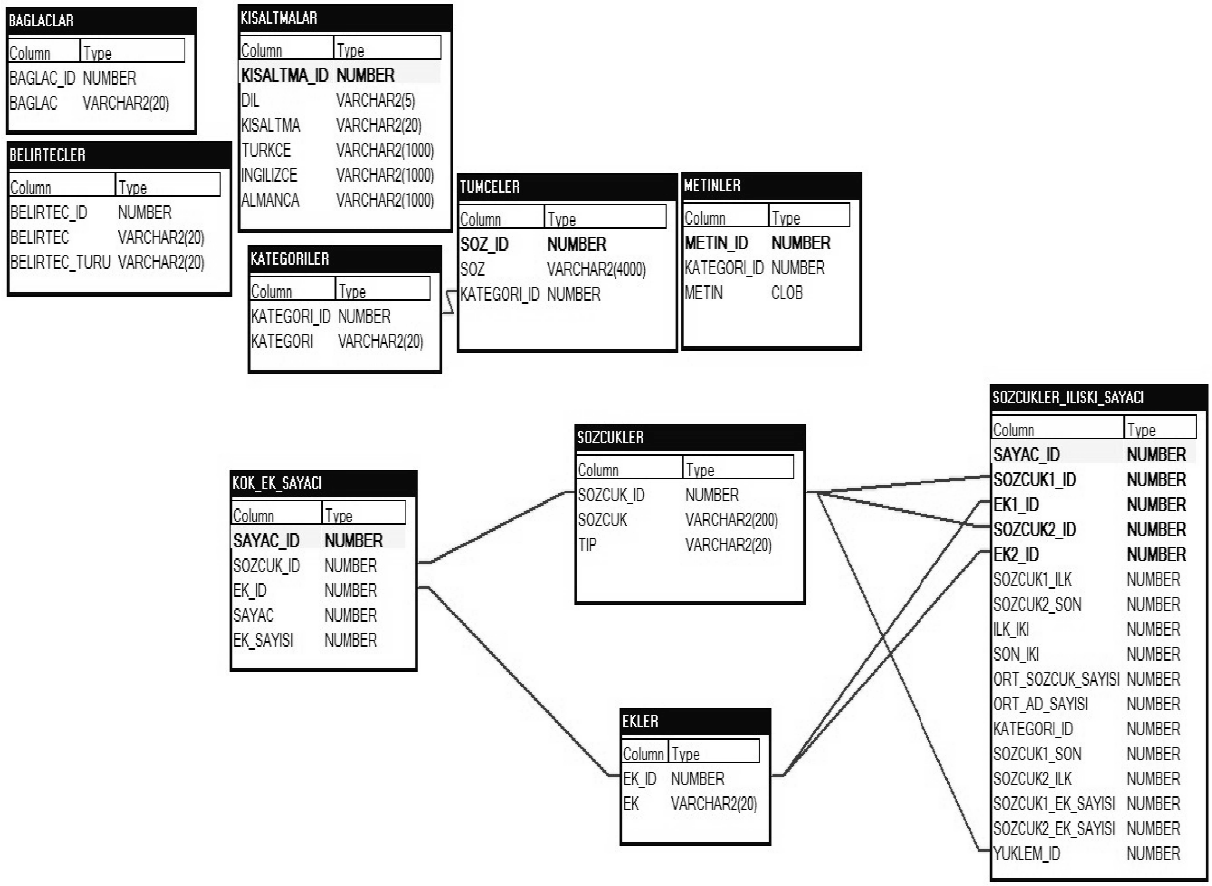
Sıklık kümesin her iki sözcük ve o anki yüklem olarak belirlenen sözcük bir anahtar olarak kullanılarak bir görüntü oluşturmak üzere bütün hareketin kaydedildiği bir kümedir. Sıklık kümesini beslemek için hedefimiz olduğu kesinleşen sözcüklerin aynı söz içerisinde bulunma durumları incelenir. Her varlık sayılan sözcük çifti için ekleri gözetilerek

- çiftin söz içerisinde ilk iki sözcük olmaları

- çiftin söz içerisinde son iki sözcük olmaları
- çiftin arasında başka sözcük bulunmama durumu
- çiftin arasında başka hedef sözcük bulunmama durumu
- çiftin arasında bulunan sözcük sayısı
- çiftin arasında bulunan hedef sözcük sayısı
- bulunduğu metnin sınıfı

değerleri bulunur ve kaydedilir. Yazar tarafından bu işlemi yapmak üzere hazırlanmış java yordamı Ek 1 de yer almaktadır.

Yapılan bu işlemlerin sonuçlarını kaydettiğimiz veri tabanının yapısı aşağıdaki Şekil 6.3 ile gösterilmiştir.



Şekil 6.3 - Önerilen sistemin veri yapısı

Oluşan veri kümesi işlem yapılmadan 2 küme ile bize istatistiksel sonuçlar vermektedir. Bu sonuçlar işlenerek farklı görüntüler oluşturulabilir.

- Kökün bir ek ile beraber kullanılma sıklığı
- Varlık olduğu varsayılan iki sözcüğün aynı kökler ile aynı söz içerisinde bulunmaları ve bulundukları andaki özellikleri

Bu iki küme den farklı görüntüler elde edilerek destek vektör makinesinde kullanılmak üzere çıktılar alınabilir. Bizim oluşturduğumuz görüntü Çizelge 6.3 de gösterildiği gibidir.

Çizelge 6.3 - Oranlar görüntüsü

s1_oran	Sözcük 1 in
s1s2_oran	Sözcük 1 in Sözcük 2 ile karşılaşma oranı
s1s2y_oran	Sözcük 1 in Sözcük 2 ile aynı yüklemde karşılaşma oranı
s1s2cesitsayisi	Sözcük 1 in kaç farklı sözcükle karşılaştığı
s1s2ycesitsayisi	Sözcük 1 in Sözcük 2 ile kaç farklı yüklem ile karşılaştığı
s1ilk	Sözcük 1 in ilk olma oranı
s2son	Sözcük 2 nin son olma oranı
İlkiki	Sözcük çiftinin ilk iki sözcük olma ortalaması
soniki	Sözcük çiftinin yüklem dışında son iki sözcük olma ortalaması
aras_ort	Sözcük çiftinin aralarında bulunan adların ortalaması
araa_ort	Sözcük çiftinin aralarında bulunan sözcüklerin ortalaması

Çizelge 6.3 de belirtildiği gibi şekillendirilen bilgi kullanılarak bu iki sözcüğün ilişki olduğu varsayımı yapılmaya çalışılmış ve varsayımın doğruluğu için destek vektör makinesi algoritmasından faydalanılmıştır. Destek vektör makinesi algoritması ile yapılan sınamada duyarlılık[7], anımsama[8] değerleri ve f-ölçüsü [9] değeri Çizelge 6.4 yardımıyla hesaplanmıştır.

Çizelge 6.4 - Karmaşıklık Matrisi

	Gerçek Değer	
Tahmin Edilen değer	İlişkililer-Doğru	İlişkili-Yanlış
	İlişkisiz-Yanlış	İlişkisiz-Doğru

$$\text{Kesinlik} = \frac{(Doğru\ İlişkililer)}{(Doğru\ İlişkililer) + (Yanlış\ İlişkililer)}$$

$$\text{Hatırlama} = \frac{(Doğru\ İlişkililer)}{(Doğru\ İlişkililer) + (Yanlış\ İlişkisizler)}$$

$$\text{F} - \text{ölçüsü} = \frac{(1 + \beta^2) * \text{Kesinlik} * \text{Hatırlama}}{\beta^2 * (\text{Kesinlik} + \text{Hatırlama}}$$

[7] Kesinlik (Precision) : Bir kümenin belirtilen sınıftaki obje oranıdır. i kümesinin j sınıfına göre kesinlik değeri precision(i,j)=pij'dir.

[8] Hatırlama (Recall) : Bir kümenin verilen sınıfın tüm objelerini içerip içermediğini ölçer. i kümesinin j sınıfına göre recall değeri: recall(i,j)= mij/ mj. mj j sınıfındaki obje sayısıdır.

[9] F-ölçüsü (F-measure) : Kesinlik ve recall birleşmesidir. Bir küme tek bir sınıfın mı objelerini içeriyor ve sınıfın tüm objelerini içeriyor mu, bunun cevabını verir.

F-ölçütü hesaplanırken β değeri 1 alınmıştır.

Bu değerlerin yanı sıra AİK[10](Alıcı İşletim Karakteristiği) alanı da sınama sonuçlarında karşılaştırma amacıyla yer almaktadır.

Oluşturulan veri kümesinden bilgi edinmek amacıyla çeşitli yorumlar yapılabilinir. Bunlardan bazılar

Eğer iki sözcük de varlık olsaydı ve ilişkili olsaydı;

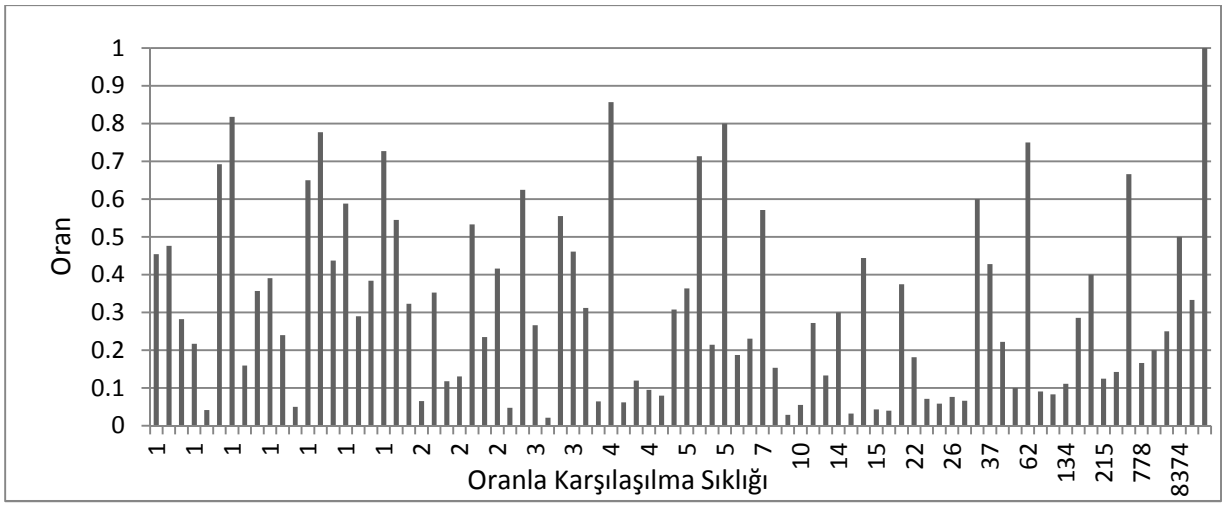
- Birlikte bulundukları durumlarda çok çeşitli yüklemlerle karşılaşabilirlerdi
- Birlikte bulunma sıklıkları yalnız karşılaşılma sıklığından fazla olurdu

Veri kümesinden yapılacak çıkarımlar çoğaltılabilir. Motivasyonumuz ilişkiyi saptamada tümcenin yüklemi gözetilerek karşılaşılma sıklıklarından faydalanmak olduğu için yukarıda belirlenen iki çıkarım incelenmiştir.

Sınamaların tamamında destek vektör makinesi aynı parametreler ile çalıştırılmıştır. Kullanılan görüntüdeki toplam 39379 kayıt bulunmaktadır ve bu kayıtların tamamı çalıştırma verisi olarak kullanılmıştır. Bunun sebebi sınıflandırma sonucunda verinin sınıflara uygun dağılmayışıdır , bir diğer değişle sınıflandırma yapıldıktan sonra herbir sınıftaki kayıt sayısının %0,0025 ve daha az olabilmesi ve sınama ve eğitim verisi olarak bölümlemelerde sınıflardan birinin olamama ihtimalidir. Bu kayıtlar ilişkili yada ilişkisiz olarak varsayılmasında yukarıda belirttiğimiz çıkarımlar kullanılmıştır ve seçilen eşik değere göre sınıflandırılmışlardır. Her bir sınama sonucunda sınıf1 ilişkili sınıf0 ilişkisiz olarak öngörülmüş kaydı etiketlemede kullanılmıştır.

İlk çıkarım hesaplanmak üzere oluşturulan görüntümüzdeki s1s2y_oran alanı kullanılır. Şekil 6.4 deki grafik, elde edilen iki sözcüğün farklı yüklemler ile karşılaşma değerleri ile oluşturulmuştur.

[10] Alıcı İşletim Karakteristiği (Receiver Operating Characteric): kesinlik ve hasiyet arasındaki dengeyi değerlendirmek için kullanılmıştır. Bir başka deyişle doğru artıların, yanlış artılara olan kesri olarak da ifade edilebilir



Şekil 6.4 - Sözcük çiftinin farklı yüklem ile karşılaşma oranları

s1s2y_oran değerleri incelendiğinde

Ortalama : 0,551

Standart Sapma : 0,314

değerleri hesaplanmıştır.

Bu alanın ilişkili olmada bir önemi var ise eşik değerinin ne olduğunu bulmak için rastgele 2 değer belirlenmesi gerekmektedir. Bu başlangıç değerleri için standart sapma ve ikinci değer olarak da ortalama ile standart sapmanın farkı kullanılmıştır. Bir diğer deyişle ilk sınamada s1s2y_oran değeri 0,314 den küçük olan kayıtlar ilişkili varsayılmış ikinci sınamada da s1s2y_oran değeri 0,237 den küçük olan kayıtların ilişkili olduğu varsayılmıştır. Sözcük çiftinin ilişkili olması durumu sınıf1 olarak ilişkisiz oluşu sınıf0 olarak etiketlenmiştir.

Herbir çizelgede karşılaştırma yapılmak üzere Doğru Artılar Oranı, Yanlış Artılar Oranı, Duyarlılık, Anımsama, F-ölçütü, AİK Alanı, destek vektör makinesine göre a ve b sınıfları bizim ayırdığımız sınıf1 ve sınıf0 a göre değerleri gösterilmiştir.Örnek bir şablon aşağıda yer almaktadır.Çizelgede cm değişkenleri ile ifade edilen alt çizelge aynı zamanda karmaşıklık matrisidir.

Çizelge 6.5 - Sınama değerlendirme sonuç şablonu

Seçilen alan ve koşulu								
	Doğru Artıların Oranı	Yanlış Artıların Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	Destek Vektör Makinesine göre sınıfa	Destek Vektör Makinesine göre sınıfb
a = sınıf0	x	x	x	x	x	roc	cm	cm
b = sınıf1	x	x	x	x	x	roc	cm	cm
Ağ.Ort.	x	x	x	x	x	roc		

Yapılan ilk sınama sonuçları Çizelge 6.6 ile belirtilmiştir.

Çizelge 6.6 - s1s2y_oran 0,314 , 0,237 sınama sonuçları

s1s2y_oran < 0,314								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	0,994	0,114	0,97	0,994	0,982	0,94	30746	187
b = sınıf1	0,886	0,006	0,976	0,886	0,929	0,94	964	7482
Ağ.Ort.	0,971	0,091	0,971	0,971	0,97	0,94		
s1s2y_oran < 0,237								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	0,995	0,143	0,986	0,995	0,99	0,926	35645	194
b = sınıf1	0,857	0,005	0,94	0,857	0,896	0,926	507	3033
Ağ.Ort.	0,982	0,131	0,982	0,982	0,982	0,926		

Bu sonuçların değerlendirmesi AİK alanının azaldığı gözlemlenmiştir. Bir sonraki sınamada s1s2y_oran değeri rastgele 0,118 seçilerek yapılmıştır. Sınama sonuçları Çizelge 6.7 da belirtilmiştir.

Çizelge 6.7 - s1s2y_oran 0,118 sınama sonucu

s1s2y_oran < 0,118								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	0,999	0,336	0,995	0,999	0,997	0,831	38809	29
b = sınıf1	0,664	0,001	0,925	0,664	0,773	0,831	182	359
Ağ.Ort.	0,995	0,332	0,994	0,995	0,994	0,831		

Bir sonraki sınama işleminde önceden alınan AİK değerlerini yükseltmek amacıyla s1s2y_oran değeri ilk belirlenen değerin üzerinde olan 0,551 belirlenmiştir.Yapılan sınama sonuçları Çizelge 6.8 de belirtilmiştir.

Çizelge 6.8 - s1s2y_oran 0,551 sınama sonucu

s1s2y_oran < 0,551								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	0,961	0	1	0,961	0,98	0,981	12028	486
b = sınıf1	1	0,039	0,982	1	0,991	0,981	1	26864
Ağ.Ort.	0,988	0,027	0,988	0,988	0,988	0,981		

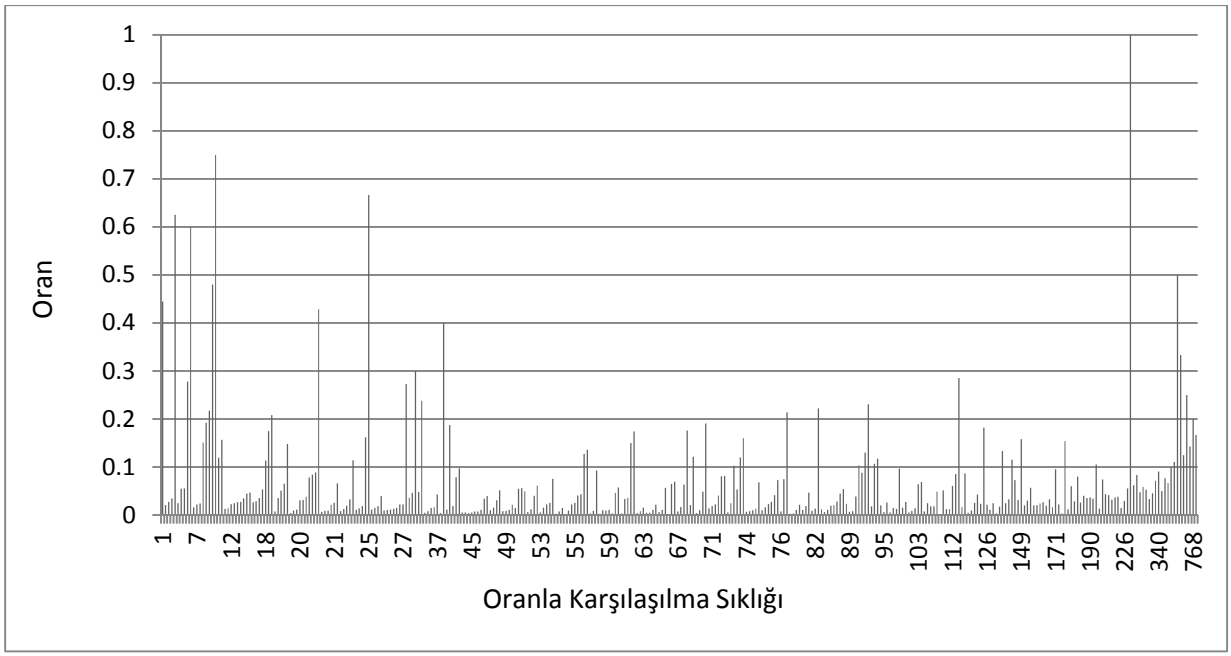
Sözcük çiftinin yüklem ile oranına dair yapılan sınamaların sonucunda belirlediğimiz iki sözcük çok çeşitli yüklemler ile karşılaşıyorsa ilişkilidir çıkarımının örneklememize göre doğru olmadığı görülür.

İkinci çıkarımımızı incelemek amacıyla aynı yöntem kullanılmıştır. Görüntümüzde hesapladığımız s1s2_oran değeri birlikte bulunma sıklığı ile yalnız bulunma sıklığını oranlayarak oluşturduğumuz alandır. Bu oranın 1 e yaklaşması iki sözcük arasında ilişki bulunma olasılığını arttırdığı varsayılmıştır. Veri kümemizden elde ettiğimiz s1s2_oran değerleri aşağıda Şekil 6.5 ile gösterilmektedir.

s1s2_oran değerleri incelendiğinde

Ortalama :0.081
Standart Sapma : 0.12

hesaplanmıştır.



Şekil 6.5 - Sözcüğün farklı sözcükler ile karşılaşma oranı

Sınama yapılmak üzere rastgele 0,081 , 0,201 , 0,321 , 0,642 değerleri s1s2_oran alanı için eşik değer olarak seçilmiş ve bu eşik değerlere göre yapılan sınıflandırma destek vektör makinesi algoritmasıyla sınanmıştır. Sınama sonuçları Çizelge6.9 de belirtilmektedir.

Alınan sonuçlara AİK Alanı sürekli artma yada sürekli azalma eğiliminde olmadığından sonraki sınamalarda s1s2_oran eşik değeri aralık olarak verilmiş ve rastgele seçilen 0,321 - 0,642 aralığı ve 0,321 - 0,522 aralığı ilişkili olarak belirlenerek sınama tekrarlanmıştır. Yapılan sınamanın sonuçları Çizelge 6.10 ile belirtilmiştir.

Çizelge 6.9 - s1s2_oran 0,081 , 0,201 , 0,321 , 0,642 sınama sonuçları

s1s2_oran > 0,081								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf1	0,892	0,036	0,91	0,892	0,901	0,928	10263	1246
b = sınıf0	0,964	0,108	0,956	0,964	0,96	0,928	1009	26861
Ağ.Ort.	0,943	0,087	0,942	0,943	0,943	0,928		
s1s2_oran > 0,201								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	0,996	0,184	0,985	0,996	0,991	0,906	36205	134
b = sınıf1	0,816	0,004	0,949	0,816	0,878	0,906	558	2482
Ağ.Ort.	0,982	0,17	0,982	0,982	0,982	0,906		
s1s2_oran > 0,321								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	0,998	0,096	0,996	0,998	0,997	0,951	37611	72
b = sınıf1	0,904	0,002	0,955	0,904	0,929	0,951	163	1533
Ağ.Ort.	0,994	0,092	0,994	0,994	0,994	0,951		
s1s2_oran > 0,642								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	1	0,136	0,999	1	0,999	0,932	39056	0
b = sınıf1	0,864	0	1	0,864	0,927	0,932	44	279
Ağ.Ort.	0,999	0,135	0,999	0,999	0,999	0,932		

Çizelge 6.10 - s1s2_oran 0,321 - 0,642 ve 0,321 - 0,522 aralıkları sınama sonuçları

s1s2_oran > 0,321 & s1s2_oran < 0,642								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf1	0,997	0,114	0,996	0,997	0,996	0,941	37891	115
b = sınıf0	0,886	0,003	0,914	0,886	0,899	0,941	157	1216
Ağ.Ort.	0,993	0,11	0,993	0,993	0,993	0,941		
s1s2_oran > 0,321 & s1s2_oran < 0,522								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	0,997	0,115	0,996	0,997	0,996	0,941	37895	122
b = sınıf1	0,885	0,003	0,908	0,885	0,896	0,941	157	1205
Ağ.Ort.	0,993	0,111	0,993	0,993	0,993	0,941		

Aralığın daraltılmasıyla sınıf1 için doğru artıların oranı azaldığı gözlemlenmiştir ve aralığın üst değeri tekrar 1 alınarak sırayla 0,402 , 0,26 , 0,483 , 0,564 , 0,523 , 0,499 değerleri ile sınıflandırma yapılarak sınamalar yapılmıştır. Yapılan sınamaların sonuçları Çizelge 6.11 ile gösterilmiştir.

Çizelge 6.11 - s1s2_oran 0,402 , 0,26 , 0,483 , 0,564 , 0,523 sınama sonuçları

s1s2_oran > 0,402								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf1	1	0,075	0,998	1	0,999	0,962	38376	19
b = sınıf0	0,925	0	0,98	0,925	0,951	0,962	74	910
Ağ.Ort.	0,998	0,073	0,998	0,998	0,998	0,962		
s1s2_oran > 0,26								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	0,998	0,181	0,991	0,998	0,994	0,909	37382	67
b = sınıf1	0,819	0,002	0,959	0,819	0,884	0,909	349	1581
Ağ.Ort.	0,989	0,172	0,989	0,989	0,989	0,909		
s1s2_oran > 0,483								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	1	0,059	0,999	1	0,999	0,971	38421	18
b = sınıf1	0,941	0	0,98	0,941	0,96	0,971	55	885
Ağ.Ort.	0,998	0,057	0,998	0,998	0,998	0,971		
s1s2_oran > 0,564								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	1	0,147	0,999	1	0,999	0,927	39045	0
b = sınıf1	0,853	0	1	0,853	0,921	0,927	49	285
Ağ.Ort.	0,999	0,145	0,999	0,999	0,999	0,927		
s1s2_oran > 0,583								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	1	0,147	0,999	1	0,999	0,927	39045	0
b = sınıf1	0,853	0	1	0,853	0,921	0,927	49	285
Ağ.Ort.	0,999	0,145	0,999	0,999	0,999	0,927		

Bu iki oran üzerinde yapılan sınamalar sonucu olarak her ikisinin de en iyi değerleri kullanılarak sınıflandırma tekrar düzenlenmiştir. Belirlenen en iyi değerler s1s2y_oran için 0,551'den küçük olmak s1s2_oran için 0,483'den büyük olmak. Bu değerlerle sınıflandırılan kümemiz tekrar destek vektör makinesi algoritmasıyla sınanmıştır ve sınama sonuçları Çizelge 6.12 ile belirtilmiştir.

Çizelge 6.12 - s1s2_oran > 0,483 ve s1s2y_oran < 0,551 sınama sonucu

s1s2_oran > 0,483 & s1s2y_oran < 0,551								
	DA Oranı	YA Oranı	Duyarlılık	Anımsama	F-Ölçütü	AİK Alanı	a	b
a = sınıf0	1	0.116	0.999	1	0.999	0.942	38876	2
b = sınıf1	0.884	0	0.996	0.884	0.937	0.942	58	443
Ağ.Ort.	0.998	0.114	0.998	0.998	0.998	0.942		

Uygulanan eşik değerlerine göre ilişkili kabul ettiğimiz kayıtların örnekleri aşağıda (Çizelge 6.13 'de) yer almaktadır.

Çizelge 6.13 - Eşik değerlere göre ilişkili varsayılan sözcük örnekleri

Sözcük 1	***Sözcük 2***	***Yüklem***
Atatürk	milliyet	ol
Atatürk	milliyet	tanımla
doküman	bedel	belirle
doküman	bedel	ol
düşünce	değişik	bekle
düşünce	değişik	ol
ekipman	yolla	ver
engel	pranga	tersin
engel	kurtul	bahset
gram	kavanoz	içer
gram	kavanoz	bulun
hamur	parça	dinlen
hamur	bölü	yuvarla
hamur	bölü	dinlen
hamur	parça	yuvarla

7. SONUÇ

Yapılan çalışma ile, ilişki tanımlamada kullanılmak üzere bir veri kümesi oluşturulmuş ve dünyada kullanılan en yaygın altıncı dil olan Türkçemizin anlamsal ağa ayak uydurması için takip edilebilecek bir şablon oluşturulmuştur. Oluşturulan şablon üzerinde çeşitli değişiklikler yapılarak sonuçların daha farklı açılardan anlamlı olması sağlanabilineceği bir ön çalışma yapılmıştır. Yapılan çalışmada açık kaynak kodlu zemberek kütüphanesi[11] gibi önceden yapılmış bazı çalışmaların ve uygulamaların sonuçları kullanılmıştır.

Yüklemin elde edilmesinde ve hedef belirlenen tümce türüne uygunluk için yapılan çalışma aynı zamanda otomatik içerik çıkarımının bir sonraki görevi olan eylem(olay) belirleme işlemi için bir ön çalışma niteliğindedir.

Veri kümesi oluşturmada kullanılan yordam aynı zamanda genel ağda yer alan gazete haber detayları yerine genel ağda yer alan ansiklopedi niteliğindeki sayfalar içinde kullanılarak farklı sonuçlar elde edilebilir. Kullanılan yordam ile 3795 bağlantı üzerinde işlem yapılarak 4479 haber detayı olabilecek metin ve bu metinlerden de 2054 tümce elde edilmiştir. Bu tümcelerde yer alan 4355 sözcük karşılaştırma için kullanılmış ve 44444 adet karşılaşma kaydedilmiştir.

Yapılan sınamalarda veri kümesinden elde edilen sözcük çiftleri farklı yüklem ile karşılaşma oranlarına göre ve sözcük çiftlerinin farklı sözcükler ile karşılaşma oranlarına göre karşılaştırılmıştır. Sınamalar sonucunda iki sözcüğün farklı yüklem ile karşılaşması oranı 0 a yakınlaştıkça elde edilen vektörlerin benzerliği artması beklenirken tam tersi sonuçlar alınmıştır. İkinci aşamada sözcüklerin farklı sözcükler ile karşılaşma oranının 1 e yaklaştıkça vektörlerin benzerliğinin artması beklenmiş ve arttığı gözlemlenmiştir. Son olarak belirlenen iki alanın birlikte kullanılarak elde edilen sınıflandırma sınanmıştır.

---

[11] Zemberek, açık kaynak kodlu Türkçe Doğal dil işleme kütüphanesi ve OpenOffice , Libre Office eklentisidir. İlk sürümü BSD lisansı ile dağıtılmıştır. Tamamen Java ile geliştirilen kütüphane, yazım denetimi, hatalı kelimeler için öneri, heceleme, deascifier, hatalı kodlama temizleme gibi işlevlere sahiptir(http://tr.wikipedia.org/wiki/Zemberek_(yaz%C4%B1l%C4%B1m)).

Anlamsal ilişkilendirme sorununa yaklaşımımız; ilişkilendirme yapılabilmesi için ifade tümce olarak ele alınmasıdır ve sözcüğün o tümcedeki görevininde hesaba katılmasıdır. Okuduğumuz yazıyı anlayabilmeyi ve sözcükleri ilişkilendirebilmeyi büyük ölçüde sözcüğün tipi ve tümcedeki görevi ile sağlamaktayız. Bir diğer deyişle bir tümceyi öğelerine ayırabiliyorsak o tümceyi anladığımızı söyleyebiliriz.

Yapılan bu çalışma sonrasında oluşturulan model genişletilerek belirlenen sözcüklerin tümce içindeki görevlerininde sıklık kümesine kaydetmek ileriye dönük çalışmalarda hedeflenmektedir. Modelde yapılacak değişiklikler ile farklı kaynaklardan beslenmesi ve veri kümesinin çeşitliliğinin arttırılması sağlanabilir.

KAYNAKLAR

[1] Adalı, Ş., ve A. C. Sönmez. "Eylem çıkarımı ve varlık tanıma için ontoloji tabanlı bilgi çıkarımı ve belge yapı analizinin tümleştirilmesi." İTÜ FBE Bilg.Müh.,YTÜ Bilg Müh., Haziran 2011.

[2] Bach, N., ve S. Badaskar. "A Survey On Relation Extraction." Language Technologies Institute Carnegie Mellon University, 2007.

[3] Bunescu, R. C., Mooney, R. J. "Subsequence kernels for relation extraction." Neural Information Processing Systems, NIPS Vancouver, British Columbia, 2005.

[4] Cucerzan, S., ve D. Yarowsky. "Language Independent Named Entity Recognition Combining Morphological and Contextual Evidence." SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 2009.

[5] Çabuk, H., Ç Yüksel, Z Mocan, B. Diri, ve Amasyalı M.F. "Metin Analizi Ve Sorgulama (MAvS)." Koç Üniversitesi İstanbul: 11. sinyal işleme ve iletişim uygulamaları, 2003.

[6] Dalkılıç, F.E, S. Gelisli, ve B. Diri. "Türkçe Kural Tabanlı Varlık İsmi Tanıma." 18. Sinyal İşleme ve Uygulama Kurultayı, Diyarbakır, 2010.

[7] Dolgun, M.Ö., TG Özdemir, ve D. Oğuz. "Veri madenciliği'nde yapısal olmayan verinin analizi:Metin ve web madenciliği." İstatistikçiler Dergisi, 2009: 48-58.

[8] Gruber, T.R. "A translation approach to portable ontologies." Knowledge Acquisition, 1993: 199-220.

[9] Harris, Z.S. "Linguistic transformations for information retrieval." International Conference on Scientic Information, 1958: 158.

[10] Maedche, A., ve S. Staab. "Ontology Learning for the Semantic Web." IEEE Intelligent Systems, 2001: 77-79.

[11] Mikheev, A., M. Moens, ve C. Grover. "Named Entity Recognition without Gazetteers." Proceedings of the 9th Conference of the European ACL (EACL 99), 1999.

[12] Miller, A. "WordNet: An On-line Lexical Resource. J. Lexicography." 1990: 106-127.

[13] Oflazer, K. "Two-level Description of Turkish Morphology." Literary and Linguistic Computing 9, 1995: 137–148.

[14] Özyurt, Ö, ve C. Köse. "Türkçe Tabanlı Diyalog Sistemi Tasarımı ve İnternet (Chat) Ortamlarından Bilgi Çıkarımı." Karadeniz Teknik Üniversitesi, 2006.

[15] Tatar, S. "Automating Iformation Extraction Task For Turkish Texts." Bilkent University , Computer Engineering Department, 2011.

[16] Tür, G., D. Hakkani-Tür, ve K. Oflazer. "A Statistical Information Extraction System for Turkish." Natural Language Engineering, Vol. 9, No. 2, 2003: 181-210.

[17] Zhao, S., Grishman, R. "Extracting relations with integrated information using kernel methods." 419–426. Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, 2005.

[18] Zhou, G., J. Su, Zhang J., ve Zhang M. "Exploring Various Knowledge in Relation Extraction." Institute for Infocomm research 21 Heng Mui Keng Terrace, Singapore, 2009.

EKLER

Geliştirilen uygulamanı bütün kaynak kodları , elde edilen veri kümesi ve sorgular https://sourceforge.net/p/thesisrepo/code-0/29/tree/ adresinde yer almaktadır. Bazı önemli yordamlar aşağıda yer almaktadır.

1. Ek1 Veri Sentezleme Yordamı

```
package name.alp.newsCollector.analyser;



import java.nio.charset.Charset;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Locale;

import java.util.StringTokenizer;



import name.alp.newsCollector.model.Sentence;

import name.alp.newsCollector.model.Word;

import name.alp.newsCollector.model.WordType;

import name.alp.newsCollector.thread.PLThread;

import name.alp.newsCollector.util.Validator;

import name.alp.tadba.cache.BaseCache;
```

```
import name.alp.tadba.constant.NON_INTEREST;

import name.alp.tadba.dao.FixDao;

import name.alp.tadba.dao.SentenceDao;

import name.alp.tadba.dao.WordDao;

import name.alp.tadba.dao.WordFixCounterDao;

import name.alp.tadba.dao.WordRelationDao;

import name.alp.tadba.model.Fix;

import name.alp.tadba.model.WordFixCounter;

import name.alp.tadba.model.WordRelation;

import net.zemberek.erisim.Zemberek;

import net.zemberek.islemler.cozumleme.CozumlemeSeviyesi;

import net.zemberek.tr.yapi.TurkiyeTurkcesi;

import net.zemberek.yapi.Kelime;

import net.zemberek.yapi.KelimeTipi;

import net.zemberek.yapi.ek.Ek;



import org.apache.log4j.Logger;



public class Analyser {

        static Logger logger = Logger.getLogger(Analyser.class);



        public static List<Sentence> parsePhrase(name.alp.tadba.model.Phrase phrase) {
```

```
                List<Sentence> sentences = new ArrayList<Sentence>();

                String dataa = phrase.getPhrase();

                dataa = dataa.replaceAll(" *, *", " ve ").replaceAll("\"", "").replaceAll("'",
"").replaceAll("\\?", "");

                Sentence sentenceBuf = new Sentence();

                StringTokenizer token = new StringTokenizer(dataa, " ");

                int wordIndex = 0;

                int sentenceIndex = 0;

                List<Word> adverbList = new LinkedList<Word>();


                while (token.hasMoreElements()) {

                        String sWord = token.nextToken();

                        Word[] words = getWords(sWord);

                        boolean sentenceEnd = words[1] != null;

                        Word firstWord = words[0];

                        Word secondWord = words[1];

                        wordIndex++;


                        if (firstWord.getWord().equals(".")) {// end of sentence list it and
create new

                                sentenceBuf.setSentenceIndex(sentenceIndex);


        sentenceBuf.setpClass(phrase.getCategory().getCategory());
```

```
                                sentences.add(sentenceBuf);

                                sentenceBuf = new Sentence();

                                firstWord = new Word(secondWord.getRoot(),
secondWord.getWord(), secondWord.getType(), secondWord.getRootType());

                                secondWord = null;

                                wordIndex = 0;

                                sentenceIndex++;

                        }


                        firstWord = describeWord(firstWord);

                        firstWord.setWordIndex(wordIndex);


                        if (firstWord.getRootType().equals(WordType.BELIRTEC)) {

                                adverbList.add(firstWord);

                        } else {// if (!firstWord.getType().equals(WordType.BELIRTEC)) {

                                if (adverbList.size() > 0) {

                                        // gelen eylem ise bunlar belirtec olarak kalir degilse
onad olur

                                        if (!(firstWord.getType().equals(WordType.EYLEM) ||
firstWord.getType().equals(WordType.ADEYLEM))) {

                                                for (Word adjective : adverbList) {

                                                        adjective.setType(WordType.ONAD);

                                                }
```

```
                                        }

                                        // liste eklenir

                                        for (Word adjectiveOrAdverb : adverbList) {

                                                sentenceBuf.append(adjectiveOrAdverb);

                                        }

                                        adverbList.clear();

                                }

                        }


                        sentenceBuf.append(firstWord);


                        if (sentenceEnd && secondWord != null) {

                                sentenceBuf.setSentenceIndex(sentenceIndex);

sentenceBuf.setpClass(phrase.getCategory().getCategory());

                                sentences.add(sentenceBuf);

                                sentenceBuf = new Sentence();

                                wordIndex = 0;

                                sentenceIndex++;

                                if (!secondWord.getWord().equals(".")) {

                                        secondWord = describeWord(secondWord);

                                        secondWord.setWordIndex(wordIndex);
```

```
                                    if
(secondWord.getRootType().equals(WordType.BELIRTEC)) {
                                        adverbList.add(secondWord);
                                    } else {// if
(!firstWord.getType().equals(WordType.BELIRTEC)) {
                                        if (adverbList.size() > 0) {
                                            // gelen eylem ise bunlar belirtec
olarak kalir degilse onad olur
                                            if
(!(secondWord.getType().equals(WordType.EYLEM) ||
secondWord.getType().equals(WordType.ADEYLEM))) {
                                                for (Word adjective :
adverbList) {

        adjective.setType(WordType.ONAD);
                                                }
                                            }
                                            // liste eklenir
                                            for (Word adjectiveOrAdverb :
adverbList) {

        sentenceBuf.append(adjectiveOrAdverb);
                                            }
                                            adverbList.clear();
```

```
                                        }

                                        sentenceBuf.append(secondWord);

                                    }

                                }


                            }


                        }

                        return sentences;

                    }


                    private static Word describeWord(Word word1) {

                        if (word1.getRootType() != null &&
word1.getRootType().equals(WordType.OZELAD)) {

                            return word1;

                        }

                        Word word = new Word(word1.getRoot(), word1.getWord(),
WordType.BILINMIYOR, WordType.BILINMIYOR);

                        word.setRootType(WordType.ILGIDISI);

                        if (BaseCache.hasConjunction(word1.getWord().toString())) {
```

```
                word.setRootType(WordType.BAGLAC);

            } else if (BaseCache.hasAdverb(word1.getWord().toString())) {

                word.setRootType(WordType.BELIRTEC);



            } else if (!BaseCache.isNonInterest(word.getWord()) &&
getZemberek().cozumleyici().cozumlenebilir(word.getWord())) {

                Kelime[] kelimeler =
getZemberek().cozumleyici().cozumle(word.getWord(),
CozumlemeSeviyesi.TUM_KOK_VE_EKLER);

                for (int i = 0; i < kelimeler.length; i++) {

                    Kelime kelime = kelimeler[i];

                    word.setRoot(kelime.kok().icerik());


        word.setRootType(WordType.convertType(kelime.kok().tip()));



                    // son iki ek onemli

                    HashMap<String, Ek> ekmap = new HashMap<String,
Ek>();

                    Ek[] ekler = kelime.ekDizisi();

                    for (int j = 1; j < ekler.length; j++) {

                        ekmap.put(ekler[j].ad(), ekler[j]);

                    }

                    if (kelime.ekDizisi().length == 1 && kelimeler.length > 1 &&
!kelime.kok().tip().equals(KelimeTipi.ISIM)) {
```

```
                                continue;

                            } else if (kelime.ekDizisi().length > 1) {

                                Ek ek2 = kelime.ekDizisi()[kelime.ekDizisi().length -
1];

                                if (ek2.ad().contains("FIIL") &&
ek2.ad().contains("ZAMAN")) {

                                    word.setType(WordType.EYLEM);

                                } else if (kelime.kok().tip().equals(KelimeTipi.FIIL)) {

                                    if (ek2.ad().contains("ISIM") ||
ek2.ad().contains("MASTAR") || ek2.ad().contains("FIIL_DONUSUM")) {

                                        word.setType(WordType.AD);

                                    } else if
(ekmap.containsKey("FIIL_OLUMSUZLUK_ME") && kelimeler.length > 1) {

                                        continue;

                                    }

                                } else if (kelime.kok().tip().equals(KelimeTipi.SAYI)
&& ek2.ad().contains("SAYI_SIRA_INCI")) {

                                    word.setType(WordType.ONAD);

                                }

                            }

                            for (Ek ek : ekmap.values().toArray(new Ek[0])) {

                                word.appendFix(new
name.alp.newsCollector.model.Fix(ek.ad().substring(ek.ad().lastIndexOf("_") +
1).toLowerCase(Locale.ENGLISH)));
```

```
                                }

                                break;

                        }

                }


                return word;

        }


        private static HashMap<String, WordFixCounter> saveWords(List<Word> words) {

                WordDao wordDao = new WordDao();

                FixDao fDao = new FixDao();

                WordFixCounterDao wfDao = new WordFixCounterDao();

                HashMap<String, WordFixCounter> result = new HashMap<String,
WordFixCounter>();

                for (Word word : words) {

                        name.alp.tadba.model.Word dbWord = new
name.alp.tadba.model.Word();

                        dbWord.setWord(word.getRoot());

                        WordType type = word.getType().equals(WordType.BILINMIYOR) ?
word.getRootType() : word.getType();

                        dbWord.setType(type.name());

                        dbWord = wordDao.saveNonExist(dbWord);

                        for (name.alp.newsCollector.model.Fix fix : word.getFixes()) {
```

```
                                Fix dbFix = new Fix();

                                dbFix.setFix(fix.getFix());

                                dbFix = fDao.saveNonExist(dbFix);

                                WordFixCounter counter = wfDao.findByKey(dbWord,
dbFix);

                                counter = (counter == null ? new WordFixCounter() :
counter);

                                counter.setWord(dbWord);

                                counter.setFix(dbFix);

                                counter.incrementCounter();

                                wfDao.saveOrUpdate(counter);

                                result.put(word.getWord(), counter);

                                word.setWordFix(counter);


                        }


                        //      fixBuffer.append(ek.ad().substring(ek.ad().lastIndexOf("_")     +
1).toLowerCase(Locale.ENGLISH));


                }

                return result;

        }
```

```
	public static void saveSentences(List<Sentence> sentences) {

		SentenceDao senDao = new SentenceDao();



		for (Sentence sentence : sentences) {

			if (!isPhraseParseble(sentence)) {

				continue;

			}



			name.alp.tadba.model.Sentence sen = new name.alp.tadba.model.Sentence();

			sen.setCategory(BaseCache.getCategory(sentence.getpClass()));

			sen.setSentence(sentence.getSentence());

			senDao.saveNonExist(sen);

			if (sen.getId() < 1) {

				continue;

			}



			HashMap<String, WordFixCounter> parses = saveWords(sentence.getWords());



			saveRelation(sentence, parses);

			// saveInverseRelation(sentence, parses);
```

```
            }


        }


        private static void saveRelation(Sentence sentence, HashMap<String,
WordFixCounter> parses) {
            WordRelationDao relationDao = new WordRelationDao();
            List<name.alp.tadba.model.Word> verbs = getVerb(sentence);
            for (name.alp.tadba.model.Word verb : verbs) {
                for (int i = 0; i < sentence.getWords().size() - 1; i++) {
                    int entitiesBetween = 0;
                    WordFixCounter wf1 =
parses.get(sentence.getWords().get(i).getWord());
                    if (wf1 == null) {
                        continue;
                    }
                    name.alp.tadba.model.Word word1 = wf1.getWord();
                    if (word1 == null ||
word1.getType().equals(WordType.ADEYLEM.name()) ||
word1.getType().equals(WordType.EYLEM.name())) {
                        continue;
                    }
```

```
                    Fix fix1 = wf1.getFix();

                    for (int j = i + 1; j < sentence.getWords().size(); j++) {


                        WordFixCounter wf2 =
parses.get(sentence.getWords().get(j).getWord());

                        if (wf2 == null) {

                            continue;

                        }

                        name.alp.tadba.model.Word word2 = wf2.getWord();

                        if (word2 == null ||
word2.getType().equals(WordType.ADEYLEM.name()) ||
word2.getType().equals(WordType.EYLEM.name())) {

                            continue;

                        }

                        Fix fix2 = wf1.getFix();


                        WordRelation relation = new WordRelation();

                        relation.setWord1(word1);

                        relation.setWord2(word2);

                        relation.setFix1(fix1);

                        relation.setFix2(fix2);

                        if
(BaseCache.getCategory(sentence.getpClass().toLowerCase(Locale.ENGLISH)) == null)
```

```
{
                                                    logger.error("null:"                                    +
sentence.getpClass().toLowerCase(Locale.ENGLISH));
                                                }


        relation.setCategory(BaseCache.getCategory(sentence.getpClass().toLowerCase(
Locale.ENGLISH)));
                                                relation.setFirstCouple((j == 1 && i == 0 ? 1 : 0));
                                                relation.setLastCouple((j    ==    i    +    1    &&    i    ==
sentence.getWords().size() - 2 ? 1 : 0));
                                                relation.setW1First((i == 0 ? 1 : 0));
                                                relation.setW1Last(0);
                                                relation.setW2First(0);
                                                relation.setW2Last((j == sentence.getWords().size() -
1 ? 1 : 0));
                                                relation.setW2FixCount(wf1.getFixCount());
                                                relation.setW2FixCount(wf2.getFixCount());
                                                relation.setEntitiesBetween(entitiesBetween++);
                                                relation.setWordsBetween(j - i);
                                                relation.setVerb(verb);
                                                relationDao.save(relation);


                                    }
```

```
                }

            }

        }


        private static List<name.alp.tadba.model.Word> getVerb(Sentence sentence) {

            List<name.alp.tadba.model.Word> result = new
LinkedList<name.alp.tadba.model.Word>();

            WordDao wDao = new WordDao();

            for (Word word : sentence.getWords()) {

                WordType type = word.getType().equals(WordType.BILINMIYOR) ?
word.getRootType() : word.getType();

                if (type.equals(WordType.ADEYLEM) ||
type.equals(WordType.EYLEM)) {

                    result.add(wDao.findByProperty("word",
word.getRoot()).get(0));

                }

            }

            return result;

        }


        private static void saveInverseRelation(Sentence sentence, HashMap<String,
WordFixCounter> parses) {

            WordRelationDao relationDao = new WordRelationDao();
```

```
List<name.alp.tadba.model.Word> verbs = getVerb(sentence);

for (name.alp.tadba.model.Word verb : verbs) {



for (int i = sentence.getWords().size() - 1; i > 0; i--) {

int entitiesBetween = 0;

WordFixCounter wf1 =
parses.get(sentence.getWords().get(i).getWord());

if (wf1 == null) {

continue;

}

name.alp.tadba.model.Word word1 = wf1.getWord();

if (word1 == null) {

continue;

}

Fix fix1 = wf1.getFix();

for (int j = i - 1; j > 0; j--) {



WordFixCounter wf2 =
parses.get(sentence.getWords().get(j).getWord());

if (wf2 == null) {

continue;

}
```

```
                                        name.alp.tadba.model.Word word2 = wf2.getWord();
                                        if (word2 == null) {
                                                continue;
                                        }
                                        Fix fix2 = wf1.getFix();


                                        WordRelation relation = new WordRelation();
                                        relation.setWord1(word1);
                                        relation.setWord2(word2);
                                        relation.setFix1(fix1);
                                        relation.setFix2(fix2);
                                        if
(BaseCache.getCategory(sentence.getpClass().toLowerCase(Locale.ENGLISH)) == null)
{
                                                logger.error("null:"                    +
sentence.getpClass().toLowerCase(Locale.ENGLISH));
                                        }


        relation.setCategory(BaseCache.getCategory(sentence.getpClass().toLowerCase(
Locale.ENGLISH)));
                                        relation.setFirstCouple((j == 1 && i == 0 ? 1 : 0));
                                        relation.setLastCouple((j == i - 1 && i ==
sentence.getWords().size() - 1 ? 1 : 0));
                                        relation.setW1First(0);
```

```
                                    relation.setW1Last((i == sentence.getWords().size() -
1 ? 1 : 0));

                                    relation.setW2First((j == 1 ? 1 : 0));

                                    relation.setW2Last(0);

                                    relation.setW2FixCount(wf1.getFixCount());

                                    relation.setW2FixCount(wf2.getFixCount());

                                    relation.setEntitiesBetween(entitiesBetween++);

                                    relation.setWordsBetween(i - j);

                                    relation.setVerb(verb);

                                    relationDao.save(relation);


                                }


                            }

                        }

                }

        private static Word[] getWords(String sWord) {

                Word[] result = new Word[2];

                sWord = sWord.replaceAll("\\(", "").replaceAll("\\)", "").replaceAll("'", "");

                String shortening1 = sWord.replaceAll("\\.", "");

                if (BaseCache.hasShortening(shortening1)) {
```

```
                String root = sWord.replaceAll("\\.", "").replaceAll(",", "");
                String word = sWord;
                result[0] = new Word(root, word, WordType.OZELAD,
WordType.OZELAD);
                return result;
            }

            if (sWord.contains(".")) {
                if (sWord.endsWith(".")) {// /xxxx.
                    result = getWords(sWord.substring(0, sWord.length() - 1));
                    if (!result[0].getType().equals(WordType.ONAD)) {// sayi
varsayilir ve nokta ezilir
                        result[1] = new Word(".", ".", null, null);
                    }

                } else if (sWord.startsWith(".")) {// .xxxx
                    result = getWords(sWord.replaceFirst("\\.", ""));
                    result[1] = result[0];
                    result[0] = new Word(".", ".", null, null);
                } else if (sWord.split("\\.").length == 2) {// xxxx.yyyy
                    result[0] = getWords(sWord.split("\\.")[0])[0];
                    result[1] = getWords(sWord.split("\\.")[1])[0];
```

```
                        } else {// /xxx.yyy.zzz...
                                result[0] = validateAndTransform(sWord);
                                if (!result[0].getType().equals(WordType.ILGIDISI)) {
                                        String words[] = sWord.split("\\.");
                                        result[0] = validateAndTransform(words[0]);
                                        result[1] = validateAndTransform(words[words.length
- 1]);
                                }
                        }

                } else {
                        result[0] = validateAndTransform(sWord);// new Word(root, word,
type, rootType);
                }
                return result;
        }

        private static Word validateAndTransform(String sWord) {
                Word result = null;
                String encStr = new String(sWord.getBytes(), Charset.forName("ISO-8859-
1"));
                if                      (Validator.UrlValidator.isValid(encStr)                  ||
Validator.DomValidator.isValid(encStr) || Validator.EmValidator.isValid(encStr)) {
```

```
                String root = sWord.replaceAll("\\.", "");
                String word = sWord;
                result = new Word(root, word, WordType.OZELAD,
WordType.OZELAD);

            } else if (Validator.DateeValidator.isValid(sWord)) {

                String root = NON_INTEREST.DATE.getValue();
                String word = sWord;
                result = new Word(root, word, WordType.ILGIDISI,
WordType.ILGIDISI);

            } else if (Validator.IntValidator.isValid(sWord)) {

                String root = sWord.replaceAll("\\.", "").replaceAll(",", "");
                String word = sWord;
                result = new Word(root, word, WordType.ONAD,
WordType.ONAD);

            } else {
                String root = sWord.split("\\.")[0];
                String word = sWord;
```

```
                result = new Word(root, word, WordType.BILINMIYOR,
WordType.BILINMIYOR);
            }
            return result;
        }

        static Zemberek zemberek;

        private static Zemberek getZemberek() {
            if (zemberek == null) {
                zemberek = new Zemberek(new TurkiyeTurkcesi());
            }
            return zemberek;
        }

        private static boolean isPhraseParseble(Sentence sentence) {
            boolean result = false;
            try {
                if (sentence.getWords().size() > 30) {
                    return false;
                }
                PLThread pl = new PLThread(sentence);
```

```
                    pl.start();
                    while (true) {
                        Thread.sleep(1000);
                        logger.info("Thread running for :" +
(System.currentTimeMillis() - pl.getStartTime()) + "s");
                        if (pl.getStatus() == 1) {
                            pl.interrupt();
                            break;
                        } else if (System.currentTimeMillis() - pl.getStartTime() >
15000) {
                            pl.interrupt();
                            break;
                        } else {
                            continue;
                        }

                    }
                    result = pl.isParseable();
                    logger.info(result);
                    return result;

                } catch (Exception e) {
```

```
                    logger.error(e);

                }

                return result;

        }


}
```

2. Ek 2 Tümce Kontrol Yordamı

```
:- dynamic ad/1,eylem/1,ozelad/1,onad/1,adil/1,baglac/1,adEylem/1,belirtec/1.



Tumce --> adObegi , yuklem.

tumce --> icTumce , bagTumce.


adObegi--> ozne , nesne ; ozne.


ozne --> adil ; ozeladObegi ; adtamlama.


nesne --> ozeladObegi ; adtamlama ; icTumce.
```

icTumce --> adObegi , yuklem.

bagTumce --> baglac , icTumce.

bagTumce --> icTumce , icTumce.

bagTumce --> icTumce ,bagTumce.

yuklem --> adEylem;eylem.

yuklem --> belirtecObegi,adEylem;belirtecObegi,eylem.

adtamlama --> ad.

adtamlama --> ozelad.

adtamlama --> baglac,adtamlama.

adtamlama --> ad,adtamlama.

adtamlama --> onadtamlama,ad.

onadtamlama --> onad.

onadtamlama --> onad,onadtamlama.

ozeladObegi --> ozelad.

ozeladObegi --> ozelad , digerOzeladObegi.

ozeladObegi --> ozelad , ozeladObegi.

```
digerOzeladObegi -->baglac , ozeladObegi.


ozelad --> [OZEL], {ozelad(OZEL)}.

Ozelad(zehrasini).

ozelad(alp).

ozelad(kumru).


adil --> [ADIL], {adil(ADIL)}.

Adil --> belirtecObegi,adil.

adil(ben).

adil(biz).

adil(ondan).


belirtecObegi --> belirtec.

belirtecObegi --> belirtec,belirtecObegi.


belirtec --> [ADIL], {belirtec(ADIL)}.

Belirtec(az).



eylem --> [EYL], {eylem(EYL)}.
```

```
Eylem(severmis).

eylem(severim).

eylem(dedi).

eylem(kosarim).

eylem(surerim).



onad --> [ONAD], {onad(ONAD)}.

Onad(ufak).

onad(sirin).

onad(guzel).

onad(hizli).

onad(kucuk).

onad(buyuk).

onad(az).



ad --> [AD], {ad(AD)}.

Ad(ev).

ad(ofis).

ad(araba).



adEylem --> [ADEYL], {adEylem(ADEYL)}.

adEylem(guzelmis).
```

```
adEylem(arabasiymis).



baglac -->[BAG],{baglac(BAG)}.

Baglac(ve).

baglac(veya).

baglac(ile).
```

3. Ek 3 Prolog Java Haberleşme Yordamı

```
package name.alp.prolog.parser;



import java.io.File;

import java.util.List;

import java.util.ResourceBundle;

import java.util.StringTokenizer;



import javax.management.RuntimeErrorException;



import name.alp.prolog.main.Factorgiting;
```

```
import org.apache.log4j.Logger;



import jpl.Atom;

import jpl.Compound;

import jpl.Query;

import jpl.Term;

import jpl.fli.Prolog;



public class PLAccessor {

	static PLAccessor analyser = new PLAccessor();

	static Logger logger = Logger.getLogger(Factorgiting.class);



	protected PLAccessor() {

		initialize();

	}



	public static PLAccessor getAnalyser() {

		return analyser;

	}



	private void initialize() {

		ResourceBundle properties =
```

```
ResourceBundle.getBundle("prologproperties");

		Prolog.set_default_init_args(properties.getString("pl_init_args").split(";"));

		File plPath = new File(getClass().getResource("Test10.pl"));

		Compound goal1 = new Compound("consult", new Term[] { new Atom(plPath.getPath()) });

		Query q1 = new Query(goal1);

		if (!q1.hasSolution()) {

			logger.error("consult[ing] " + plPath + " failed!");

			return;

		}

	}


	public void assertNew(List<String> asserts, WordTypeKeys type) {


		for (String sassert : asserts) {

			sassert = convertChar( sassert.toLowerCase());

			String q1 = type.getValue() + "(" + sassert + ")";

			Query q2 = new Query(q1);

			if (!q2.hasSolution()) {

				q1 = "assert(" + type.getValue() + "(" + sassert + "))";

				q2 = new Query(q1);

				logger.error(q1);
```

```
                    if(!q2.hasSolution()){
                        throw new RuntimeException(q1 + " : " + q2.hasSolution());
                    }
                }
            }
        }

        private String convertChar(String word) {
            return word.replaceAll("ı", "i").replaceAll("ö", "o").replaceAll("ç", "c").replaceAll("ü", "u").replaceAll("ş", "s").replaceAll("ğ", "g");
        }

        public void testAsserts(List<String> asserts, WordTypeKeys type) {

            for (String sassert : asserts) {
                sassert = sassert.toLowerCase();
                String q1 = type.getValue() + "(" + sassert + ")";
                Query q2 = new Query(q1);
                logger.error(q1 + q2.hasSolution());
            }
        }
```

```
private String buildQuery(String phrase) {
        StringBuffer result = new StringBuffer("tumce([");
        StringTokenizer token = new StringTokenizer(phrase, " ");
        while (token.hasMoreTokens()) {
                result.append(convertChar(token.nextToken().toLowerCase()));
                result.append(",");
        }
        result.deleteCharAt(result.lastIndexOf(","));
        result.append("],[])");
        return result.toString();
}

public boolean isPhraseParseble(String phrase, int pid) {
        boolean result = false;
        try {
                String query = buildQuery(phrase);
                Query q2 = new Query(query);
                logger.error(query);
                result=q2.hasSolution();
        } catch (Exception e) {
                logger.error(e);
```

```
                    return false;

                }



                return result;

            }




}
```

4. Ek 4 Destek Vektör Makinesi parametreleri

weka.classifiers.functions.LibSVM

DVM tipi : C-SVC

Önbellek (cache size): 120

Katsayı (coefficient) : 0

Maliyet (cost) : 1.0

Çekirdek Derecesi (degree) : 3

Tolerans (epsilon) : 0.001

Çekirdek tipi (kernel type) : Radyal tabanlı fonksiyon