FREEQA - BAĞLI VERİ ÜZERİNDE HİBRİD SORU CEVAPLAMA SİSTEMİ

ŞENOL ATAÇ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AĞUSTOS 2015

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Erdoğan DOĞDU
Anabilim Dalı Başkanı

ŞENOL ATAÇ tarafından hazırlanan FreeQA - BAĞLI VERİ ÜZERİNDE HİBRİD SORU CEVAPLAMA SİSTEMİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Prof. Dr. Erdoğan DOĞDU
Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Doç. Dr. Pınar KARAGÖZ _____

Üye : Prof. Dr. Erdoğan DOĞDU _____

Üye : Yard. Doç. Dr. Murat ÖZBAYOĞLU _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Şenol ATAÇ

Üniversitesi	**:**	**TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü	**:**	**Fen Bilimleri**
Anabilim Dalı	**:**	**Bilgisayar Mühendisliği**
Tez Danışmanı	**:**	**Prof. Dr. Erdoğan DOĞDU**
Tez Türü ve Tarihi	**:**	**Yüksek Lisans – Ağustos 2015**

Şenol ATAÇ

FREEQA - BAĞLI VERİ ÜZERİNDE HİBRİD SORU CEVAPLAMA SİSTEMİ

ÖZET

"Soru cevaplama" (Question Answering) sistemleri doğal dilde verilen soruların bilgi tabanları kullanılarak cevaplanması problemi ile ilgilenmektedirler. DBpedia, Freebase, Yago gibi bağlı veri (linked data) kaynaklarının artması ile birlikte soru cevaplama sistemleri bu tür kaynakları daha yoğun olarak kullanmaya başlamışlardır. IBM Watson, Google Knowledge Graph, Wolfram Alpha, Siri bu tür sistemlere örnek olarak verilebilir. Bağlı veri üzerinden soru cevaplama sistemleri temelde yapısal (genelde RDF) veriler ile cevap vermeye çalışmaktadır. Ancak bu tür sistemlerde cevabı yapısal veri içinde yer almayan bazı sorular sadece yapısal veri ile cevaplanamamaktadır. Bu durumda bağlı verilerde yer alan yapısal olmayan verilerin de kullanılması ile sonuç alınabilmektedir. Buna "hibrid soru cevaplama" problemi denmektedir. Bu çalışmada hibrid soru cevaplama sistemine farklı bir bakış açısı sunulmaktadır. Bu sunulan çalışma da FreeQA diye adlandırılmaktadır. FreeQA, doğal dildeki soruları doğal dil işleme araçlarıyla işleyip sorgu kalıpları çıkarmaktadır ve sonrasında bu kalıplardan yararlanarak, bağlı veri üzerinde çalıştırılabilir ve bir cevap alınabilir SPARQL sorguları oluşturmaktadır. FreeQA, ilk olarak mümkün olan tüm sorgu kalıplarıyla bir sorgu ağacı oluşturmaktadır ve bu ağaç üzerinde aç gözlü arama (greedy search) stratejisi ile arama yaparak tekil SPARQL sorguları elde etmektedir. Bu stratejiyi kullanarak FreeQA değerlendirildiğinde 45 adet hibrid sorusu içinden 37'sine cevap verilmektedir, bunlardan 33 tanesine doğru cevap ve 4 tanesine de kısmen doğru cevap verebilmektedir.

Anahtar Kelimeler: Bağlı Veriler, Soru Cevaplama, Hibrid Sorular, Doğal Dil Soruları.

University	**:**	**TOBB University of Economics and Technology**
Institute	**:**	**Institute of Natural and Applied Sciences**
Science Programme	**:**	**Computer Engineering**
Supervisor	**:**	**Prof. Dr. Erdoğan DOĞDU**
Degree Awarded and Date	**:**	**M.Sc. – August 2015**

Şenol ATAÇ

FREEQA - HYBRID QUESTION ANSWERING SYSTEM ON LINKED DATA

ABSTRACT

Question answering deals with natural language questions that can be answered using knowledge resources. With increasing in linked data resources such as DBpedia, Freebase and Yago, question answering systems start to use these types of resources more intensively. IBM Watson, Google Knowledge Graph, Wolfram Alpha, Siri can be example for these systems. Question answering systems over linked data basically try to answer using structured data. However, not all questions can be answered using only structured data. In such cases, textual or unstructured data, which are present in linked data as abstracts, labels, etc., can be utilized to answer such queries. This is called hybrid question answering. In this thesis, there is a different approach to hybrid question answering called FreeQA. This system utilizes natural language tools to map natural language questions to query templates and then to SPARQL queries, which can then be executed on linked data to answer the questions. FreeQA first builds a query tree with possible query templates and uses greedy search strategy to form the final SPARQL query. FreeQA can answer 37 of the 45 hybrid questions in the dataset, of which 33 of them are answered correctly and 4 of them are partially correctly answered.

Keywords: Linked Data, Question Answering, Hybrid Questions, Natural Language Questions.

TEŞEKKÜR

İlk olarak, benim buralara gelmemde çok büyük emeği olan anneme, babama ve kardeşlerime teşekkür ederim. Tezimin ortaya çıkmasında çok büyük pay sahibi olan danışman hocam Prof. Dr. Erdoğan DOĞDU'ya da teşekkürü bir borç bilirim. Tez jürimde yer alan Yrd. Doç. Dr. Murat ÖZBAYOĞLU'na ve Doç. Dr. Pınar KARAGÖZ'e, ilkokuldan bu zamana kadar üzerimde emeği olan bütün hocalarıma, lisans ve yüksek lisans eğitimim boyunca destekleriyle yanımda olan tüm arkadaşlarıma ve okuluma teşekkürü bir borç bilirim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

1. GİRİŞ

Son zamanlarda web'de çok fazla yapısız, düzensiz veri (unstructured data) düzenli ve yapısal (structured data) hale gelmekte ve bu oran gittikçe artmaktadır. Birbirlerine RDF [16] (Resource Description Framework) veri kümeleri ile bağlı olan bağlı veri bulutları (data linked cloud) günümüzde 30 milyondan fazla RDF üçlüsü içermektedir. Freebase [3], Yago [28] ve DBpedia [1] gibi bağlı veri kümeleri bu alanda kullanılan en büyük uygulamaların başında gelmektedir ve önemli uygulamalar tarafından kullanılan popüler veri kümeleridir. Ayrıca düzenli veri Google gibi bazı arama motorları tarafından da keşfedilmekte ve sağlanmaktadır. Örnek olarak Google'ın sağlamış olduğu knowledge graph'da sonuçları iyileştirmek için sürekli düzenli veri üretilmektedir.

Soru cevaplama sistemleri doğal dildeki soruları alıp bağlı verileri (linked data) [2] kullanarak soruları cevaplamaya çalışan uygulamalardır. Bağlı verilerde yapısal verilerin çok olmasına rağmen, tüm soruları yalnız yapısal veriyle çözmek mümkün değildir. Bu yüzden hibrid soru cevaplama yaklaşımı, yani yapısal verinin ve yapısız verinin her ikisinide aynı anda kullanılması, doğal dildeki soruların cevaplanması adına önemlidir.

Hibrid soru cevaplama, soru cevaplama alanında yeni bir tekniktir. Bu teknikle ilgili birkaç çalışma vardır. Ancak bu çalışmalar yavaş (en iyi performans 5sn), yetersiz (tüm bilgileri kapsamıyor) ve soru çözmede başarısızdır (en iyi performans %60 civarında). Ek olarak bu çalışmalar sadece belli soru tipleri için çalışmaktadır, evet/hayır soruları ve sayısal sorular gibi sorular için çalışmamaktadır.

Soru cevaplama ile ilgili var olan yarışmalardan bir tanesi QALD[1]'dir. QALD her yıl düzenli olarak soru cevaplama hakkında sorular hazırlayarak bu konuda geliştirilen uygulamaları test ederek değerlendirmektedir. Hazırlanan sorular belli kategorilerden oluşmaktadır. Bunlar: çok dil (multilingual) destekli soru cevaplama, biyomedikal soru cevaplama ve hibrid soru cevaplamadır.

HAWK [32], hibrid soru cevaplama alanında ortaya konan ilk çalışmadır. HAWK, şablonlardan (templates) bağımsız ve sorunun yapısal analize bağlı genel bir yaklaşım sunmaktadır. FreeQA, hibrid soru cevaplamada HAWK'u temel almıştır. Bu kapsamda HAWK'un kullandığı methodlardan ve HAWK'un sunmuş olduğu 0.1.0 versiyonlu açık kaynak kodundan yararlanmıştır.

Bu çalışmada bir hibrid soru cevaplama yaklaşımı olan FreeQA sunulmaktadır. FreeQA yukarıda bahsedilen problemleri günümüzde kullanılan hibrid soru cevaplama

[1] http://greententacle.techfak.uni-bielefeld.de/ cunger/qald/

sistemlerine uyarlamaya çalışmaktadır. FreeQA yavaşlık problemini çözmek için aç gözlü arama algoritması (greedy search) kullanıp, doğal dil girdilerini semantik olarak doğru eşleştirmek için yeni bir varlık tanıma (custom entity recognition) ve jenerik semantik ilişki yakalama metodu ve tüm soru çeşitleri için jenerik soru cevaplama önermektedir. Kısaca FreeQA'in yaptıkları sıralanacak olursa: (1) soruyu parçalama ve etiketleme (part-of-speech tagging), (2) soru tipi tanımlama, (3) varlıkları (entity) yakalama ve anlam karmaşıklığını çözme (disambiguation), (4) doğal dildeki soruyu bağımlılıklarına göre ayrıştırma (dependency parsing), (5) bağımlılıklara göre dilsel birleşmeleri uygulama ve anlamsız kalan kelimeleri budama, (6) bağımlılık ağacı bileşeneleri semantik olarak DBpedia kaynakları ile eşleştirme, (7) çizge (graph) yaklaşımına göre SPARQL sorgularını üretme ve aç gözlü arama algoritması (greedy search) ile cevapları bulma.

FreeQA'in katkıları özetle şunlardır:

1. Hibrid doğal dil sorularının bağlı veri kaynakları ile başarılı bir şekilde cevaplanması.
2. Pekçok soru tipine (Evet/hayır, sayısal ve Wh soruları) cevap verebilmesi.
3. Sorularda geçen varlıkların (named entity) tanınmasının daha başarılı yapılması.

Tezin içeriği şu bölümlerden oluşmaktadır: 2.bölümde soru cevaplama problemi hakkında genel bilgilerden bahsedilmektedir. İlgili çalışmalar 3.bölümde tartışılmaktadır. 4.bölümde hibrid soru cevaplama için FreeQA'in yaklaşımı sunulmaktadır. FreeQA'in performansı ve hata analizi 5.6.bölümde sunulmaktadır. Son olarak da 6.bölümde FreeQA çalışması özetlenmekte ve gelecek çalışmalarda neler yapılabileceği incelenmektedir.

2. SORU CEVAPLAMA

Bu bölümde soru cevaplamanın temelinde yatan genel bilgilerden, bağlı veriden ve soru cevaplama ile hibrid soru cevaplama arasında nasıl bir fark olduğundan bahsedilecektir.

2.1 Semantik Web ve Bağlı Veri (Linked Data)

İnternette milyarlarca veri vardır ve bu veriler gün geçtikçe artmaktadır. Bu verilerin büyük kısmı kendi içinde tutarlı olsa da diğer uygulamalar tarafından kullanılamamaktadır. Ayrıca bu veriler insanlar tarafından kolayca anlaşılsa da makineler tarafından anlaşılamamaktadır. Semantik web, verilerin anlaşılır yapılmasına ve verilerin ilişkilendirilmesi temeline dayanmaktadır. Bu ilişkilerin oluşturduğu veri kümelerine bağlı veri (Linked Data) denir. Bağlı veriler belli bir formatta tutulmaktadır. Bu veri formatına RDF[1] (Resource Description Framework) denir. RDF, bilgiyi makinelerin anlayabileceği şekilde ifade etmektedir. RDF verisi oluşturmak için kullanılan veri şemalarına RDFS[2] (RDF Schema) denir. Örnek RDF:

```
<?xml version= "1.0" encoding = "UTF-16" ?>
<rdf:RDF xmlns:rdf ="http://www.w3c.org/1999/02/22-rdf-syntax-ns#"
xmlns:agent="www.example.com/agent">
<rdf:Description rdf:about="www.example.com/agent/Martin Luther King">
<agent:birthName>Michael King, Jr.</agent:birthName>
<agent:birthPlace>Atlanta</agent:birthPlace>
</rdf:Description>
</rdf:RDF>
```

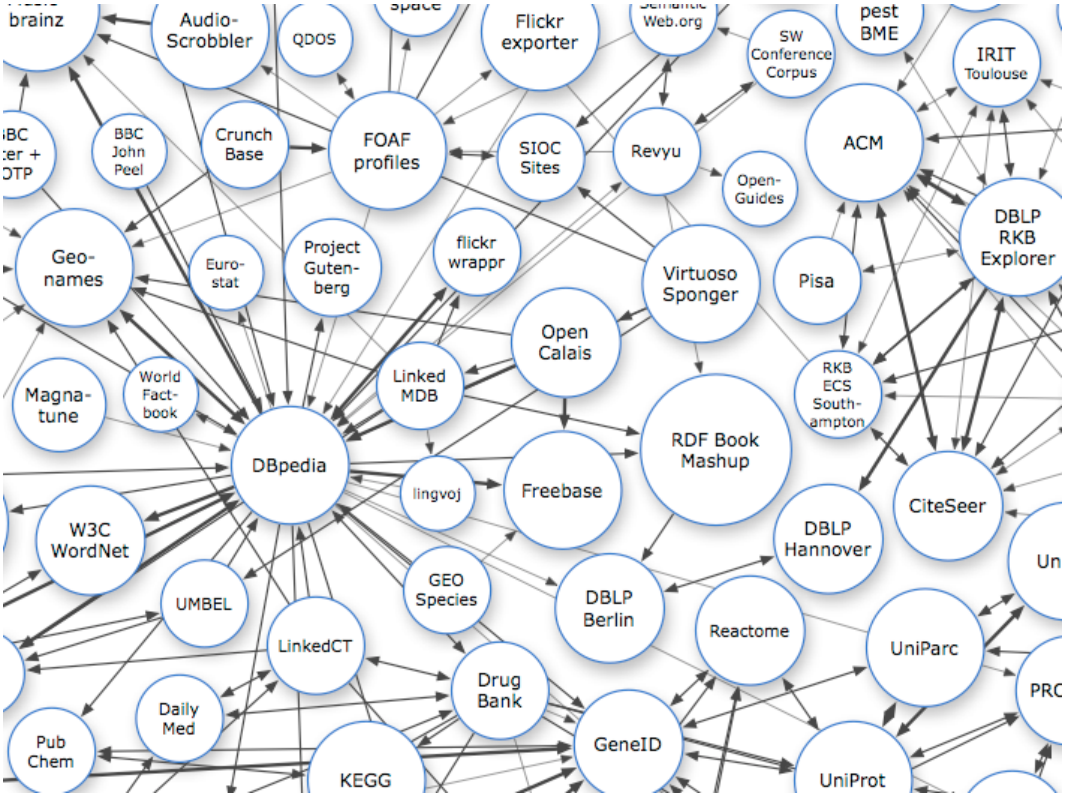
Bağlı veri için DBpedia[3] en güzel örnektir. DBpedia, Wikipedia[4]'dan elde edilen bilgilerin oluşturduğu bir veri kümesidir. Wikipedia'dan gelen veriler, DBpedia'da RDF formatında üçlüler (triple) halinde bulunmaktadır. Örneğin, Bill Gates[5] hakkında yazılmış olan Wikipedia kaynağında geçen bir yazı "Gates was born in Seattle" dır. Bu yazının DBpedia[6]'da karşılığı "Bill Gates, dbo:birthPlace, Seattle" üçlüsüdür. Görüldüğü üzere DBpedia'da Wikipedia verileri yapısal olarak tutulmaktadır.

[1] http://www.w3.org/TR/rdf-primer/
[2] http://www.w3.org/TR/rdf-schema/
[3] http://dbpedia.org/
[4] https://en.wikipedia.org
[5] https://en.wikipedia.org/wiki/Bill_Gates
[6] http://dbpedia.org/page/Bill_Gates

Bağlı veri yapısında bilgiler SPARQL[7] ile sorgulanabilmektedir. Örnek SPARQL:

```
Select ?x Where
?x a <http://dbpedia.org/ontology/Agent>
LIMIT 100
```

Ayrıca bağlı veri üzerinde sınıflar arası ilişkileri işlemek için kullanılan dile OWL (Web Ontology Language) denir.



Şekil 2.1: Bağlı veri bulutu (http://linkeddata.org/)

2.2 Soru Cevaplama

Soru cevaplama, doğal dilde sorulan sorulara cevap döndürülmesidir. Bu dönen cevap arama motorlarının yaptığı gibi bir bilgiye yönlendirme değildir, net bir bilgidir. Örneğin, "Who is assassin of Martin Luther King"; bu soru Google vb. arama motorlarına sorulduğunda alınan cevap soruda geçen kelimeleri içeren sayfaların listesidir ama soru cevaplama sistemleri için dönen sonuç "James Early Ray" yani

[7] http://www.w3.org/TR/rdf-sparql-query/

sorunun cevabıdır. Soru cevaplama sistemleri ilk olarak 1960'lı yıllarda ortaya çıkmaya başlamıştır. Bu konuda ilk geliştirilen sistemler ise BASEBALL [11] ve LUNAR'dır [33].

Soru cevaplama sistemlerine dört farklı tipte soru yönlendirebilirler. Bunlar: tek bir cevabı olan sorular, listeleme soruları, tanım soruları ve yorum sorularıdır. Bu sorular için örnekler sırasıyla şunlardır: "Where is the capital of Turkey?", "List all footballers in Barcelona.", "Who is Tom Cruise?" ve "Is TOBB ETU a qualified university?".

Günümüzde soru cevaplama ile ilgili çalışmalar devam etmektedir. Bu çalışmalardan en önemlileri Watson[8], Siri[9], Knowledge Graph[10] ve Facebook Graph Search[11]'dir.

Watson, IBM tarafından 2010 yılında sunulan bir soru cevaplama sistemidir. Watson devasa büyüklükteki veri kaynaklarını hızlı bir şekilde tarayıp anlamsal bilgiler çıkarmaktadır. Diğer bir taraftan Watson 2011 yılında Jeopardy adında bir yarışmada yarıştırılmıştır ve en iyi iki yarışmacı ile rekabet ederek yarışmayı kazanmıştır.

Siri, Apple tarafından 2010 yılında sunulan bir uygulamadır. Bu uygulama ürün verilerini, interneti, konum bilgilerini, önceden sorulan soruları ve cevaplarını kullanarak çalışan bir soru cevaplama sistemidir. Siri tarafından cevaplanabilen sorulardan bazıları şunlardır: "How is weather?", "How can I go to Istanbul?" vb.

Knowledge Graph, Google tarafından 2012 yılında ortaya konulan çizge temelli bir uygulamadır. Freebase[12] bilgi veritabanını kullanarak sorulara cevap vermektedir. Freebase kaynaklarından gelen ilişki örüntüsü ile çizge oluşturmaktadır. Örneğin, Google'da "Tom Cruise" araması yapılırsa, "Tom Cruise" hakkında Wikipedia bilgisi ve ilişkili olduğu kişiler ("Katie Holmes", "Mimi Rogers" vb.) gelmektedir.

Facebook Graph Search, Facebook tarafından 2013 yılında sunulan çizge temelli soru cevaplama sistemidir. Facebook verilerini (kişi bilgileri, arkadaş ilişkileri vb.) kullanmaktadır. Örneğin, "People who are from my country?" gibi sorulara cevap vermektedir.

Soru cevaplama hakkında daha fazla bilgi için var olan iki araştırma ([17], [19]) incelenebilir.

[8] http://www.ibm.com/smarterplanet/us/en/ibmwatson/
[9] http://www.apple.com/tr/ios/siri/
[10] https://en.wikipedia.org/wiki/Knowledge_Graph
[11] https://en.wikipedia.org/wiki/Facebook_Graph_Search
[12] https://www.freebase.com/

2.3 Bağlı Veri Kaynakları ile Soru Cevaplama

Bağlı veri, bilginin anlamsal olarak modellenip, bu bilgilerin biribiri ile ilişkilendirilmesidir. Bağlı veri kaynaklarına örnek olarak FreeBase, DBpedia ve Yago verilebilir. Bilgi veritabanları yapısal olarak bünyelerinde iki tür veri barındırmaktadırlar. Bunlar; yapısal (structured) ve yapısız (unstructured) verilerdir. Yapısal veri, bilgi veritabanlarında tutulan yapılı, formatlı verilerdir. Bu tür verilere örnek üçlü; `"Martin Luther King, birthPlace, Atlanta"`. Bu örneğe göre Martin Luther King'in doğum yeri (birth place) bilgisi yapısal olarak tutulmaktadır. Yapısız veri ise bilgi veritabanlarında doğrudan karşılığı olmayan verilerdir. Yapısız veri için örnek; `"Martin Luther King, abstract, ...when he was assassinated on April 4 in Memphis..."`. Burada Martin Luther King'in assassin bilgisine doğrudan ulaşılamamaktadır. Ancak bu bilgiye `"abstract"` bilgisi içinden arama sonucunda ulaşılabilir.

Pek çok soru cevaplama sistemi bağlı veri kaynaklarını kullanmaktadır. Google Knowledge Graph, Freebase'i kullanan sistemdir. Treo [10], PowerAqua [18] ve HAWK [32] DBpedia'yı kullanan başlıca soru cevaplama sistemleridir.

2.4 Hibrid Soru Cevaplama (Hybrid Question Answering)

Soruların cevaplandırılması için kullanılabilecek veriler yapısal (structured) ve yapısız (unstructured) verilerdir. Bazı soruların cevaplandırılması için sadece yapısal verilerin kullanılması yeterli değildir, yapısal ve yapısız verilerin her ikisinin bir arada kullanılması gerekmektedir. Bu tür sorulara Hibrid Soru denilmektedir. Bu sorulara örnek vermek gerekirse "In which city was the assassin of Martin Luther King born?". Bu soruda "Martin Luther King"'in "assassin" bilgisine yapısal verilerden ulaşılamamaktadır onun için "assassin" yapısız veri olarak ele alınmaktadır. Ama "born in" ve "city"'ye yapısal veri olarak ulaşılabilmektedir.

```
born in:  "Martin Luther King, birthPlace, Atlanta"
city:  "Atlanta, type, Settlement"
assassin:  "Martin Luther King, abstract, ...when he was assassinated on
April 4 in Memphis..."
```

Bu üçlülerden elde edilen nihai SPARQL sorgusu aşağıdaki gibidir. SPARQL sorgusunda belirtilen "birthPlace" ve "Settlement" bilgisine doğrudan yapısal veri

olarak ulaşılabilir ama "assassin" bilgisine ulaşmak için kaynakta yapısız verilerde arama yapılması gerekir. Bu arama SPARQL sorgusunda "text:" komutuyla yapılmaktadır.

```
SELECT distinct ?y WHERE {
  res:Martin\_Luther\_King,\_Jr. birthPlace ?y.
  ?y a Settlement.
  res:Martin\_Luther\_King,\_Jr. text:"assassin"
}
```

Hibrid soru cevaplama QALD [31] yarışması tarafından 2014 yılında sunulan bir yaklaşımdır. QALD, 2010 yılından bu yana süre gelen ve son beş yıldır düzenli yapılan bir yarışmadır. Bu yarışma CLEF[13] konferansı kapsamında yapılmaktadır. QALD, uygulamaları üç farklı kapsamda değerlendirmektedir. Bunlar: (1) Çok dilli soru cevaplama, (2) biyomedikal soru cevaplama, (3) Hibrid soru cevaplama. QALD, bu üç farklı kapsamda sistemleri kendi test verileri ile değerlendirmektedir.

Hibrid soru cevaplama, yeni bir yaklaşım olduğu için yapılan çalışmalar da azdır. Bu yapılan çalışmalardan bir tanesi HAWK tarafından sunulmuştur.

[13] http://www.clef-initiative.eu/

3. İLGİLİ ÇALIŞMALAR

Literatürde hibrid soru cevaplama üzerine yapılmış sınırlı sayıda çalışma mevcuttur. Ancak literatürde, soru cevaplama üzerine yapılmış çok sayıda çalışma bulunmaktadır. Bu çalışmada, hem yapısal veri hem de yapısız veri kullanıldığı için bu konular üzerinde literatür araştırması yapılmıştır.

Yapısal veri üzerine yapılan çalışmalarda temel alınan bilgi veritabanları DBpedia, Yago ve Freebase'dir. Bunların kullanılmasının temel nedeni veritabanlarının çok geniş olmasıdır. Ayrıca istenilen bilgiye sorgularla (SPARQL, MQL) ulaşma imkanı sağlamaktadır.

Bu bölümde literatürde yapılan çalışmalar üç başlık altında incelenecektir. Bunlar sırasıyla; soru cevaplama, bağlı veri üzerinde soru cevaplama ve hibrid soru cevaplamadır.

3.1 Soru Cevaplama

Soru cevaplama sistemi, bilgisayar dillerinin ilk çıkışından bu yana süre gelen bir araştırma konusudur. 1960'lardan bu yana birçok soru cevaplama sistemi geliştirilmiştir [26]. Soru cevaplama üzerine yapılan ilk çalışmalar özel alanlara yönelik geliştirilmiştir. BASEBALL [11], bu sistemlere örnek olarak verilebilecek bir soru cevaplama sistemidir. BASEBALL, Amerika Bezybol Ligi istatistiklerini temel alarak sorulan soruları cevaplayan bir sistemdir. LUNAR [33], ay taşları ile ilgili sorulan soruları cevaplamak için tasarlanan diğer bir soru cevaplama sistemidir. LUNAR, ilk test edilen soru cevaplama sistemlerinden biridir. LUNAR, 111 soru ile test edilmiş ve %78 oranında başarılı olmuştur. BASEBALL ve LUNAR'ın en belirgin ortak özelliği kendi bigilerini depolamak için veritabanlarını kullanmasıdır. Sorulan sorular veritabanı sorgularına dönüştürülmektedir. Dönüştürülen sorgulara, eğer veritabanından ulaşılabiliyorsa sistem sağlıklı çalışmaktadır.

İlk ağ (web) tabanlı soru cevaplama sistemleri 1990'lı yıllarda ortaya çıkmıştır. START [15], bu sistemlere örnek olarak verilebilecek bir soru cevaplama sistemidir. START, ağdan (web) bilgi veritabanı elde etmektedir ve bu bilgi veritabanını kullanarak soruları cevaplamaktadır. START, bilgi veritabanında verileri üçlüler (Özne, İlişki, Nesne) halinde tutmaktadır. START, gelen soruyu üçlüler haline çevirmektedir ve çevrilen üçlüler bilgi veritabanında aranmaktadır. Eğer aranan üçlüye bilgi veritabanında

ulaşılırsa cevap döndürülmektedir.

2000'li yılların başından itibaren soru cevaplama üzerine yapılan çalışmalar yaygınlaşmıştır. Bu çalışmaların başlıca örnekleri Wolfram Alpha[1] ve IBM'in Watson[9] uygulamalarıdır. Wolfran Alpha, çevrimiçi servis sağlayan iyi bilinen matematiksel soruları cevaplayan bir sistemdir. Wolfram Alpha, ayrıca tanım soruları ("Who is Putin?") ve tek cevaplı soruları da ("What is the capital of Turkey?") cevaplamaktadır. Bu uygulamalardan diğeri ise Watson'dır. Watson 2011 yılında Jeopardy adlı programda en iyi iki yarışmacı ile yarışmada yarıştırılmıştır ve oyunu kazanmıştır. Watson çok sayıda veriyi derleyerek anlamlı hale getirmektedir.

Soru cevaplama ile ilgili daha detaylı bilgi için şu araştırmalar ([17], [19]) incelenebilir.

3.2 Bağlı Veri Üzerinde Soru Cevaplama

Günümüzde birçok düzensiz veri yapısal hale getirilmektedir. Bu yapısal veriler bilgi veritabanlarında birbirine anlamsal olarak bağlı bir şekilde tutulmaktadır. Bu bilgi veritabanlarının başlıca örnekleri; DBpedia, Yago ve Freebase'dir. DBpedia [1], 103 milyon civarında birbirine bağlı RDF üçlüsü içermektedir. Yago [28], yaklaşık olarak 5 milyon RDF üçlüsü içermektedir. Freebase [3], yaklaşık olarak 125 milyon RDF üçlüsü içermektedir. Bunlarla ilgili yapılan çalışmalara bu başlık altında değinilmektedir.

Unger vd. [29] soru cevaplama için taslak tabanlı yaklaşım önermektedir. Doğal dildeki soruları önceden belirlenen özel taslaklarla eşleştirmeye çalışmaktadır. Taslak kullanmak performans açısından iyi olabilir. Ancak pek çok soru taslağı vardır ve bu soru tiplerinin hepsini özel taslaklarla kavramak mümkün değildir.

Damljanovic vd. [6] FreyA'yı sunmaktadırlar. FreyA doğal dil arayüzlerini kullanırken anlamsal karmaşıklık (ambiguity) problemini ele almaktadır. Semantik Web'de çoğu ilişkiyi eşleştirmek zordur. Örneğin; soru "How long..." ile başlıyorsa bu "zaman" veya "uzaklık" olarak elde edilebilir. FREyA bu problemleri kullanıcıdan gelen geri dönütlerle çözmektedir. Kullanıcı tercihleri FREyA için belirleyici olmaktadır. Bu yüzden kullanıcılar sisteme müdahale edebilmektedirler. Bu nedenle sistemde bir denetçiye gereksinim duyulmaktadır.

Lopez vd. [18] PowerAqua'yı tanıtmaktadırlar. PowerAqua farklı ve heterojen kaynakları kullanarak çalışan soru cevaplama sistemidir. PowerAqua parçala ve fethet

[1] http://www.wolframalpha.com/

prensibine göre çalışmaktadır. İlk olarak kullanıcıdan gelen soru parçalanmaktadır ve semantik ilişkiler farklı bilgi veritabanlarından bulunmaktadır. Sonrasında tüm parçalar bir sorgu içerisinde birleştirilmektedir. Heterojen verileri kullanmak gelecek çalışmalar için önemlidir ama heterojen kaynakları kullanmak sakıncalıdır çünkü veriler tutarlı değildir ve kirlidir.

Lei vd. [35] gAnswer adında bir sistem önermektedirler. gAnswer çizge odaklı yaklaşımı temel almaktadır. Sorunun bağlılık (dependency) ilişkilerini göz önünde bulundurarak soruyu bir çizgeye dönüştürmeye çalışmaktadır. Çizge alt çizgelerden (sub graph) oluşmaktadır. Bu noktada en iyi kompakt alt çizgeyi bulmaya çalışmaktadırlar. gAnswer sorulara çok kısa sürede cevap verebilmektedir ama varlık tanımlamada başarılı değildir. Ayrıca sayısal ve evet/hayır sorularına cevap verememektedir.

Xu vd. [34] Xser'i sunmaktadır. Xser doğal dildeki soruları girdi olarak almaktadır ve cevabı iki adımda vermektedir. İlk adım olarak, sorunun dilsel olarak analizini yapmaktadır. İkinci adım ise bilgi veritabanı ile ifadelerin ilişkisini kurmaktadır. Xser ifadeleri yakalamak için DAG (Directed Acyclic Graph) algoritmasını kullanmaktadır. İfade yakalama tam olarak doğru çalışmamaktadır ve deneme veri kümesine ihtiyaç duymaktadır.

CASIA [13] Markov Mantık Ağları (Markov Logic Network) temelli bir algoritma sunmaktadır. Bu algoritmayı kullanarak ifade yakalaması yapmaktadır. Sonrasında ifadeleri semantik olarak eşleştirmektedir ve eşleşen ifadeleri çizgede gruplamaktadır. CASIA, sayısal olmayan (non-aggregation) sorular için başarılı bir şekilde çalışmaktadır ama sayısal (count, filter, compare...) soru tipleri için çalışmamaktadır.

Intui3 [7] doğal dildeki soruyu girdi olarak almaktadır. Soruyu sözdizimsel ve semantik olarak yorumlamaktadır. İlk olarak, soru parçalara ve yığınlara ayrılmaktadır. Sonrasında her bir yığın yorumlanmaktadır. Son olarak bu yorumlar birleştirilmektedir ve SPARQL sorgusu elde edilmektedir. Intui3 evet/hayır sorularına ve üstünlük belirten sorulara (superlative) cevap verememektedir.

ISOFT [24] problemleri çözmek için taslak temelli yaklaşımı (template-based approach) öne sürmektedir. İlişkileri metin benzerliği (string similarity) ve belirgin semantik analizine (Explicit Semantic Analysis) göre bulmaktadır. ISOFT verilen soruyu doğru kelime öbeklerine ayırmada ve semantik eşlemede başarılı değildir. 50 sorudan 22 sorunun kelime öbeğini ve semantik eşlemesini doğru bulamıyorlar.

Shekarpour vd. [25] ilişkili veri kümeleri arasında anahtar kelime (keyword) temelli

aramaya dayanan soru cevaplama sistemi SINA'yı geliştirmektedir. Sistem doğru veri kümesini seçmek için Hidden Markov Model'i temel almaktadır. Hidden Markov Model'den dolayı cevaplama süresi uzun sürmektedir.

Unger vd. [30] Phytia'yı iki aşamaya dayanan bir soru cevaplama sistemi olarak tanımlamaktadır. Birincisi, Phytia bir sorgunun, yüklem, belirteç ,wh-kelimeleri gibi sorudan bağımsız bir gösterimini kullanmaktadır. İkincisi, Phythia, sorguları f-mantığa göre çevirmek için, sorudan bağımsız ontoloji temelli bir arayüz sunmaktadır. Ancak, Phytia ontoloji terimlerinin Lexinfo[2] vasıtasıyla manüel eşleştirmesini gerektirdiği için daha geniş alanların ölçeklendirmesini yapmamaktadır.

Cabrio vd. [4] QAKIS isimli ontoloji-ilişki eşleşmelerinde sezgisel soru-cevap temelli bir sistem sunmaktadırlar. İlişki eşleşmeleri, zengin içerik eşleşmesi sağlamak için Wikipedia'dan alınan yüzeysel formlara dayanmaktadır. Örneğin; X, Y şehrinde doğdudan; Y X'in doğum yeridir ilişkisi çıkarılabilmektedir. Ancak QAKIS sorgu başına bir ilişki bulmaktadır. Ayrıca doğal dillerin çeşitliliğini göz önüne almayan temel sezgilere dayanmaktadır.

3.3 Hibrid Soru Cevaplama

Hibrid soru cevaplama hem yapısal hem de yapısız verileri bir arada kullanan bir yaklaşımdır. Hibrid soru cevaplama son yıllarda ortaya çıkan bir yaklaşımdır. Bu nedenle literatürde sınırlı sayıda çalışma mevcuttur. Bu başlık altında bu alanla ilgili yapılmış olan geniş kapsamlı bir çalışma olan HAWK'dan bahsedilmektedir.

HAWK [32], yapısal (structured) ve yapısız (unstructured) verilerin her ikisini de kullanan bir soru cevaplama yaklaşımı sunmaktadır. İlk olarak soruyu parçalarına ayırmakta ve varlık tanımlama yapmaktadır. Sonrasında parçalarına ayırdığı bileşenlerin tüm elamanları için semantik eşleştirmelerini bulmaktadır. Bu bulunan eşleştirmeleri birleştirerek tüm olası kombinasyonları bulmaktadır ve değerli olan kombinasyonları ele almaktadır. Tüm olasılıkları denediği için HAWK çok yavaşdır (60sn'den fazla).

[2] http://lexinfo.net/

4. FreeQA: HİBRİD SORU CEVAPLAMA SİSTEMİ

Bağlı veri (linked data) üzerinde soru cevaplama temel olarak doğal dildeki soruları biçimsel sorgulara çevirmektedir, genellikle bu sorgular da yapısal veri kaynakları üzerinde çalıştırılabilen ve cevap alınabilen SPARQL sorgularıdır. Şekil 4.1 FreeQA'in soru cevaplama sırasında takip ettiği adımları göstermektedir.

FreeQA soru cevaplama yaklaşımı adım adım bu bölümde açıklanmaktadır. Yaklaşım anlatılırken alttaki hibrid soru, süreci daha iyi anlatabilmek için örnek olarak kullanılacaktır.

In which city was the assassin of Martin Luther King born?

4.1 Soru Analizi (Question Analysis)

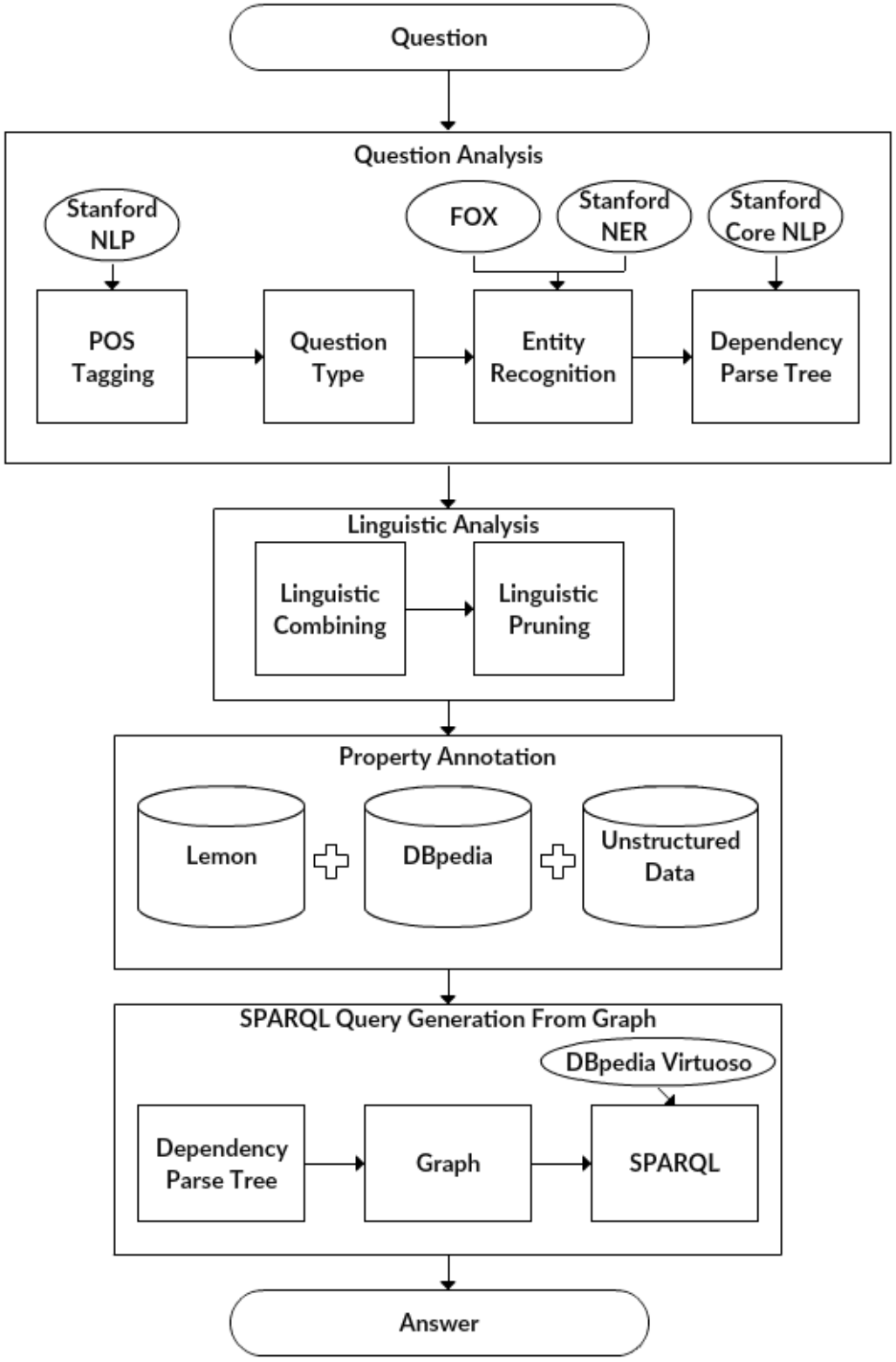
Doğal dildeki sorular çeşitli formatlarda gelebilmektedir ve bu formatlar doğal dil işlemeye uygun olmayabilmektedir. Bu yüzden biçimsel sorgulara dönüştürürken ön hazırlık olarak yapılması gerekli olan birkaç işlem vardır. Aşağıda bu işlemler ele alınacaktır.

4.1.1 Soruyu Parçalama ve Etiketleme (POS-Tagging)

Doğal dildeki bir soruyu anlamak sorunun parçalarını anlamsal olarak anlamayı gerektirir. Bunun için doğal dil işleme (natural language processing) zorunludur. Bu sürecin bir parçası sorunun kelimelerinin etiketlerini (tag) bulmaktır. Buna Part-of-Speech (POS) etiketleme denmektedir. ClearNLP [5] ve StanfordNLP [20] gibi bu işleri otomatik yapan araçlar vardır. ClearNLP'nin daha fazla bellek alanına (heap space) ihtiyaç duymasından ve StanfordNLP'nin özel varlıkları bulma (named entity recognition (NER)) gibi bir özelliği olmasından dolayı StanfordNLP tercih edilmektedir.

Çalışan örnek soru için StanfordNLP şu şekilde POS etiketlerini döndürmektedir:

"In/IN which/WDT city/NN was/VBD the/DT assassin/NN of/IN Martin/NNP Luther/NNP King/NNP born/VBN ?/"



Şekil 4.1: FreeQA doğal dildeki soruları cevaplamak için bu adımları takip eder.

IN edatlar için, WDT Wh-tipindeki soru kökleri için, NN tekil isimler için, VBD geçmiş zamandaki fiiller için, NNP özel isimler için, vb. Etiketlerin tüm listesine buradan[1] ulaşılabilir.

4.1.2 Soru Tipi Tanımlama (Question Type Determination)

İngilizcede doğal dildeki sorular belli kalıplardan oluşmaktadır ve bu kalıplar tiplerine göre sınıflandırılabilmektedir. Bu soru tipleri; evet/hayır (yes/no), kişi (agent veya person), yer (location), zaman, miktar (aggregation (count gibi)), liste (list), vb. Örneğin, çalışan örnek soru için soru tipi bir *yer sorusudur*. Soru tipleri POS etiketlerine göre tanımlanmaktadır.

Soru tipleri 3 farklı şekilde sınıflandırılabilir. Bunlar (1) eğer soru Wh soru kalıplarından (what/where/when/who) biriyle başlıyorsa, soruya bağlı olarak o bir şey (thing)/yer (location)/zaman (time)/kişi (agent) sorusudur; (2) eğer soru yardımcı fiille başlıyorsa o bir evet/hayır (yes/no) sorusudur; (3) eğer soru "How much", "How old", "How many" gibi soru kalıplarıyla başlıyorsa o bir miktar (aggregation) sorusudur.

4.1.3 Varlık Tanıma (Entity Recognition)

Soru cevaplamadaki önemli bir adım da sorudaki özel varlıkları (named entity) belirlemektir. Soru doğrudan bu varlıklarla ilişkilidir. Bu yüzden doğru varlıkları belirlemek ve bilgi veritabanında (knowledge base) bulunan doğru varlıklarla eşleştirmek kritiktir.

Kolayca erişilebilen ve açık kaynak olan varlık tanıma (entity recognition) araçları mevcuttur ([22], [21], [27], [8]) ve varlıkları tanımada oldukça başarılıdırlar. Bu araçlardan birkaçı bu başlık altında incelenmektedir. Bunlar; DBpedia Spotlight [21], FOX [27], ve TagMe [8]'dir. Hepsi varlık tanıma (entity recognition) ve varlıkları anlamsal olarak ayrıştırmada (disambiguation) farklı başarı oranlarına sahiptir. Örneğin, hepsi çalışan örnek soru için doğru varlığı döndermektedir yani *"Martin Luther King, Jr."*. Fakat diğer taraftan başka bir soru için örneğin *"Are the Rosetta Stone and the Gayer-Anderson cat exhibited in the same museum?"*, dönmesi gereken doğru varlıklar "Rosetta Stone" ve "Gayer-Anderson cat" olması gerekirken, bu araçlar alttaki gibi yanlış ve eksik sonuçlar döndermektedir:

[1] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

- DBpedia Spotlight: "Rosetta Stone" ve "cat" varlıkları döndermektedir.
- TagMe: "Rosetta Stone", "Gayer-Anderson cat" ve "museum" varlıkları döndermektedir.
- FOX: Rosetta Stone, "Anna Troberg" olarak tamemen bağımsız bir şey yakalamaktadır ve diğer bir varlık olarak "Gayer Anderson"'ı döndermektedir.

Yukarıdaki sonuçlardan da görüleceği üzere varolan varlık tanıma araçları yetersiz görünmektedir. Bu yüzden yeni bir varlık tanıma aracı oluşturulmalı ya da var olanlar iyileştirilmelidir. Sonuç olarak yukarıda bahsedilen araçlardan da yararlanılarak aşağıda açıklanan yeni bir yaklaşım sunulmaktadır.

4.1.3.1 FreeQA Varlık Tanıma (FreeQA Entity Recognition)

Algoritma 1'de, FreeQA'in sunmuş olduğu varlık tanıma (entity recognition) ve anlamsal ayrıştırıcı (disambiguation) fonksiyonları gösterilmektedir. Algoritma girdi olarak soruyu alıp, çıktı olarak da tanımlanan varlıkları döndermektedir.

Algoritma 1 varlık tanıma için hibrid bir yaklaşım sunmaktadır. Varlık tanıma için FOX ve StanfordNER[2] birlikte kullanılmaktadır. Algoritma ilk olarak, FOX (satır 1-2) ve Stanford NER (satır 3-4) ile ayrı ayrı varlıkları bulmaktadır. Sonra FOX ve Stanford NER'den gelen varlıkları karşılaştırmakta (satır 5-6) ve eğer onlar eşleşirse FOX'dan gelen varlığı doğru olarak kabul etmektedir, yoksa soru ve varlık özelliklerine göre (varlık tipi, üst-alt (parent-child) bağlılıkları) Stanford NER varlıklarını DBpedia varlıkları ile eşleştirmektedir (satır 15-26).

Varlık tanıma sonucunda gelen varlıklar semantik olarak anlamsal karmaşıklık içinde (ambiguity) olabilir. Bu yüzden bir sonraki adımda anlatılacak olan anlamsal ayrıştırıcıya (disambiguation) ihtiyaç duyulmaktadır.

4.1.3.2 Anlamsal Ayrıştırıcı (Disambiguation)

Anlamsal ayrıştırıcı (disambiguation), doğru olması mümkün olan birçok varlık arasından en doğru varlığı seçme işlemidir. Bu varlıklar aynı isim ve aynı etikete sahip olabilmektedir. Örneğin, "apple" bir şirket veya bir meyve olabilir, burada varlık anlamsal ayrıştırıcının işlevi, bunlardan hangisinin doğru olduğunu bulmaktır.

[2] http://nlp.stanford.edu/software/CRF-NER.shtml

Algoritma 1 Entity recognition

Input: Question (q) $q = \langle text, entity\ list(L_e) \rangle$

Output: Entity list (L_e) ▷ L_e is q attribute.

L_f = *extract q entities from FOX with q.text as input parameter.*
  $i = 0, ..., n\ f_i \in L_f,\ f_i = \langle label, uri\langle label, resource \rangle, type \rangle$
L_s = *find q entities with Stanford NER*
  $i = 0, ..., n\ s_i \in L_s,\ s_i = \langle label, type, parent, children \rangle$
for $i = 0, ..., n\ f_i \in L_f,\ s_i \in L_s$ **do**
  if $f_i.uri.label$ *is similar with* $s_i.label$ **then**
    $wikiPageDisambiguates = get\ f_i\ ambiguity\ links\ from\ DBpedia$
    if *wikiPageDisambiguates not empty* **then**
      $L_e.add(disambiguation(q, wikiPageDisambiguates))$
    else
      $L_e.add(f_i.uri)$
    end if
  else
    //d is a DBpedia entity
    L_d = *get all DBpedia entities which d.label contains* $s_i.label$ *andd.type* = $s_i.type$
    l is temporary list
    for $i = 0, ..., k$ *each* $d_k \in L_d$ **do**
      if $d_k.label = s_i.label$ *and* $d_k.wikiPageDisambiguates$ *not empty* **then**
        $l.add(d_k.wikiPageDisambiguates)$
      else if $d_k.label = s_i.label$ **then**
        $l.add(item)$
      else if $d_k.label$ *contains* $s_i.parent$ *or* $s_i.children$ **then**
        $l.add(item)$
      end if
    end for
    $L_e.add(disambiguation(q, l))$
  end if
end for
return L_e

Algoritma 2 Disambiguation

function disambiguation(*Question* (q), *DBpedia Entities* (L_e))
  L_s *is score list of* L_e
  for $i = 0, ..., n$ *each* $e_i \in L_e$ **do**
    $score = new\ Score$; score = $\langle weight,\ entity \rangle$
    $weight = 0$
    //weightOfRedirect is an assigned weight for wikiPageRedirectsOf and weightOfDisambiguate is an assigned weight for wikiPageDisambiguatesOf.
    $score.weight = \#e_i.wikiPageRedirectsOf\ links * weightOfRedirect$
      $+ \#e_i.wikiPageDisambiguatesOf\ links * weightOfDisambiguate$
      $+ e_i.label\ resemble\ ratio\ with\ q$
    $score.entity = e_i$
    $L_s.add(score)$
  end for
  return *element of* L_s *with max weight*
end function

Algoritma 1'de yeni bir anlamsal ayrıştırıcı yaklaşımı sunulmaktadır. Sunulan yaklaşımın diğerlerinden farkını göstermek için öncelikle diğerlerinin yaklaşımlarını gösterip ardından yeni yaklaşımdan bahsedilecektir. Varlık anlamsal ayrıştırıcı üzerine yapılmış birçok çalışma vardır. Bunların bazılarından aşağıda bahsedilmektedir.

Hakimov vd. [12] NERSO adlı sistemleriyle varlıkları anlamsal olarak ayrıştırmak (disambiguation) için merkezcil çizge (graph-centrality) temelli bir yaklaşım sunmaktadırlar. NERSO ilk olarak metinsel söylemleri (text mention) DBpedia varlıkları ile eşleştirmektedir ve sonrasında DBpedia'dan bu varlıkların arasındaki ilişkileri bulmaktadır. Bu ilişkilerden ve varlıklardan çizge oluşturmaktadır. Çizgenin düğümleri (vertex) varlıkları temsil etmektedir, kenarları (edge) ise ilişkileri temsil etmektedir. Son olarak bu çizgede merkezcil çizge (graph-centrality) yaklaşımına göre varlıklar arasında skorlama yapıp en iyi eşleşen varlığı bulmaktadır.

Moro vd. [23] Babelfly sistemini sunmaktadırlar. Babelfly anlamsal ayrıştırmayı üç adımda yapmaktadır. İlk adım, BabelNet[3] kullanarak girdi olarak verilen sorunun veya cümlenin dilsel analizini yapmaktadır. Sonrasında BabelNet'e göre tanımlanan tüm mümkün anlamları listelemektir. Son olarak çıkarılan anlamlar arasındaki ilişkilere göre çizge oluşturmaktır ve sonrasında en çok bağlantılı alt çizgenin (sub-graph) seçilmesini sağlamaktır.

[3] http://babelnet.org/

Hoffart vd. [14] AIDA'yı sunmaktadırlar. AIDA, DBpedia ve YAGO gibi bilgi veritabanlarından yararlanarak varlıkların anlamsal ayrıştırımını sağlayan bir araçtır. AIDA metinde bahsedilen varlıkların alt çizgesini (sub-graph) aç gözlü arama algoritması (greedy search) ile oluşturmaktadır. Aç gözlü arama algoritmasının ağırlıklandırması ise varlık bağlantı sayısına göre çalışmaktadır.

FreeQA'in sunmuş olduğu anlamsal ayrıştırma fonksiyonunda (Algoritma 1'deki anlamsal ayrıştırma fonksiyonu) kullanılan çizge yapısı geçmiş sistemlerle aynı fakat skorlama diğerlerine göre farklıdır. Skorlama takip eden parametrelere göre yapılmaktadır: varlık etiketi benzerliği, `wikiPageRedirectsOf` link sayısı, `wikiPageDisambiguatesOf` link sayısı. Bu skorlama parametrelerine göre en yüksek skoru alan varlık anlamsal ayrıştırmanın sonucunda dönen varlık olmaktadır (Algoritma 1 satır 14).

Çalışan örnek soru için anlamsal ayrıştırma fonksiyonu uygulandığında *Martin Luther King* üç farklı varlığa referans olabilmektedir. Bunlar: *Martin Luther King, Jr.* , *Martin Luther King III.*, veya *Martin Luther King, Sr.*. FreeQA ve diğer sistemler anlamsal ayrıştırma sonucunda doğru varlığı (*Martin Luther King, Jr.*) vermektedir.

Başka bir soru göz önünde bulundurulursa *"Of the people that died of radiation in Los Alamos, whose death was an accident?"*, bu soru için FreeQA ve diğer sistemler farklı sonuçlar vermektedir. Los Alamos varlığı tanımlanmaktadır ve anlamsal ayrıştırma adımları Algoritma 1'e göre takip eden adımlar şeklinde olmaktadır:

- FOX "Los Alamos"'u "dbpedia.org/resource/Los_Alamos" şeklinde bulmaktadır.
- Stanford NER soruda varlık olarak "Los Alamos"'u bulmaktadır.
- FOX'dan gelen varlık ile Stanford NER'den gelen varlık karşılaştırılmaktadır. İkisinden de "Los Alamos" geldiği için FOX'un çıkarımda bulunduğu varlık doğru olarak kabul edilmektedir.
- FOX'un çıkarımda bulunduğu varlığın anlamsal ayrıştırmaya ihtiyacının olup olmadığı kontrol edilmektedir. Bu FOX'dan gelen varlığın anlamsal ayrıştırma linklerinden anlaşılmaktadır yani kısacası FOX, DBpedia linki içeren bir varlık döndermektedir. Bu varlık da bazı özelliklere sahiptir. Bu özelliklerden birisi de "dbo:wikiPageDisambiguates"'dir. Bu özellik varlık sınıfının disambiguation linklerini göstermektedir. Los alamos "Los Alamos, New Mexico" ve "Los Alamos, California" olabilmektedir bu yüzden anlamsal ayrıştırma gerekmektedir.

- Anlamsal ayrıştırma için her iki varlık için skorlama yapılmaktadır. Skorlama "Los Alamos New Mexico" için şu şekilde yapılmaktadır:
  $\#wikiPageDisambiguatesOf\ link\ sayisi\ *\ c_D$
  $+\#wikiPageRedirectsOf\ link\ sayisi\ *\ c_R$
  $+\ resemble\_ratio$
  $= (1 * 0.1)\ +\ (5 * 0.1)\ +\ (0.47)\ = 1.57$
  (c_R ve c_D 0.1 olarak alındı.) "Los Alamos California" için şu şekilde yapılmaktadır:
  $\#wikiPageDisambiguatesOf\ link\ sayisi\ *\ c_D$
  $+\#wikiPageRedirectsOf\ link\ sayisi\ *\ c_R$
  $+\ resemble\_ratio$
  $= (1 * 0.1)\ +\ (2 * 0.1)\ +\ (0.47)\ = 0.77$
  (c_R ve c_D 0.1 olarak alındı.)

- En yüksek skora sahip olan varlık anlamsal ayrıştırma sonucunda dönen, ağırlığı 1,57 olan "dbpedia.org/resource/Los_Alamos,New_Mexico" olmaktadır.

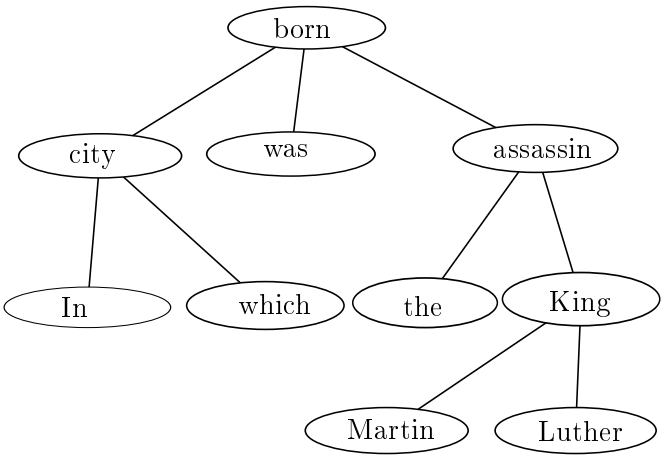
4.1.4 İlişki Ağacını Bulma (Dependency Parse Tree)

StanfordNLP[4] gibi NLP araçları sorunun ilişki ağacını (dependency parse tree) çıkarabilmektedirler. İlişki ağacı sorunun gramatik yapısını göstermektedir ve sorunun içinde geçen kelimelerin arasındaki ilişkileri göstermektedir. Bu ilişkilerin birbirleri ile kurduğu bağlılıkları ele almaktadır. Şekil 4.2 çalışan örnek soru için oluşan ilişki ağacını göstermektedir. İlişki ağacı, sorular için temel unsurdur. Sorgu taslağını bulmak ve SPARQL sorgusunu oluşturmak için kullanılmaktadır.

4.2 Dilsel Analiz (Linguistic Analysis)

FreeQA'deki önemli olan diğer bir adım ise dilsel analizdir. Yani bölüm 4.1.1'den elde edilen POS etiketleri ve bölüm 4.1.4'den elde edilen ilişki ağacındaki bağlılıkları (dependency) kullanarak ağacın (parse tree) analiz edilmesidir. Analiz sonucunda ağaçda (parse tree) bağlı olan kelime öbekleri birleştirilmekte ve anlamı olmayan kelimelerde ağaçtan atılmış olmaktadır. Burada doğal dil işleme teknikleri uygulanmaktadır.

[4] http://nlp.stanford.edu/



Şekil 4.2: İlişki Ağacı (Dependency Parse Tree)

Bizim çalışan örnek sorumuz için dilsel analiz sonucunda ilişki ağacı şu şekilde olmaktadır:

>born in (kelime öbeği)
    >city
    >assassin
        >Martin Luther King, Jr. (varlık)

Anlamsız kelimeler (Stop words), *the, of, which* gibi, ağaçtan çıkartılan kelimelerdir.

4.2.1 Dilsel Birleştirme (Linguistic Combining)

Bazı ifadeler (kelime öbekleri gibi) bütün olarak göz önünde bulundurulmalıdır. Örneğin: *Golden, Globe, ve awards* kelimeleri ayrı ayrı tekil olarak düşünülemez çünkü üçüde birbirine gramatik olarak bağlıdır ve doğru sonuca ulaşmak için üçü birlikte düşünülmelidir.

FreeQA dilsel birleşme (linguistic combining) için HAWK [32]'un sağlamış olduğu algoritmayı kullanmaktadır. HAWK'un yaptığı şey temel olarak POS etiketlerden yararlanarak ve yapısal dil taslakları kullanarak sorudaki kelime gruplarını bulmaya çalışmakdır. Bu taslaklar şu şekildedir: Eğer kelime gruplarının POS etiketleri (CD | JJ | NN(.)* | RB(.)*)'den biriyle başlarsa, (NN(.)* | RB | CD | CC | JJ | DT | IN | PRP | HYPH | VBN)'den biriyle devam ederse ve (NNP(S)?, (VB | WDT | IN))'den biriyle biterse, bu kelimelerin oluşturduğu grup bir kelime öbeğidir. (Daha fazla detay için HAWK Algoritma 1 incelenebilir.) Fakat HAWK'un sunmuş olduğu bu algoritma

tek başına yeterli değildir çünkü bu algoritma ile birbirine bağlı olmayan kelime grupları da söz öbeği gibi bulunabilmektedir. Bu yüzden bu algoritmaya ek olarak söz öbeğini oluşturan kelimelerin birbirleri ile ilişkili olup olmadığı kontrolü yapan bir mekanizmaya ihtiyaç vardır. Bu bağlantı da ilişki ağacındaki üst-alt (parent-child) ilişkisine göre kontrol edilmektedir.

4.2.2 Dilsel Ayrıştırma (Linguistic Pruning)

Soru anlamsız kelimeleri içerebilmektedir. Bu kelimelere "stop words" denilmektedir. Bu kelimeler cümle kurgusu bakımından önemli ama cümleden çıkarıldığı zaman bir anlam kaybına yol açmayan kelimelerdir. Bu kelimelerin cümledeki gramatik görevleri belli olduğu için POS etiketleri de bellidir. Bu kelimelerin POS etiketleri: DT (Belirteç), IN (Edat), POS (Sahiplik ekleri), PRP (Zamir), WDT (Wh-Soru ifadeleri), WP (Wh-zamirleri), WRB (Wh-zarfları). Bu kelimeleri incelemenin soru cevaplamaya bir etkisi olmayacağı için bu kelimelerin hepsi çıkarılmaktadır.

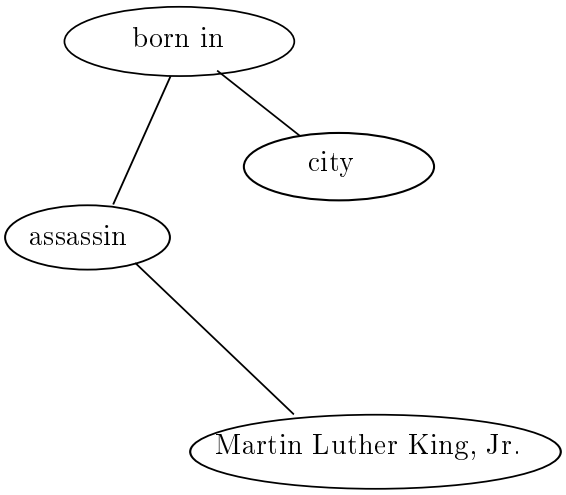
Çalışan örnek soru için *"the"*, *"of"* ve *"which"* anlamsız olan kelimelerdir. Bu nedenle, bu kelimelerin hepsi ilişki ağacından kaldırılmaktadır. Şekil 4.3 çalışan örnek soru için dilsel analiz sonucunda oluşan veya geriye kalan ilişki ağacını göstermektedir. "born" ve "in" birleşerek "born in" söz öbeğine dönüşmektedir çünkü her iki kelime de ilişki ağacında birbirine bağlı ve bölüm 4.2.1'de bahsedilen dilsel yapıya uygun taslaklardan birine sahiptir. Ayrıca "the", "of" ve "which" gibi anlamsız kelimeler (stop word) de ilişki ağacından kaldırılmıştır. Ek olarak "Martin Luther King" bölüm 4.1.3'de "Martin Luther King, Jr." varlığı olarak bulunmuştur, bundan sonraki bölümler için de bu şekilde devam edecektir.

4.3 Semantik İlişkilendirme (Semantic Annotation)

Semantik ilişkilendirme dilsel analiz sonucunda oluşan ağacın bilgi veritabanı (knowledge base) ile eşleştirilme sürecidir. Semantik olarak eşleştirme üç farklı süreçte olmaktadır.

Süreçlerden ilki, bazı kelimeler doğrudan bilgi veritabanı ile eşleşmeyebilir çünkü bazı kelimelerin yazılışları farklı anlamları aynı olabilmektedir veya anlamları ilişkili olabilmektedir. Örneğin; "married" kelimesi "spouse", "engaged", "wife", "husband"... gibi kelimelerle ifade edilebilmektedir, farklı yazılışta ama benzer

anlamlardadır. Bu benzerlikleri yakalamak için Lemon[5]'un hazırlamış olduğu *"kelime, rdf:type, semantik eşleniği"* üçlülerinin oluşturduğu sözlük kullanılmaktadır. Bu sayede ağaçdaki (tree) düğümlerin çoğunun semantik eşleniği bulunabilmektedir.



Şekil 4.3: Dilsel analiz sonucunda oluşan ilişki ağacı.

Süreçlerden ikincisi, bazı kelimeler doğrudan bilgi veritabanı ile eşleşebilmektedir yani örneğin "city" kelimesi için doğrudan bilgi veritabanında etiketi (label) "city" olan bir varlık (http://dbpedia.org/ontology/city) mevcuttur. Bu varlıkları yakalamak için bilgi veritabanı etiketi ilişki ağacı düğümünün etiketine eşit olan varlıkları getir şeklinde bir SPARQL sorgusu yazılmaktadır. Ayrıca bu varlıkları yakalamak için Lemon'un hazırlamış olduğu *"kelime, rdf:type, semantik eşleniği"* üçlülerinin oluşturduğu yago sözlüğü de kullanılmaktadır.

Süreçlerden sonuncusu ise bazı kelimelerin bilgi veritabanında bulunan hiçbir yapısal veri ile eşleniği olmayabilmektedir. Bu durumda kelimenin yalın hali semantik ilişki (annotation) olarak alınmaktadır. Örneğin, bir söz öbeği olan "stage name" için birinci ve ikinci süreçler uygulandığı zaman herhangi bir semantik ilişki elde edilememektedir çünkü semantik olarak herhangi bir eşleniği yoktur bu durumda "stage name" yalın haliyle semantik ilişki listesine eklenmektedir.

Diğer bir taraftan da süreçler içinde belirtilmeyen bir adım ise sorunun tipine göre (bölüm 4.1.2'de bulunan) ilişkidir. Burada sabit bir *"soru tipi, semantik eşleniği"* yapısında bir liste vardır. Örneğin soru tipi "person" ise semantik eşleniği "http://dbpedia.org/ontology/Agent" dir.

[5]http://lemon-model.net/

Semantik ilişki bulma işlemi yukarıda belirtilen süreçlere göre farklılık göstermektedir ama diğer taraftan bulunan semantik ilişkiler tiplerine göre de farklılık göstermektedir. Semantik ilişki tipleri dört farklı şekilde olabilmektedir. Bu tipler varlık (bilgi veritabanı kaynağı), sınıf (?x a City gibi kaynakların sınıflandırılması), özellik (veri veya nesneler arasındaki ilişkiyi sağlayan data property veya object property) ve serbest metin (kelimenin yalın hali) olabilmektedir. Semantik ilişkilerin tip bilgisine kelimelerin semantik eşleniğinin "rdf:type" bilgisinden ulaşılmaktadır. Çalışan örnek soruda, "city" kelimesi için semantik eşlemesi "http://dbpedia.org/ontology/City" olan semantik ilişkinin "rdf:type"'ı "owl:Class"'dır, bu yüzden "City" sınıf (class) olarak sınıflandırılmaktadır. Düğüm "born in" için semantik eşleniği "http://dbpedia.org/ontology/birthPlace" olan semantik ilişkinin "rdf:type"'ı "rdf:Property"'dir, bu yüzden "birthPlace" özellik (property) olarak sınıflandırılmaktadır. Tipi varlık (entity) olan semantik ilişkiler için bir sınıflandırmaya gerek yoktur çünkü onlar daha öncesinde varlık olarak tanımlanmaktadır.

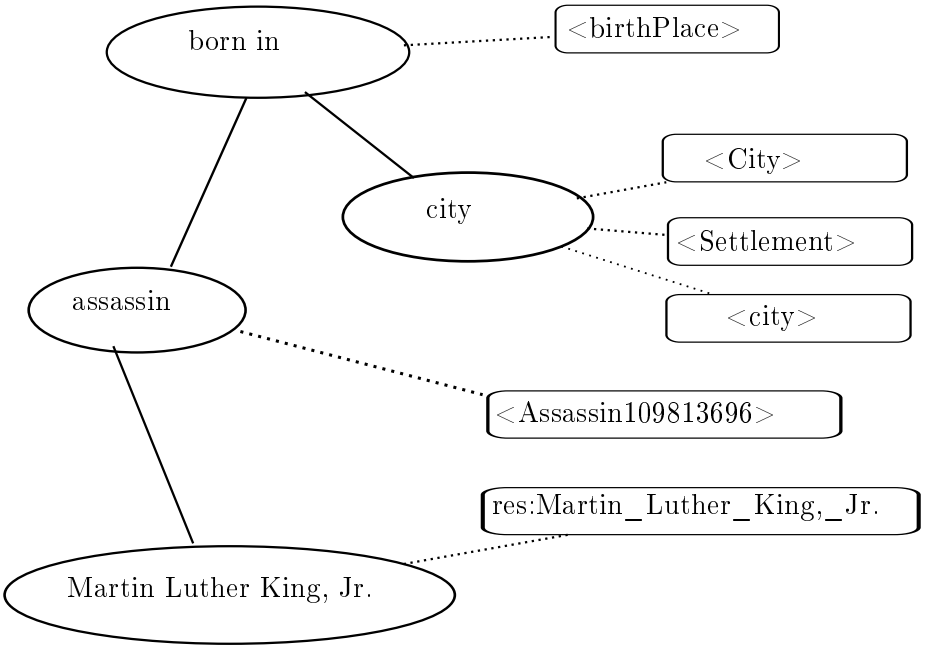
Algoritma 3'de, ilişki ağacında (dependency tree) bulunan tüm düğümler için semantik ilişkiler bulunmaktadır. Girdi olarak bölüm 4.1.4'de bulunan ilişki ağacı ve soru tipi kullanılmaktadır. Çıktı olarak da ilişki ağacı, düğümlerin semantik ilişkileri ile birlikte geri alınmaktadır. İlişki ağacının her bir düğümü etiket ve semantik ilişki listesinden oluşmaktadır.

Çalışan örnek soru için semantik ilişki sonucu Şekil 4.4'deki gibi olmaktadır. Diktörtgen kutucuklar düğümün semantik ilişkilerini göstermektedir. Her düğüm bir veya birden fazla semantik ilişkiye sahip olabilmektedir. Her düğüm için en az bir tane semantik ilişki vardır çünkü düğüm etiketinin yalın hali yani serbest metin (free-text) bir semantik ilişkidir. Bu semantik ilişkiler; sınıf (class), varlık (entity), özellik (property) veya serbest metin (free-text) olabilmektedir bu yüzden bunlar Tablo 4.1'de detaylı olarak sınıflandırılmaktadır. "Martin Luther King" düğümü için, daha öncesinde bölüm 4.1.3'de varlık olarak sınıflandırıldığı için semantik ilişki sürecinden geçirilmemektedir çünkü varlık bir semantik ilişki tipidir.

4.4 Çizgeden SPARQL Sorgusu Üretme (SPARQL Query Generation from Graph)

İlişki ağacı (dependency tree) düğümlerinden bir çizge (graph) oluşturulmaktadır. Çizge oluştururken temel alınan şey, iki düğüm arasında semantik ilişki (annotation)

bileşimi olup olmadığıdır yani diğer bir deyişle ilişki ağacındaki düğümler arasında olan bağlılık (dependency) ilişkilerin düğümlerin oluşturduğu semantik ilişkilere dönüşmesine dayanmaktadır. Bu ilişkileri bulmak için birbirine bağlı olan düğümlerin semantik ilişkileri arasında olabilecek tüm kombinasyonlar denenmektedir ve oluşan ilişkilerin değerli olup olmadığına bakılmaktadır. Eğer kombinasyon değerli ise iki düğüm arasına ilişki (edge) olarak eklenmektedir değilse kombinasyon kaldırılmaktadır. Semantik ilişkiler arasında kombinasyon oluştururken taslaklardan yararlanılmaktadır. Bu taslaklara daha sonra değinilecektir.



Şekil 4.4: Semantik İlişkilendirme

Tablo 4.1: Düğümlerin Semantik İlişki Sınıflandırması

Node	Free-Text	Property	Entity	Class
born in	born in	birthPlace	-	-
assassin	assassin	-	-	Assassin109813696
city	city	city	-	City, Settlement
Martin	Martin		Martin	
Luther	Luther	-	Luther	-
King	King		King, Jr.	

Çalışan örnek soru için düğümlerin semantik ilişkilerinin tiplerine göre sınıflandırılmış hali Tablo 4.1'de verildiği gibidir.

Algoritma 3 Semantik İlişki Bulma Algoritması

Input: Question(q); $q = \langle type, tree(T) \rangle$

Output: Annotated tree(T); T=$\langle root \rangle$;

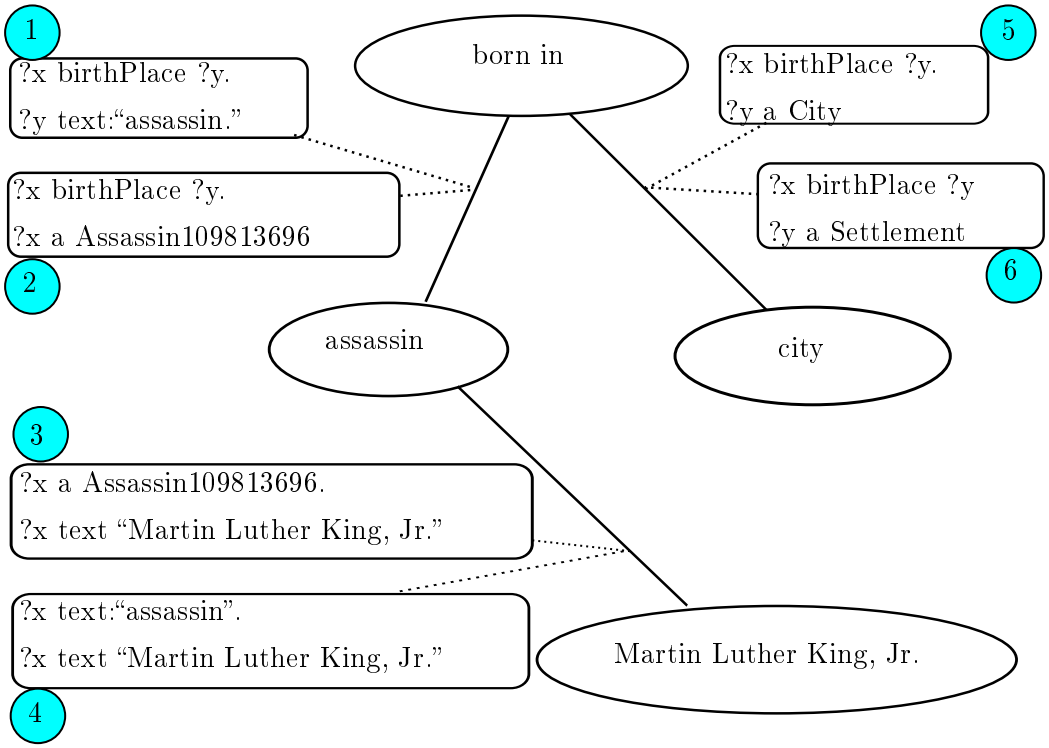
```
root = ⟨label, children, annotation list(L_a)⟩          ▷ T is q attribute.
Stack stack = q.tree.root
if root.label match with q.type then                    ▷ Who->Agent, Where->Location
    root.L_a.add(Property of q.type)
end if
while stack is not empty do
    item = stack.pop
    //Detect DBpedia properties that are related or match with label of item.
    list = get DBpedia and lemon resources(item.label)
    item.L_a.addAll(list)
    for eachchild ∈ item.children do
        stack.push(child)
    end for
end while
```

Çizge G = (V,E) şeklinde gösterilirse, V düğümleri temsil etmektedir, E'de bu düğümler arasındaki semantik ilişkilerin kombinasyonları sonucunda oluşan ilişkileri göstermektedir. Her ilişki $(e_0, ..., e_n)$ ilişki ağacı düğüm çiftlerinden $(\overrightarrow{v_i v_j})$ oluşmaktadır. Düğümlerin özellikleri:
i=0,...,n $\forall v_i = \langle$*entity(*Nv_i*), free-text(*Fv_i*), list of property(*$Pv_i = p_0, ..., p_n$*), list of class(*$Cv_i = c_0, ..., c_n$*)*$\rangle$. Düğümler arasındaki ilişkiler de düğümlerin semantik ilişkilerinin kombinasyonlarının oluşturduğu SPARQL sorgu parçacıklarından oluşmaktadır. Bu SPARQL sorgu parçacıkları farklı yapılarda ve formlarda olabilmektedir. Bu formlar oluşturulurken taslaklardan yararlanılmaktadır. Bunlar:

- ?x p_i ?y . ?y rdf:type c_j . p_i is in Pv_i and c_j in Cv_j
- ?x p_i ?y . ?y p_j ?z . p_i is in Pv_i and p_j in Pv_j
- ?x p_i Nv_j . p_i is in Pv_i
- ?x p_i ?y . ?y text:"Fv_j" . p_i is in Pv_i
- ?x ?r Nv_i . ?x text:"Fv_j"
- ?x rdf:type c_i . ?x text:"Fv_j" c_j is in Cv_j
- ?x text:"Fv_i" . ?x text:"Fv_j"

Örnek soru için oluşan çizge Şekil 4.5'deki gibi olmaktadır. Bu çizge yukarıda belirtilmiş olan taslakları kullanarak, düğümlerin semantik ilişkilerinin kombinasyonları sonucunda oluşturulmaktadır. Oluşan kombinasyonların doğruluğu "ASK" SPARQL sorgusuyla

kontrol edilmektedir. "ASK" sonucu true olanlar çizgeye eklenmektedir. Bu nedenle çizgede yer alan kombinasyonların 1, 2... 6 hepsinin ASK sonucu true'dur veya diğer bir deyişle çizgede bulunan tüm kombinasyonlar anlamlıdır.



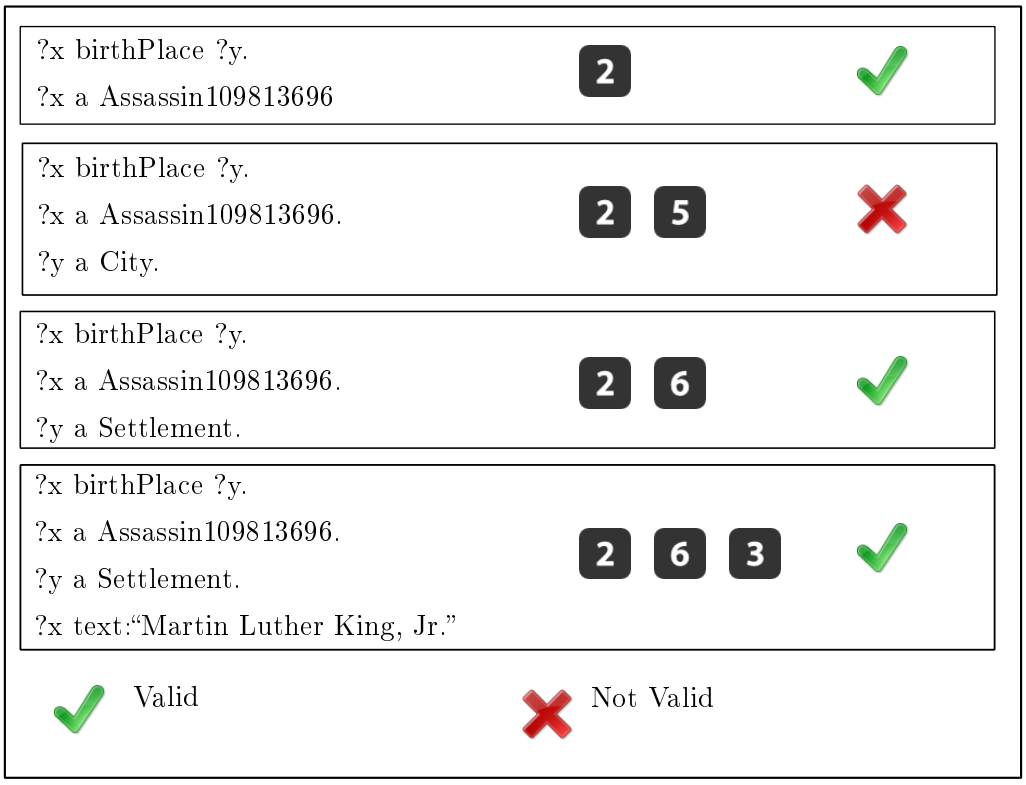
Şekil 4.5: Çizge

Çizgeyi oluşturduktan sonra, aç gözlü arama algoritması (Greedy Search) ile düğümler arasındaki SPARQL sorgu parçacıklarını birleştirilerek nihai cevabı verecek olan SPARQL sorgusu oluşturulmaktadır. Aç gözlü aramayı çizge üzerinde uygulayabilmek için semantik ilişki tiplerine göre ağırlık verilmektedir. Çünkü düğümler arasındaki sorgu parçacıkları bu semantik ilişkilerin birleşmesinden oluşmaktadır ve haliyle bu sorgu parçacıklarının bir ağırlığı olmaktadır. Semantik ilişkilere atanan ağırlıkların büyükten küçüğe doğru sıralaması şu şekildedir: varlık (entity) >özellik (property) >sınıf (class) >serbest metin (free-text).

Algoritma 4'de çizgeye aç gözlü arama algoritması uygulayarak SPARQL sorgusu elde etme adımları gösterilmektedir. Girdi olarak soru ve çizge kullanılmaktadır. Sonuç olarak da çalışan bir SPARQL sorgusu elde edilemektedir.

Çalışan örnek soru için algoritma uygulanırsa, adımlar Şekil 4.6'deki gibi olmaktadır. Şekil 4.6, önceki Şekil 4.5'de gösterilen sorgu parçacıklarına atıfta bulunmaktadır. Bu yüzden Şekil 4.6'de siyah renkte gösterilen sayılar Şekil 4.5'de gösterilen mavi renkteki

sayılara referans vermektedir. Şekil 4.6'de gösterilen doğru/yanlış sembollerinin anlamı ise sembollerin sol tarafında verilen sorguların ASK sonuçlarının doğru olup olmadığıdır.



Şekil 4.6: Algoritmanın Örnek Üzerinde Uygulanışı

Şekil 4.6 sonucunda nihai olarak şu şekilde bir SPARQL sorgusu elde edilmektedir:

```
SELECT distinct ?y WHERE {
  ?x birthPlace ?y.
  ?x a Assassin109813696.
  ?y a Settlement.
  ?x text:"Martin Luther King, Jr."
}
```

Yukarıda belirtilen SPARQL sorgusu çalıştırıldığı zaman, alınan cevap:

```
http://dbpedia.org/resource/Alton,_Illinois
```

Algoritma 4 Çizgeden SPARQL Sorgusu Üretme

Input: Question (q), graph (G)

Output: SparqlQuery

PriorityQueue queue = new PriorityQueue; Priority queue is comparable list and when remove element from queue, most heaviest element will return. Queue elements are comparable according to their weights that consist of entity = 1.3, property = 1.2, class = 1.1 and free-text = 1.0

SparqlQuery sq = new SparqlQuery

sq.sourceNode = q.tree.root

queue.add(sq)

while *queue is not empty* **do**

  item = queue.remove; Item will be the most heavy element of queue.

  if *item.children.size = q.tree.nodes.size* **then**

    return item

  end if

  //Get all outgoing edges(triples) of item from G.

  edges = G(item.sourceNode).edges

  for *each e in edges* **do** ▷ $e = \langle v_s, label, v_t \rangle$

    item = new SparqlQuery

    query = concatenate(item.query , e.label)

    if *query is valid* **then** ▷ query sparql result is not null

      if *item.children not contain* $e.v_s$ **then**

        $item.children.add(e.v_s)$

      end if

      $item.children.add(e.v_t)$

      $item.query = query$

      $item.sourceNode = e.v_s$

      $item.weight = item.weight + e.weight$; Each edge has a weight.

      $queue.add(item)$

    end if

  end for

end while

FreeQA ile test edilen sorulara ve bu soruların sonucunda dönen sorgu ve cevaplara projenin sitesinden[6] erişilebilmektedir. Sorgularda karşılaşılabilecek farklı terimlerden bazıları "contains" ve "bif:contains" dir. Bu terimler "text:" ile benzer işleri görmektedirler. "text:" ve "contains" ile tüm alanlar içinde arama (full-text-search) yapılabilmektedir ama "bif:contains" ile sadece metinsel kısımlarda (label, abstract gibi) arama yapılabilmektedir.

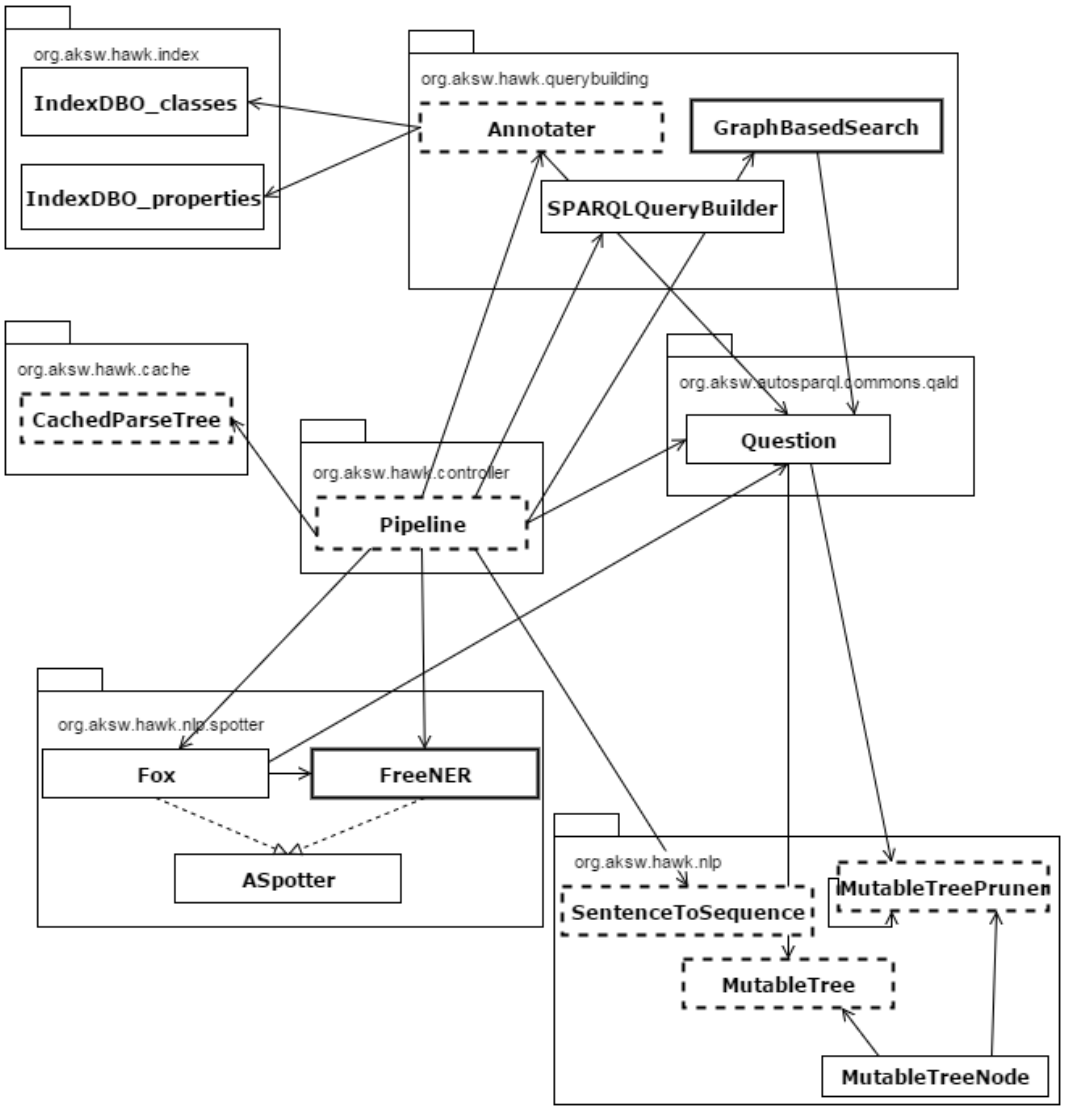
4.5 Yazılım

FreeQA, Java programlama dili ile yazılmaktadır ve yazılım olarak HAWK programının kullanmış olduğu altyapıdan yararlanılmaktadır. Bu altyapının da sınıflarından bazıları aynen kullanılmakta, bazıları ise değiştirilerek kullanılmaktadır. Yukarıda Şekil 4.7'de sınıflar 3 farklı şekilde gösterilmektedir. Bunlar (1) ince çizgi ile gösterilenler: HAWK'dan aynen alınan, üstünde değişiklik yapılmadan kullanılan sınıflar, (2) kesikli kalın çizgi ile gösterilenler: HAWK'dan alınıp üzerinde değişiklik yapılarak kullanılan sınıflar, (3) düz kalın çizgi ile gösterilenler: FreeQA tarafından eklenen sınıflardır. Sınıfların yaptıkları görevlere göre işlevleri aşağıda belirtildiği şekildedir.

IndexDBO_classes: Lemon tarafından sağlanmış olan yago sınıf üçlü sözlüğünde arama yapmaktadır. Bu arama sonucunda semantik ilişkiler yakalanmaktadır.
IndexDBO_properties: Lemon tarafından sağlanmış olan özellik sözlüğünde arama yapmaktadır. Bu arama sonucunda semantik ilişkiler yakalanmaktadır.
CachedParseTree: StanfordNLP kütüphanesi kullanılarak girdi olarak alınan soru bileşenlerine ayrılmaktadır ve bu bileşenlerin birbirleri ile ilişkileri göz önünde bulundurularak ağaç (tree) haline getirilmektedir. Bu ağaçta ön bellekte saklanmaktadır.
Fox: Girdi olarak soru alınarak Fox varlık tanıma aracı uygulanmaktadır.
FreeNER: FreeQA tarafından sunulan varlık tanıma ile ilgili metodları içermektedir.
ASpotter: Fox ve FreeNER'in ortak kullandığı metodları (post, get) içeren arayüzdür.
Pipeline: Tüm süreçleri baştan sona içeren ana (main) sınıftır.
Annotater: Tüm semantik eşleme süreçlerini içeren sınıftır.
GraphBasedSearch: Çizge temelli aramanın yapıldığı ve SPARQL sorgusunun oluşturulduğu sınıftır.
SPARQLQueryBuilder: Oluşturulan SPARQL sorgusunun çalıştırıldığı ve cevabının alındığı sınıftır.

[6] http://wis.etu.edu.tr/freeqa

Question: Doğal dildeki soruyu ve bu sorunun özelliklerini içeren varlık sınıfıdır.
SentenceToSequence: Dilsel birleştime işlemlerinin (Linguistic Combining) uygulandığı sınıftır.
MutableTree: İlişkinin ağacının (dependency tree) bulunduğu sınıftır.
MutableTreeNode: İlişki ağacının her bir düğümünü ve bu düğümlerin özelliklerini içeren sınıftır.
MutableTreePruner: Dilsel ayrıştırma işlemlerinin (Linguistic Pruning) uygulandığı sınıftır.



Şekil 4.7: FreeQA UML Diagramı

HAWK'un 0.1.0 versiyonundan yararlanılmıştır. FreeQA'in kullanmış olduğu tüm kodlara wis.etu.edu.tr/freeqa adresinden ulaşılabilir.

5. TEST VE DEĞERLENDİRME

Soru cevaplama sistemlerini değerlendirmek için çeşitli teknikler kullanılmaktadır. Bunlardan en sık kullanılanı, sorudan oluşturulan SPARQL sorusundan elde edilen cevapların doğru ve yanlış olarak sınıflandırılıp, doğruluk oranlarının recall, precision ve f-measure metotları ile gösterilmesidir. Bölüm 5.1'de kullanılan metotlarla ilgili detaylı bilgi verilecektir.

Soru cevaplamada testler belli veri kümeleri üzerinden yapılmaktadır. Bunların kullanılmasının temel nedeni bu veri kümelerinin birçok uygulama tarafından da değerlendirmede kullanılmasıdır. Bu sayede diğer sistemler ile kolay bir şekilde sistem karşılaştırılabilir.

Bu bölümde test ve değerlendirme için yapılan çalışmalar altı başlık altında incelenecektir. (1) Değerlendirme kriterleri ve test ortamı; sistemin değerlendirilirken kullanılan metrikler ve test ortamı için kullanılan bilgisayarın özellikleri anlatılılacaktır. (2) Test veri kümeleri; sistemi değerlendirmek ve diğer sistemler ile karşılaştırmak için kullanılan veri kümesi ele alınacaktır. (3) Hibrid sorular yapılan testler ve sonuçları sunulacaktır. (4) FreeQA'in varlık tanımadaki başarısı değerlendirelecektir. (5) FreeQA'in diğer sistemler ile karşılşatırması yapılacaktır. (6) Son olarak da FreeQA'de hatalı yapılan soruların değerlendirmesi yapılacaktır.

5.1 Değerlendirme Kriterleri ve Test Ortamı

Oluşturulan sorguların çalıştırılması için DBPedia virtuoso[1] kullanılmaktadır ve sorguların cevabı bu adrese istek gönderilerek online olarak alınmaktadır. Cevapların doğruluğu recall, precison ve f-measure metotları ile kontrol edilmektedir.

Öngörülen doğru cevaplar: Sorunun olması gereken sorgudan gelen cevap sayısı.
Üretilen sorgudan gelen doğru cevaplar: Sistemin sorgu için ürettiği sorgudan gelen, öngörülen doğru cevaplarla eşleşen cevap sayısı.
Üretilen sorgudan gelen tüm cevaplar: Sistemin sorgu için ürettiği tüm cevaplar.

$$\textbf{Recall} = \left(\frac{\mid \textit{Üretilen sorgudan gelen doğru cevaplar} \mid}{\mid \textit{Öngörülen doğru cevaplar} \mid} \right) \tag{5.1}$$

$$\textbf{Precision} = \left(\frac{\mid \textit{Üretilen sorgudan gelen doğru cevaplar} \mid}{\mid \textit{Üretilen sorgudan gelen tüm cevaplar} \mid} \right) \tag{5.2}$$

[1] http://dbpedia.org/sparql

$$\textbf{F-measure} = \left(\frac{|\ 2*precision*recall\ |}{|\ precision+recall\ |} \right) \tag{5.3}$$

FreeQA'in test edilmesi için kullanılan bilgisayarın özellikleri şunlardır:

- Windows 7 işletim sistemi
- 8 GB bellek
- Intel Core i5 işlemci
- 2,60 GHz işlemci hızı
- 2 GB ekran kartı

5.2 Test Veri Kümeleri

FreeQA, QALD (Question Answering Linked Data)'nin sağlamış olduğu veri kümeleri ile test edilmiştir. QALD, son beş yıldır düzenli olarak yapılan bir yarışmadır. Soru cevaplama üzerine sorular hazırlayarak, soru cevaplama üzerine yapılmış çalışmaları değerlendirmektedir. Bu değerlendirmeleri üç farklı kategoride yapmaktadır. Bunlar; çoklu dilde soru cevaplama (multilingual question answering), farklı bilgi veritabanlarını kullanarak soru cevaplama (question answering on interlinked data) ve hibrid soru cevaplama (hybrid question answering)'dır.

QALD, toplamda beş veri kümesi yayınlamıştır ancak bunlardan son ikisi, FreeQA ile ilgilidir. Bu veri kümeleri QALD-4[2] ve QALD-5[3]'dir. Bu veri kümelerinin içerdiği soru sayıları Tablo 5.1'de gösterildiği gibidir.

Tablo 5.1: QALD Veri Kümeleri

Veri Kümesi	**Çoklu Dilde Soru Adedi**	**Farklı Bilgi Veritabanlarını Kullanan Soru Adedi**	**Hibrid Soru Adedi**	**Toplam Soru Adedi**
QALD-4	89	25	25	139
QALD-5	150	150	40	340

[2]http://greententacle.techfak.uni-bielefeld.de/ cunger/qald/4/data/qald-4_hybrid_train.xml
[3]http://greententacle.techfak.uni-bielefeld.de/ cunger/qald/5/data/qald-5_train.xml

QALD-4 hibrid sorularından bazıları (S3, S4, S9) güncel DBpedia versiyonuna göre atıl kaldığından dolayı bu soruların bilgileri güncellenmiştir. FreeQA değerlendirme sonuçlarıyla birlikte bu güncellenmiş verilere projenin sitesinden[4] ulaşılabilir.

5.3 Hibrid Soru Cevaplama Testleri ve Sonuçları

FreeQA, QALD-4 [31] ve QALD-5 veri kümelerinin sağlamış olduğu hibrid sorularla test edilmektedir. Bu veri kümeleri QALD-4 toplam 25 sorudan, QALD-5 toplam 40 sorudan oluşmaktadır ama QALD-5 sorularında FreeQA'in ilgilendiği QALD-4'den farklı olarak 16 soru bulunmaktadır.

Öncelikle sistem QALD-4'ün 25 sorusu ile test edilmektedir ve bu sorulardan 24'üne %93 precision, %93 recall ve %92 F-measure ile cevap verilmektedir. Sonrasında sistem QALD-5'in QALD-4'den farklı olan 16 sorusu (normalde 40 hibrid sorusu var ama bunlardan 20'si QALD-4 soruları ile aynı ve 4 tanesi için ise kullanılan sistemler yeterli değil) ile test edilmektedir ve bu sorulardan 13'üne %81 recall, %74 precision ve %76 F-measure ile cevap verilmektedir.

Tablo 5.2: FreeQA Test Sonuçları

Veri Kümesi	**Toplam Soru**	**Doğru**	**Kısmen Doğru**	**Yanlış**	**Precision**	**Recall**	**F-measure**
QALD-4	25	22	2	1	%93	%93	%92
QALD-5	40	28	4	8	%81	%74	%76

5.4 Varlık Tanıma Sonuçları

Soruların sorguları semantik ilişkilendirmelerin (varlık, özellik, sınıf and serbest metin) birinin veya daha fazlasının bir araya gelmesi ile oluşmaktadır. Bu yüzden semantik ilişkileri (annotation) doğru bir şekilde bulmak önemlidir. Semantik ilişkileri bulurken varlık ve diğerleri ayrı şekillerde bulunduğu için ayrı şekilde değerlendirilecek olursa: veri kümelerinde bulunan toplam 41 farklı soru 30 adet varlık (entity) içermektedir,

[4] http://wis.etu.edu.tr/freeqa

FreeQA bunların hepsini bulmaktadır bu yüzden hiçbir soru varlık tanımlamadan kaçırılmamaktadır. Diğer taraftan semantik ilişkileri yakalamak için birçok farklı metot kullanılmaktadır. Bu metotlar lemon kütüphanesi kullanarak yakalama, yago sınıf kütüphanesi kullanarak yakalama ve DBpedia ile doğrudan eşleştirme. Bu metotların hepsi kullanışlıdır ama bazen bazı sorular için fazladan kısıtlama getirebilmektedir ve bu yüzden de sonuçların eksik alınmasına veya fazla alınmasına sebep olabilmektedir. Veri kümelerinde bulunan 41 sorudan 41'i de semantik ilişki veya semantik ilişki listesini içermektedir. FreeQA bu semantik ilişkilerden hepsini bulmaktadır ama SPARQL üretimi sırasında aç gözlü arama algoritması ile bu semantik ilişkilerden ikisini fazladan seçmektedir yani kısıt olarak sorguya fazladan semantik ilişki eklemektedir.

Tablo 5.3: FreeQA ve HAWK Varlık Tanıma Sonuçları

Sistem	**Veri Kümesi**	**Toplam Soru**	**Varlık İçeren Soru**	**Bulunamayan**
FreeQA	**QALD-4**	25	20	0
FreeQA	**QALD-5**	40	30	0
HAWK	**QALD-4**	17	14	3
HAWK	**QALD-5**	26	20	4

5.5 Karşılaştırma

Hibrid soru cevaplama ile ilgili bugüne kadar yapılmış birkaç çalışma bulunmaktadır ve bunlardan bir tanesi HAWK'dur. Aşağıda Tablo 5.4'de, FreeQA ile HAWK'un aynı veri kümeleri üzerindeki performansları incelenmektedir. HAWK, QALD-4 veri kümesinde bulunan 25 soru içinde 17 soru ile ilgilenmektedir. 25 sorudan 17 soru ile ilgilenmesinin nedeni; (1) HAWK sadece DBpedia bilgi veritabanına ihtiyaç duyan soruları çözebilmektedir. (2) HAWK, sayısal soru tiplerine cevap verememektedir. QALD-4, 4 adet yago bilgi veritabanı kullanan soru içerdiği için ve 4 adet de sayısal soru içerdiği için HAWK toplamda bu 8 soruyla ilgilenmemektedir. Geriye kalan 17 soru için HAWK, 0,72 F-measure ile soruları doğru cevaplamaktadır. Ek olarak QALD-5'de bulunan 40 sorudan 26 soru ile ilgilenmektedir ve bu soruları 0,30 F-measure ile doğru cevaplamaktadır. Diğer taraftan FreeQA QALD-4 veri kümesinde bulunan 25

sorunun hepsi ile ilgilenmektedir ve 0,92 F-measure ile doğru cevaplamaktadır. Ek olarak QALD-5'de bulunan 40 sorunun hepsi ile ilgilenmektedir ve 0,81 F-measure ile doğru cevaplamaktadır (Yukarıda ele alınan kısımda QALD-5 in tekil soru sayısı 16'dır, diğer soruları QALD-4 ile aynıdır).

HAWK ve FreeQA'in arasında kullandıkları araçlar ve uyguladıkları yöntemler bakımından da farklılıklar vardır. Bunlar: (1) HAWK soruyu parçalama ve etiketleme (POS) için ClearNLP kullanırken FreeQA StanfordNLP kullanmaktadır. (2) HAWK varlık tanıma için FOX, Wikipedia Miner, Spotlight, TagMe veya bunların birleşiminden yararlanırken FreeQA, FOX ve StanfordNER'in birleşiminden yararlanmaktadır. (3) HAWK SPARQL sorgu üretiminde semantik ilişkilerin tüm kombinayonlarını denerken FreeQA aç gözlü arama algoritması ile çalışıp sadece belli ağırlıkta olan semantik ilişkilerin kombinasyonlarını denemektedir. (4) HAWK SPARQL sorgusunu çalıştırmak için Jena Fuseki kullanırken FreeQA, DBpedia virtuoso kullanmaktadır.

Tablo 5.4: FreeQA ve HAWK karşılaştırması

Sistem	**Veri Kümesi**	**Toplam Soru**	**Test Edilen Soru**	**Doğru**	**Kısmen Doğru**	**Yanlış**	**F-measure**
FreeQA	QALD-4	25	25	22	2	1	%92
HAWK	QALD-4	25	17	9	6	2	%72
FreeQA	QALD-5	40	40	28	4	8	%81
HAWK	QALD-5	40	26	7	11	8	%30

5.6 Değerlendirme

Tablo 5.5'de ve 5.6'de FreeQA hatalarını ve bu hataların nedenlerini gösterilmektedir. Bu hatalardan yanlış yapılanlar sistemin bazı sorular için hatalı çalıştığını göstermemektedir, sistemin altyapı olarak yetersiz olduğunu göstermektedir. QALD-4 ve 5'de yanlış yapılan soruların hepsi serbest metni kullanmaya bağlı değil, serbest metinden bilgi çıkarımına bağlıdır. Serbest metinden özel bilgi çıkarımı da Jena Fuseki gibi araçlarla yapılmaktadır. FreeQA, DBpedia virtuoso kullandığı için ve DBpedia virtuoso'nun bilgi çıkarım özelliği olmadığı için bu sorular kaçırılmaktadır.

Tablo 5.5: FreeQA'ın QALD-4 veri kümesinde cevaplayamadığı veya kısmen cevapladığı soruların analizi

Soru	**Öngörülen Cevap**	**FreeQA Cevabı**	**Hatanın Nedeni**
Q7	SELECT ?n WHERE{ res:Steve_Jobs dbo:relative ?uri. ?uri text:"did not meet until she was" ?n}	SELECT ?n WHERE{ res:Steve_Jobs dbo:relative ?uri. ?uri text:"meet". ?uri dbo:birthDate ?n}	DBpedia virtuoso kullanıldığı için cevap bulunamıyor
Q8	SELECT ?uri WHERE{ res:Wu-Tang_Clan dbo:bandMember ?uri. ?uri text:"stage name" ?x. ?x text:"from" text:"movie". }	SELECT ?var WHERE{ res:Wu-Tang_Clan dbo:bandMember ?var. res:Wu-Tang_Clan ?r ?c. ?c text:"film". ?var text:"stage name". }	Sistemde "from" edat olarak yakalandığı için ağaçtan kaldırılmaktadır.
Q17	SELECT ?uri WHERE{ ?uri dbo:architect res:Shreve ,_Lamb _&_Harmon. { ?uri dbo:architecturalStyle res:Art_Deco. } UNION { ?uri text:"style text:"art deco". } }	SELECT ?uri WHERE{ ?uri dbo:architect res:Shreve ,_Lamb _&_Harmon. ?uri text:"style text:"art deco". ?uri a yago:Building.}	"Building" fazladan ilişkilendirilmiştir. yago:Building fazladan bir semantik ilişkidir.

Tablo 5.6: FreeQA'ın QALD-5 veri kümesinde cevaplayamadığı veya kısmen cevapladığı soruların analizi

Soru	**Öngörülen Cevap**	**FreeQA Cevabı**	**Hatanın Nedeni**
Q6, Q10, Q16	-	-	DBpedia virtuoso kullanıldığı için cevap bulunamıyor
Q7	SELECT ?uri WHERE{ ?x text:"is" text:"first person to climb all fourteen eight-thousanders". ?x dbo:birthPlace ?uri. ?uri rdf:type dbo:Settlement. }	SELECT ?cmd WHERE{ ?var dbo:birthPlace ?cmd. ?var text:'"fourteen eight-thousanders" text:"climb" AND "first"'. ?var a dbo:Agent. ?cmd a dbo:Settlement}	Kelime öbeği doğru olarak yakalanamadı.
Q17	SELECT ?uri WHERE{ ?x rdfs:label ?l. FILTER regex(?l,"Around the World in 80 Days","i") ?x dbo:director res:Buzz_Kulik. ?x dbo:starring ?uri. ?uri text:"as" text:"Phileas Fogg" . }	SELECT ?uri WHERE{ ?var dbo:director res:Buzz_Kulik. ?var dbo:starring ?uri. ?var ?r ?cmd. ?cmd text: "80 Days" text:"World". ?var text: '"Phileas Fogg" AND "adaptation"'}	Kelime öbeği doğru olarak yakalanamadı.

Ekler kısmında FreeQA'in QALD-4 ve QALD-5 veri kümelerindeki soruları cevaplarken izlediği süreçler adım adım incelemektedir.

6. SONUÇ VE GELECEK ÇALIŞMALAR

Bu tez çalışmasında bağlı veriler üzerinde hibrid soru cevaplama ele alınmıştır. Literatürde, hibrid soru cevaplama ile ilgili fazla çalışma olmadığından dolayı, bu alan için soru cevaplama kapsamında yapılan çalışmalar ele alınmıştır.

FreeQA diye adlandırılan bu çalışmada öncelikle hibrid soru cevaplama alanında eksik olan temel maddeler araştırılmıştır. Bu kapsamda var olan iki temel eksiklik giderilmiştir. Bunlar: (1) Tüm soru tipleri için genel bir modelleme sağlanmıştır. (2) Farklı bilgi veritabanlarının bir arada kullanılması sağlanmıştır.

FreeQA, var olan eksiklikleri gidermenin yanında bir de yenilik getirmiştir. FreeQA'de, doğal dildeki soruları çizgeye dönüştürerek, bu çizgede aç gözlü arama algoritması uygulayarak SPARQL sorgusu elde edilmiştir.

FreeQA, QALD-4 ve QALD-5 veri kümeleri ile test edilmiştir. QALD-4 veri kümesi için %91 precision, %96 recall ve %92 f-measure ile başarılı olmuştur. QALD-5 için ise %74 precision, %81 recall ve %76 f-measure ile başarılı olmuştur.

Gelecek için iki geliştirme planlanmaktadır. İlk olarak, SPARQL üretimini paralel işleme (parallel processing) tabi tutarak performansta iyileştirme hedeflenmektedir. Son olarak da aç gözlü arama algoritmasının (greedy search) ağırlıklandırmasının optimum olacak şekilde ayarlanmasıdır. Ağırlıklandırma doğrudan sonucu etkilediği için burada bir iyileştirme yapılması hem sonuçların daha doğru gelmesini hem de performans olarak daha hızlı olmasını sağlayacaktır.

KAYNAKLAR

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data*, volume 4825 LNCS. Springer, 2007.

[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, 5(3):1–22, 2009.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, New York, NY., 2008. ACM.

[4] E. Cabrio, J. Cojan, F. Gandon, and A. Hallili. Querying multilingual dbpedia with qakis. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 194–198. Springer, 2013.

[5] J. D. Choi and M. Palmer. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 687–692. Association for Computational Linguistics, 2011.

[6] D. Damljanović, M. Agatonović, H. Cunningham, and K. Bontcheva. Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. *Web Semantics: Science, Services and Agents on the World Wide Web*, 19:1–21, 2013.

[7] C. Dima. Answering natural language questions with intui3. In *Conference and Labs of the Evaluation Forum (CLEF)*. QALD-4, 2014.

[8] P. Ferragina and U. Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *IEEE Software*, 29(1):70–75, 2012.

[9] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[10] A. Freitas, J. G. Oliveira, E. Curry, S. O'Riain, and J. C. P. da Silva. Treo: combining entity-search, spreading activation and semantic relatedness for querying linked data. In *Proc. of 1st Workshop on Question Answering over Linked Data (QALD-1) at the 8th Extended Semantic Web Conference (ESWC 2011)*, 2011.

[11] B. F. Green Jr, A. K. Wolf, C. Chomsky, and K. Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

[12] S. Hakimov, S. A. Oto, and E. Dogdu. Named entity recognition and disambiguation using linked data and graph-based centrality scoring. In *Proceedings of the 4th international workshop on semantic web information management*, pages 4:1–4:7, New York, NY, USA, 2012. ACM.

[13] S. He, Y. Zhang, K. Liu, and J. Zhao. Casia@ v2: A mln-based question answering system over linked data. *Proc. of QALD-4*, 2014.

[14] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, Edinburgh, 2011*, pages 782–792. Association for Computational Linguistics, 2011.

[15] B. Katz, G. Borchardt, and S. Felshin. Syntactic and semantic decomposition strategies for question answering from multiple resources. In *Proceedings of the AAAI 2005 workshop on inference for textual question answering*, pages 35–41, 2005.

[16] G. Klyne and J. J. Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

[17] O. Kolomiyets and M.-F. Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434, 2011.

[18] V. Lopez, M. Fernández, E. Motta, and N. Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3):249–265, 2011.

[19] V. Lopez, V. Uren, M. Sabou, and E. Motta. Is question answering fit for the semantic web? a survey. *Semantic Web*, 2(2):125–155, 2011.

[20] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[21] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM, 2011.

[22] D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.

[23] A. Moro, A. Raganato, and R. Navigli. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[24] S. Park, H. Shim, and G. G. Lee. Isoft at qald-4: Semantic similarity-based question answering system over linked data. In *CLEF 2014 Labs and Workshops, Notebook Papers (Qald task). CEUR Workshop Proceedings*. QALD-4, 2014.

[25] S. Shekarpour, E. Marx, A.-C. N. Ngomo, and S. Auer. Sina: Semantic interpretation of user queries for question answering on interlinked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 30:39–51, 2015.

[26] R. F. Simmons. Answering english questions by computer. *Automated Language Processing, edited by Harold Borko. Wiley, New York*, pages 253–289, 1967.

[27] R. Speck and A.-C. N. Ngomo. Ensemble learning for named entity recognition. In *The Semantic Web–ISWC 2014*, pages 519–534. Springer, 2014.

[28] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8–12, 2007*, pages 697–706. ACM, 2007.

[29] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16–20, 2012*, pages 639–648. ACM, 2012.

[30] C. Unger and P. Cimiano. Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In *Natural Language Processing and Information Systems*, pages 153–160. Springer, 2011.

[31] C. Unger, C. Forascu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question answering over linked data (qald-4). In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*, volume 1180 of CEUR Workshop Proceedings, pages 1172–1180, 2014.

[32] R. Usbeck, A.-C. N. Ngomo, L. Bühmann, and C. Unger. Hawk–hybrid question answering using linked data. In *The Semantic Web. Latest Advances and New Domains*, pages 353–368. Springer, 2015.

[33] W. Woods, R. Kaplan, and B. Nash-Webber. The lunar sciences natural language information system: Final report: Bbn report #2378., 1972.

[34] K. Xu, Y. Feng, and D. Zhao. Xser@ qald-4: Answering natural language questions via phrasal semantic parsing, 2014.

[35] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324. ACM, 2014.

EKLER

Tablo Ek1: QALD-4 Hibrid Soruları Sonuçları

1. Give me the currencies of all G8 countries.	
Entity	G8 countries : dbpedia.org/resource/G8_nations
Dependency Tree	>Give : give (0|VB) =>me : I (1|PRP) =>currencies : currency (2|NNS) ==>the : the (3|DT) ==>of : of (4|IN) ===>dbpedia.org/resource/G8_nations: (5|NNS)
Property Annotation	>currencies: dbpedia.org/ontology/currency, dbpedia.org/resource/Currency, dbpedia.org/class/yago/Currency113385913, dbpedia.org/ontology/Currency =>G8_nations: dbpedia.org/resource/G8_nations
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"member of" text:"G8". ?x dbo:currency ?uri . }
Our Query	SELECT DISTINCT ?var WHERE { ?uri ?r ?uri1. FILTER((CONTAINS(str(?uri1),"G8_nations") OR CONTAINS(str(?uri1),"G8 nations"))). ?uri <dbpedia.org/ontology/currency>?var.}
Gold Answer	dbpedia.org/resource/Euro dbpedia.org/resource/Euro_sign dbpedia.org/resource/Pound_sterling dbpedia.org/resource/United_States_dollar dbpedia.org/resource/Canadian_dollar dbpedia.org/resource/Russian_ruble dbpedia.org/resource/Japanese_yen
Our Answer	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall: = 1 Precision: = 1 F-measure: = 1

2. In which city was the assassin of Martin Luther King born?	
Entity	Martin Luther King : res:Martin_Luther_King,_Jr.
Dependency Tree	>born : bear (0|VBN) =>In : in (1|IN) ==>city : city (2|NN) ===>which : which (3|WDT) =>was : be (4|VBD) =>assassin : assassin (5|NN) ==>the : the (6|DT) ==>of : of (7|IN) ===>res:Martin_Luther_King,_Jr. (8|NN)
Property Annotation	>born(0|VBN|1(dbo:birthPlace : [])) =>city(2|NN|2(dbo:city, res:City : yago:City108524735, dbo:Settlement, yago:CityState108177958, dbo:City)) =>was(4|VBD|0([] : [])) =>assassin(5|NN|0([] : yago:Assassin109813696)) ==>res:Martin_Luther_King,_Jr.)
Gold Query	SELECT DISTINCT ?uri WHERE { res:Martin_Luther_King,_Jr. text:"assassin" ?x. ?x dbo:birthPlace ?uri. ?uri rdf:type dbo:Settlement. }
Our Query	SELECT DISTINCT ?target WHERE { ?source <dbo:birthPlace>?target. ?target a <dbo:Settlement>. ?source a <yago:Assassin109813696>. ?source <dbo:abstract>?abc. ?abc bif:contains '("Martin_Luther_King,_Jr.")'.}
Gold Answer	res:Alton,_Illinois
Our Answer	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall: = 1 Precision: = 1 F-measure: = 1

3. Which anti-apartheid activist graduated from the University of South Africa?	
Entity	University of South Africa: res:University_of_South_Africa
Dependency Tree	>graduated : graduate (0|VBD) =>activist : activist (1|NN) ==>Which : which (2|WDT) ==>anti-apartheid : anti-apartheid (3|JJ) =>from : from (4|IN) ==>res:University_of_South_Africa ===>the : the (6|DT)
Property Annotation	>graduated : graduate (0|VBD|1(dbo:almaMater :)) =>anti-apartheid_activist : anti-apartheid activist (1|CombinedNN|0(:)) =>res:University_of_South_Africa
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri rdf:type text:"anti-apartheid activist". ?uri dbo:almaMater res:University_of_South_Africa.}
Our Query	SELECT DISTINCT ?var WHERE { ?var dbo:almaMater <res:University_of_South_Africa>. ?var ?r ?c. FILTER(CONTAINS(str(?c),"Anti-apartheid_activist")). }
Gold Answer	res:Emma_Mashinini res:Cyril_Ramaphosa res:Gill_Marcus res:Winnie_Madikizela-Mandela res:Ahmed_Kathrada
Our Answer	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall: = 1 Precision: = 1 F-measure: = 1

4. How many Golden Globe awards did the husband of Mimi Rogers win?	
Entity	Mimi Rogers: res:Mimi_Rogers
Dependency Tree	>did : do (0|VBD) =>awards : award (1|NNS) ==>HOW_MANY : HOW_MANY (2|NNP) ==>Golden : Golden (3|NNP) ==>Globe : Globe (4|NNP) =>win : win (5|VB) ==>husband : husband (6|NN) ===>the : the (7|DT) ===>of : of (8|IN) ====>res:Mimi_Rogers
Property Annotation	>did(0|VBD|0(:)) =>Golden_Globe_awards((res:Golden_Globe_Award :)) ==>HOW_MANY(2|NNP|0(:)) =>win(5|VB|1(dbo:award :)) ==>husband((dbo:spouse, dbp:spouse, dbp:husband :)) ===>res:Mimi_Rogers
Gold Query	SELECT COUNT(DISTINCT ?awards) WHERE { res:Mimi_Rogers dbp:spouse ?uri. ?uri text:"win" ?award. ?award rdf:type text:"Golden Globe".}
Our Query	SELECT DISTINCT ?cmd WHERE { res:Mimi_Rogers dbp:spouse ?var. ?var ?r ?cmd. ?cmd ?r1 ?cmd1. FILTER ((CONTAINS(str(?cmd1), "Golden_Globe_Award") OR CONTAINS(str(?cmd1), "Golden_Globe_Award")) AND CONTAINS(str(?cmd) , "win")). } }
Gold Answer	3
Our Answer	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall: = 1 Precision: = 1 F-measure: = 1

5. Which recipients of the Victoria Cross died in the Battle of Arnhem?	
Entity	Victoria Cross: res:Victoria_Cross
Dependency Tree	>died : die (0|VBD) =>recipients : recipient (1|NNS|0(:)) ==>Which : which (2|WDT|0(:)) ==>of : of (3|IN|0(:)) ===>res:Victoria_Cross : res:victoria_cross (4|NN|0(:)) ====>the : the (5|DT|0(:)) =>in : in (6|IN|0(:)) ==>Battle : battle (7|NN|0(:)) ===>the : the (8|DT|0(:)) ===>of : of (9|IN|0(:)) ====>Arnhem : Arnhem (10|NNP|0(:))
Property Annotation	>died(0|VBD|1(dbo:deathPlace :)) =>recipients(1|NNS|1(dbo:award : yago:Recipient109627906)) ==>res:Victoria_Cross : res:victoria_cross (4|NN|0(:)) =>Battle of Arnhem : battle of (7|CombinedNN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri dbo:award res:Victoria_Cross. ?uri text:"died in" text:"Battle of Arnhem".}
Our Query	SELECT DISTINCT ?uri WHERE { ?uri dbo:award res:Victoria_Cross. ?uri dbo:deathPlace ?cmd. ?uri dbo:abstract ?abc. ?abc bif:contains '"Battle of Arnhem"' }
Gold Answer	res:Robert_Henry_Cain res:John_Grayburn
Our Answer	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall: = 1 Precision: = 1 F-measure: = 1

6. Where did the first man in space die?	
Entity	first man in space: res:Yuri_Gagarin
Dependency Tree	>die : die (0|VB|0(:)) =>Where : where (1|WRB|0(:)) =>did : do (2|VBD|0(:)) =>res:Yuri_Gagarin : res:yuri_gagarin (3|NN|0(:)) ==>the : the (4|DT|0(:))
Property Annotation	>die : die (0|VB|1(dbo:deathPlace :)) =>Where : where (1|WRB|0(: dbo:Place)) =>did : do (2|VBD|0(:)) =>res:Yuri_Gagarin : res:yuri_gagarin (3|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"is" text:"first man in space". ?x dbo:deathPlace ?uri . }
Our Query	SELECT DISTINCT ?var WHERE { res:Yuri_Gagarin dbo:deathPlace ?var. ?var a dbo:Place. }
Gold Answer	res:Kirzhach
Our Answer	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall: = 1 Precision: = 1 F-measure: = 1

7. How old was Steve Job's sister when she first met him?	
Entity	Steve Job (res:Steve_Jobs)
Dependency Tree	>sister : sister (0|NN|0(:)) =>HOW_OLD : how_old (1|NN|0(:)) =>was : be (2|VBD|0(:)) =>res:Steve_Jobs : res:steve_jobs (3|NN|0(:)) ==>'s : 's (4|POS|0(:)) =>met : meet (5|VBD|0(:)) ==>when : when (6|WRB|0(:)) ==>she : she (7|PRP|0(:)) ==>first : first (8|RB|0(:)) ==>him : he (9|PRP|0(:))
Property Annotation	>sister(0|NN|3(dbo:sisterStation, dbo:relative, res:Sibling :)) =>HOW_OLD(1|NN|1(dbo:birthYear :)) =>was(2|VBD|0(:)) =>res:Steve_Jobs(3|NN|0(:)) =>met(5|VBD|0(:)) ==>when(6|WRB|0(:))
Gold Query	SELECT ?n WHERE { res:Steve_Jobs dbo:relative ?uri. ?uri text:"did not meet until she was" ?n.}
Our Query	SELECT DISTINCT ?var1434799170907 WHERE { res:Steve_Jobs dbo:relative ?var. ?var dbo:birthYear ?var1434799170907. res:Steve_Jobs ?r ?cmd. FILTER(CONTAINS(str(?cmd) , "meet")). }
Gold Answer	25
Our Answer	1957
Accuracy	Recall = 0 Precision = 0 F-measure = 0

8. Which members of the Wu-Tang Clan took their stage name from a movie?	
Entity	Wu-Tang Clan: res:Wu-Tang_Clan)
Dependency Tree	>took : take (0|VBD|0(:)) =>members : member (1|NNS|0(:)) ==>Which : which (2|WDT|0(:)) ==>of : of (3|IN|0(:)) ===>res:Wu-Tang_Clan : res:wu-tang_clan (4|JJ|0(:)) =>name : name (5|NN|0(:)) ==>their : they (6|PRP$|0(:)) ==>stage : stage (7|NN|0(:)) =>from : from (8|IN|0(:)) ==>film : film (9|NN|0(:))
Property Annotation	>took((dbo:place :)) =>members((dbo:genus, dbo:phylum, dbo:order, dbo:board, dbo:kingdom, dbo:family, dbo:class, dbp:pastMembers, dbo:bandMember, dbo:formerBandMember, res:Member : yago:Member110307234, yago:Extremity105559908)) ==>res:Wu-Tang_Clan((:)) =>stage_name((dbp:stageName :)) =>film((: yago:Film106262567, yago:Movie106613686))
Gold Query	SELECT DISTINCT ?uri WHERE { res:Wu-Tang_Clan dbo:bandMember ?uri. ?uri text:"stage name" ?x. ?x text:"from" text:"movie".}
Our Query	SELECT DISTINCT ?var WHERE { res:Wu-Tang_Clan dbo:bandMember ?var. ?var dbo:abstract ?abs. ?abs bif:contains '("stage name")'. res:Wu-Tang_Clan ?r ?c.FILTER(CONTAINS(str(?c), "film"))}}
Gold Answer	res:Ghostface_Killah, res:Method_Man
Our Answer	res:Cappadonna, res:Ghostface_Killah, res:Raekwon, res:Method_Man, res:Inspectah_Deck, res:Rza
Accuracy	Recall = 1 Precision = 0.33 F-measure = 0.49

9. Which writers had influenced the philosopher that refused a Nobel Prize?	
Entity	Nobel Prize: res:Nobel_Prize)
Dependency Tree	>influenced : influence (0|VBN|0(:)) =>writers : writer (1|NNS|0(:)) ==>Which : which (2|WDT|0(:)) =>had : have (3|VBD|0(:)) =>philosopher : philosopher (4|NN|0(:)) ==>the : the (5|DT|0(:)) ==>refused : refuse (6|VBD|0(:)) ===>that : that (7|WDT|0(:)) ===>res:Nobel_Prize : res:nobel_prize (8|NN|0(:))
Property Annotation	>influenced(0|VBN|1(dbo:influencedBy :)) =>writers(1|NNS|3(dbo:writer, dbp:presenter, res:Writer : yago:Writer110801291, yago:Writer110794014,dbo:Writer)) =>had(3|VBD|0(:)) =>philosopher(4|NN|1(res:The_Philosopher : yago:Philosopher110423589,dbo:Philosopher)) ==>refused(6|VBD|0(:)) ===>res:Nobel_Prize : res:nobel_prize (8|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x rdf:type dbo:Philosopher. ?x text:"refuse" text:"Nobel Prize". ?x dbo:influencedBy ?uri. ?uri rdf:type dbo:Writer.}
Our Query	SELECT DISTINCT ?var WHERE { ?uri dbo:abstract ?uriAbc. ?uriAbc bif:contains '("Nobel_Prize")'. ?uri dbo:influencedBy ?var. ?var a dbo:Writer. ?uri a yago:Philosopher110423589. ?uri ?r ?c. FILTER(CONTAINS(str(?c) , "refuse")). }
Gold Answer	res:Gustave_Flaubert
Our Answer	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

10. Under which king did the British prime minister that signed the Munich agreement serve?	
Entity	Munich agreement (res:Munich_Agreement)
Dependency Tree	>serve : serve (0|VB) =>Under : under (1|IN) ==>king : king (2|NN) ===>which : which (3|WDT) =>did : do (4|VBD) =>minister : minister (5|NN) ==>British : british (7|JJ) ==>prime : prime (8|JJ) ==>signed : sign (9|VBD) ===>that : that (10|WDT) ===>res:Munich_Agreement : res:munich_agreement (11|NN)
Property Annotation	>serve(:) =>king(dbo:monarch, dbp:king, res:Monarch : yago:King110231515) =>British prime minister (res:Prime_Minister_of_the_United_Kingdom) ==>signed(:) ===>Munich_Agreement(:)
Gold Query	SELECT DISTINCT ?uri WHERE { ?x rdf:type yago:PrimeMinistersOfTheUnitedKingdom. ?x text:"sign" text:"Munich Agreement". ?x dbo:monarch ?uri.}
Our Query	SELECT DISTINCT ?var WHERE { ?uri dbo:abstract ?a. ?a bif:contains '("Munich_Agreement")'. ?uri dbo:monarch ?var. ?uri ?r ?c. ?uri ?r1 ?c1. ?uri ?r2 ?c2. filter(contains(str(?c2),"sign") and contains(str(?c),"serve") and contains(str(?c1),"Prime_Minister_of_the_United_Kingdom")) }
Gold A.	res:George_VI
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

11. Who composed the music for the film that depicts the early life of Jane Austin?	
Entity	Jane Austen (res:Jane_Austen)
Dependency Tree	>composed : compose (0|VBN) =>Who : who (1|WP) =>music : music (2|NN) =>for : for (4|IN) ==>film : film (5|NN) ===>depicts : depict (7|VBZ) ====>that : that (8|WDT) ====>life : life (9|NN) =====>early : early (11|JJ) =====>of : of (12|IN) ======>res:Jane_Austen : res:jane_austen (13|NN)
Property Annotation	>composed(0|VBN|1(dbo:musicComposer :)) =>Who(1|WP|0(: dbo:Agent)) =>music(2|NN|3(dbo:musicComposer, dbp:music)) =>film(5|NN|3(dbo:film, res:Film : yago:Film106262567)) ==>depicts(7|VBZ|0(:)) ===>early_life(9|CombinedNN|0(:)) ====>res:Jane_Austen(13|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x rdf:type dbo:Film. ?x dbo:musicComposer ?uri. ?x text:"depicts the early life of" text:"Jane Austin"}
Our Query	SELECT DISTINCT ?var WHERE { ?uri dbo:musicComposer?var. ?uri dbo:abstract ?a. ?a bif:contains '("early life") AND ("Jane_Austen")'. ?uri a yago:Movie106613686. ?var a dbo:Agent. ?uri ?r ?c. FILTER(CONTAINS(str(?c),"depict")).}
Gold A.	res:Adrian_Johnston
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

12. Who succeeded the pope that reigned only 33 days?	
Entity	-
Dependency Tree	>succeeded : succeed (0|VBD) =>Who : who (1|WP) =>pope : pope (2|NN) ==>the : the (3|DT) ==>reigned : reign (4|VBD) ===>that : that (5|WDT) ===>days : day (6|NNS) ====>33 : 33 (7|CD) =====>only : only (8|RB)
Property Annotation	>succeeded(0|VBD|1(dbp:successor :)) =>Who(1|WP|0(: dbo:Agent)) =>pope(2|NN|2(res:Pope, dbp:pope : yago:Pope110453533)) ==>reigned(4|VBD|0(:)) ===>only_33_days(6|CombinedNN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x rdf:type yago:Popes . ?x text:"reign" text:"33 days". ?x dbo:successor ?uri . }
Our Query	SELECT DISTINCT ?var WHERE { ?uri dbo:abstract ?a. ?a bif:contains '("33 day") AND ("Pope")'. ?uri dbp:successor ?var. ?uri a dbo:Agent.?uri ?r ?c. FILTER(CONTAINS(str(?cmd),"reign")).}
Gold A.	res:Pope_John_Paul_II
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

13. On which island did the national poet of Greece die?	
Entity	Greece (res:Greece)
Dependency Tree	>die : die (0|VB|0) =>On : on (1|IN|) ==>island : island (2|NN) ===>which : which (3|WDT) =>did : do (4|VBD) =>poet : poet (5|NN) ==>the : the (6|DT) ==>national : national (7|JJ) ==>of : of (8|IN) ===>res:Greece : res:greece (9|NN)
Property Annotation	>die(0|VB|1(dbo:deathPlace :)) =>island(2|NN|2(dbo:island : yago:Island109316454,dbo:Island)) =>national_poet(5|CombinedNN|1(dbp:nationalPoet :)) ==>res:Greece(9|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"considered" text:"national poet of Greece". ?x dbo:deathPlace ?uri. ?uri a yago:Island109316454.}
Our Query	SELECT DISTINCT ?trg WHERE { ?var dbo:deathPlace ?trg. ?var ?r1 res:Greece. ?trg a yago:Island109316454. ?var dbo:abstract ?a. ?a bif:contains '("national poet")'. }
Gold A.	res:Corfu
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

14. Which horses did The Long Fellow ride?	
Entity	The Long Fellow: res:Lester_Piggott
Dependency Tree	>did : do (0|VBD) =>horses : horse (1|NNS) ==>Which : which (2|WDT) =>ride : ride (3|NN) ==>res:Lester_Piggott : res:lester_piggott (4|NN)
Property Annotation	=>horses(1|NNS|4(dbp:horses,dbo:raceHorse, dbp:horse, res:Horse : yago:Horse102374451)) =>ride(3|NN|2(dbo:raceHorse, res:Ride :)) ==>res:Lester_Piggott(4|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"known as" text:"The Long Fellow". ?x dbo:raceHorse ?uri . }
Our Query	SELECT DISTINCT ?var WHERE { res:Lester_Piggott dbo:raceHorse ?var. }
Gold Answer	res:Sir_Ivor, res:Crepello, res:Roberto_(horse) res:Nijinsky_(horse), res:The_Minstrel res:Petite_Etoile, res:Royal_Academy_(horse) res:Rodrigo_de_Triano, res:St._Paddy res:Empery, res:Never_Say_Die_(horse) res:Alleged_(horse), res:Teenoso, res:Shadeed
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

15. Of the people that died of radiation in Los Alamos, whose death was an accident?	
Entity	Los Alamos: res:Los_Alamos,_New_Mexico
Dependency Tree	>Of : of (0|IN) =>people : people (1|NNS) ==>died : die (3|VBD) ===>that : that (4|WDT) ===>of : of (5|IN) ====>radiation : radiation (6|NN) ===>in : in (7|IN) ====>res:Los_Alamos,_New_Mexico (8|NN) =====>accident : accident (9|NN) ======>whose : whose (10|WP$) ======>death : death (11|NN) ======>was : be (12|VBD)
Property Annotation	>people(1|NNS|1(dbp:people :)) =>died(3|VBD|1(dbo:deathPlace :)) ==>radiation(6|NN|2(res:Acute_radiation_syndrome :)) ==>res:Los_Alamos,_New_Mexico(8|NN|0(:)) ===>accident(9|NN|1(res:Accident : yago:Accident107301336)) ====>death(11|NN|5(dbo:deathPlace, dbo:deathCause, dbp:death, res:Death : yago:Death107355491))
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri dbo:deathPlace res:Los_Alamos,_New_Mexico. ?uri dbo:deathCause res:Acute_radiation_syndrome. ?uri text:"death circumstance" text:"accident" . }
Our Query	SELECT DISTINCT ?var WHERE { ?var dbo:deathPlace res:Los_Alamos,_New_Mexico. ?var ?r ?cmd. ?var ?r1432497095955 ?cmd1. FILTER(CONTAINS(str(?cmd1), "Acute_radiation_syndrome" AND CONTAINS(str(?cmd) , "Accident")). }
Gold A.	res:Louis_Slotin, res:Harry_K._Daghlian,_Jr.
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

16. Which buildings owned by the crown overlook the North Sea?	
Entity	North Sea (res:North_Sea)
Dependency Tree	>owned : own (0|VBN) =>buildings : building (1|NNS) ==>Which : which (2|WDT) =>by : by (3|IN) ==>crown : crown (4|NN) ===>the : the (5|DT) =>overlook : overlook (6|VB) ==>res:North_Sea : res:north_sea (7|NN) ===>the : the (8|DT)
Property Annotation	>owned(0|VBN|1(dbo:owner :)) =>buildings(1|NNS|2(dbp:buildings, res:Building : yago:Building102913152,dbo:ArchitecturalStructure,dbo:Building)) =>crown(4|NN|1(res:The_Crown : yago:Peak108617963)) =>overlook(6|VB|0(:)) ==>res:North_Sea(7|NN|0(:))
Gold Query	SELECT ?uri WHERE { ?uri rdf:type dbo:Building. ?uri dbo:owner res:The_Crown. ?uri text:"overlook" text:"North Sea". }
Our Query	SELECT DISTINCT ?var WHERE { ?var dbo:owner res:The_Crown. ?var a dbo:ArchitecturalStructure. ?var ?r ?cmd. FILTER(CONTAINS(str(?cmd) , "overlook")). ?var dbo:abstract ?a. ?a bif:contains '("North_Sea")'. }
Gold Answer	res:Scarborough_Castle
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

17. Which buildings in art deco style did Shreve, Lamb and Harmon design?	
Entity	Shreve, Lamb and Harmon: res:Shreve,_Lamb_&_Harmon
Dependency Tree	>did : do (0|VBD) =>buildings : building (1|NNS) ==>Which : which (2|WDT) ===>style : style (4|NN) ====>art : art (5|NN) ====>deco : deco (6|NN) =>res:Shreve,_Lamb_&_Harmon(11|NN) ==>design : design (10|NN)
Property Annotation	>did(0|VBD|0(:)) =>buildings(1|NNS|1(res:Building : yago:Building102913152, dbo:ArchitecturalStructure)) ==>art_deco_style(4|CombinedNN|0(:)) =>res:Shreve,_Lamb_&_Harmon(11|NN|0(:)) ==>design(10|NN|4(dbo:architect,dbo:designer, dbp:design, res:Design : yago:Design105728678))
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri dbo:architect res:Shreve,_Lamb_&_Harmon. { ?uri dbo:architecturalStyle res:Art_Deco . } UNION { ?uri text:"style text:"art deco" . } }
Our Query	SELECT ?uri WHERE { ?uri dbo:architect res:Shreve,_Lamb_&_Harmon. ?uri dbo:abstract ?a. ?a bif:contains '"style" AND "art deco"'. ?uri a yago:Building102913152. }
Gold Answer	res:Empire_State_Building, res:Hill_Building res:Joel_W._Solomon_Federal_Building_and_United_States _Courthouse, res:500_Fifth_Avenue
Our A.	res:Empire_State_Building
Accuracy	Recall = 0.25 Precision = 1 F-measure = 0.40

18. Which birds are protected under the National Parks and Wildlife Act?	
Entity	the National Parks: res:The_National_Parks
Dependency Tree	>protected : protect (0|VBN) =>birds : bird (1|NNS) ==>Which : which (2|WDT) =>are : be (3|VBP) =>under : under (4|IN) ==>res:The_National_Parks(5|NNS) ===>and : and (6|CC) ===>Act : Act (7|NNP) ====>Wildlife : Wildlife (8|NNP)
Property Annotation	>protected(0|VBN|0(:)) =>birds(1|NNS|1(res:Bird : yago:Bird107644382, dbo:Bird)) =>res:The_National_Parks(5|NNS|0(:)) ==>Wildlife_Act(7|CombinedNN|1(res:Wildlife_Act :))
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri rdf:type dbo:Bird. ?uri text:"protected under" text:"National Parks and Wildlife Act". }
Our Query	SELECT DISTINCT ?uri WHERE { ?uri dbo:abstract ?a. ?a bif:contains '("The_National_Parks") AND ("Wildlife_Act")'. ?uri a dbo:Bird. ?uri ?r ?cmd . FILTER(CONTAINS(str(?cmd) , "protect")). }
Gold Answer	res:Chestnut_Teal res:Bar-shouldered_Dove res:Black-faced_Cuckooshrike res:Australian_Shelduck res:Australasian_Shoveler res:Yellow_Thornbill
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

19. Which country did the first known photographer of snowflakes come from?	
Entity	first known photographer of snowflake: res:Wilson_Bentley
Dependency Tree	>come : come (0|VBN) =>did : do (1|VBD) =>res:Wilson_Bentley(2|NN) =>from : from (3|IN) ==>country : country (4|NN) ===>Which : which (5|WDT)
Property Annotation	>come(0|VBN|1(dbo:birthPlace :)) =>res:Wilson_Bentley(2|NN|0(:)) =>country(4|NN|2(dbo:country, res:Country : yago:Country108544813 ,dbo:Country))
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri text:"is" text:"first known photographer of snowflakes". ?uri dbo:birthPlace ?x . ?x rdf:type dbo:Country . }
Our Query	SELECT DISTINCT ?var1 WHERE { res:Wilson_Bentley dbo:birthPlace ?var. ?var dbo:country ?var1. }
Gold Answer	res:United_States
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

20. List all the battles fought by the lover of Cleopatra.	
Entity	Cleopatra (res:Cleopatra)
Dependency Tree	>List : list (0|VB) =>battles : battle (1|NNS) ==>all : all (2|PDT) ==>the : the (3|DT) ==>fought : fight (4|VBN) ===>by : by (5|IN) ====>lover : lover (6|NN) =====>the : the (7|DT) =====>of : of (8|IN) ======>res:Cleopatra(9|NN)
Property Annotation	>battles(1|NNS|2(dbo:battle : yago:Battle100953559)) =>fought(4|VBN|1(dbo:battle :)) ==>lover(6|NN|5(dbo:spouse, dbp:predecessor, dbp:spouse,dbo:predecessor, res:The_Lover : yago:Lover109622302, yago:Lover109622745)) ===>res:Cleopatra(9|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"lover" text:"Cleopatra". ?x dbo:battle ?uri . }
Our Query	SELECT DISTINCT ?var1 WHERE { res:Cleopatra dbo:spouse ?var. ?var dbo:battle ?var1. }
Gold Answer	res:Gallic_Wars res:Final_War_of_the_Roman_Republic res:Liberators'_civil_war res:Battle_of_Mutina res:Caesar's_Civil_War res:Antony's_Parthian_War
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

21. Are the Rosetta Stone and the Gayer-Andersen cat exhibited in the same museum?	
Entity	Rosetta Stone: res:Rosetta_Stone, Gayer-Anderson cat: res:Gayer-Anderson_cat
Dependency Tree	>exhibits : exhibit (0|VBZ) =>res:Rosetta_Stone(1|NN) ==>Are : be (2|VBP) ==>and : and (4|CC) ==>res:Gayer-Anderson_cat(5|NN) =>in : in (7|IN) ==>museum : museum (8|NN) ===>same : same (10|JJ)
Property Annotation	>exhibits(0|VBZ|0(:)) =>res:Rosetta_Stone(1|NN|0(:)) ==>res:Gayer-Anderson_cat(5|NN|0(:)) =>same_museum(8|CombinedNN|4(dbp:location, dbp:museum, dbo:museum, res:Museum : yago:Museum103800563,dbo:Museum))
Gold Query	ASK WHERE { ?m1 text:"exhibits" res:Rosetta_Stone. res:Gayer-Anderson_Cat dbo:museum ?x . ?m2 rdfs:label ?m2 . FILTER (?m1=?m2) }
Our Query	SELECT DISTINCT ?source WHERE { ?source dbo:abstract ?sourceabc. ?sourceabc bif:contains '("exhibits" OR "exhibit*")'. ?t dbp:location ?source. FILTER(?t=res:Rosetta_Stone). } SELECT DISTINCT ?source WHERE { ?source dbo:abstract ?sourceabc. ?sourceabc bif:contains '("exhibits" OR "exhibit*")'. ?t dbo:museum ?source.FILTER(?t=res:Gayer-Anderson_cat). }
Gold A.	true
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

22. Which actress starring in the TV series Friends owns the production company Coquette Productions?	
Entity	Coquette Productions (res:Coquette_Productions)
Dependency Tree	>starring : star (0|VBG) =>actress : actress (1|NN) ==>Which : which (2|WDT) =>in : in (3|IN) ==>series : series (4|NN) ===>TV : tv (6|NN) ===>owns : own (7|VBZ) ====>Friends : Friends (8|NNPS) ====>res:Coquette_Productions(9|NNS) =====>production : production (11|NN) =====>company : company (12|NN)
Property Annotation	>starring(0|VBG|1(dbo:starring :)) =>actress(1|NN|1(res:Actor : yago:Actress109767700)) =>TV_series(4|CombinedNN|0(:)) ==>owns(7|VBZ|1(dbo:owner :)) ===>Friends(8|NNPS|2(dbp:friends, res:Friends :)) ===>Coquette_Productions(9|NNS|0(:)) ====>production_company(11|CombinedNN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { res:Friends dbo:starring ?uri . ?uri text:"own" text:"the production company Coquette Productions" . }
Our Query	SELECT DISTINCT ?var WHERE { res:Friends dbo:starring ?var. res:Friends ?r ?c. ?var ?r1 ?c1. ?var dbo:abstract ?a.?a bif:contains '("Coquette_Productions")'. FILTER(CONTAINS(str(?c1),"own") AND CONTAINS(str(?c),"Actor")) }
Gold A.	res:Courteney_Cox
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

23. Dakar is the capital of which country member of the African Union?	
Entity	Dakar (res:Dakar)
Dependency Tree	>capital : capital (0|NN) =>res:Dakar : res:dakar (1|NN) =>is : be (2|VBZ) =>the : the (3|DT) =>of : of (4|IN) ==>member : member (5|NN) ===>which : which (6|WDT) ===>country : country (7|NN) ===>of : of (8|IN) ====>Union : Union (9|NNP) =====>the : the (10|DT) =====>African : African (11|NNP)
Property Annotation	>capital(0|NN|2(dbo:capital : yago:Capital108518505)) =>res:Dakar(1|NN|0(:)) =>member(5|NN|10(res:Member : yago:Member110307234)) ==>country(5|NN|2(dbo:country, res:Country : yago:Country108544813, dbo:Country)) ==>African_Union(8|CombinedNN|1(res:African_Union :))
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri text:"member of" text:"the African Union". ?uri dbo:capital res:Dakar . }
Our Query	SELECT DISTINCT ?var WHERE { ?var dbo:capital res:Dakar. ?var1 dbo:country ?var. ?var dbo:abstract ?a. ?a bif:contains '"member"'. ?var ?r ?cmd. FILTER(CONTAINS(str(?cmd),"African_Union"))}
Gold A.	res:Senegal
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

24. When was the composer of the opera Madama Butterfly born?	
Entity	Madama Butterfly: res:Madama_Butterfly
Dependency Tree	>born : bear (0|VBN) =>When : when (1|WRB) =>was : be (2|VBD) =>composer : composer (3|NN) ==>the : the (4|DT) ==>of : of (5|IN) ===>res:Madama_Butterfly(6|NN) ====>the : the (7|DT) ====>opera : opera (8|NN)
Property Annotation	>born(0|VBN|1(dbo:birthDate :)) =>composer(3|NN|3(dbo:composer,dbo:musicComposer : yago:Composer109947232)) ==>res:Madama_Butterfly(6|NN|0(:)) ===>opera(8|NN|1(res:Opera : yago:Opera107026352))
Gold Query	SELECT DISTINCT ?date WHERE { res:Madama_Butterfly text:"composer" ?x . ?x dbo:birthDate ?date . }
Our Query	SELECT DISTINCT ?var WHERE { ?uri dbo:abstract ?a. ?a bif:contains '("Madama_Butterfly")'. ?uri dbo:birthDate ?var. ?var1 dbo:musicComposer ?uri. ?uri ?r ?cmd . FILTER(CONTAINS(str(?cmd),"Opera")). }
Gold A.	1858-12-22
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

25. Which street basketball player was diagnosed with Sarcoidosis?	
Entity	Sarcoidosis (res:Sarcoidosis)
Dependency Tree	>diagnosed : diagnose (0|VBN) =>player : player (1|NN) ==>Which : which (2|WDT) ==>street : street (3|NN) ==>basketball : basketball (4|NN) =>was : be (5|VBD) =>with : with (6|IN) ==>res:Sarcoidosis(7|NN)
Property Annotation	>diagnosed(0|VBN|0(:)) =>street_basketball_player(1|CombinedNN|0(: yago:StreetBasketballPlayers)) =>res:Sarcoidosis(7|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri rdf:type yago:StreetBasketballPlayers. ?uri text:"diagnosed with" text:"Sarcoidosis". }
Our Query	SELECT DISTINCT ?uri WHERE { ?uri dbo:abstract ?a. ?a bif:contains '("Sarcoidosis")'. ?uri a yago:StreetBasketballPlayers. ?uri ?r ?cmd. FILTER(CONTAINS(str(?cmd) , "hospital"))}
Gold A.	res:Slick_Watts
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

Tablo Ek2: QALD-5 Hibrid Soruları Sonuçları

1. Who was vice president under the president who authorized atomic weapons against Japan during World War II?	
Entity	Japan (res:Japan)
Dependency Tree	>Who : who (0|WP) =>was : be (1|VBD), president : president (2|NN) ==>vice : vice (3|NN), under : under (4|IN) ===>president : president (5|NN) ====>the : the (6|DT), approved : approve (7|VBD) =====>who : who (8|WP), weapons : weapon (9|NNS) ======>atomic : atomic (10|JJ) =====>against : against (11|IN) ======>res:Japan : res:japan (12|NN) =====>during : during (13|IN) ======>II : II (14|NNP) =======>War : War (15|NNP), World : World (16|NNP)
Property Annotation	>Who (0|WP|0(res:Agent :)) =>vice president (2|CombinedNN|2(dbp:vicePresident :)) ==>president(5|NN|2(res:The_President, dbp:president : yago:President110467179, dbo:President)) ===>approved(7|VBD|0(:)) ====>atomic weapons(9|CombinedNN|0(:)) ====>res:Japan : res:japan (12|NN|0(:)) ====>World War II(14|CombinedNN|1(res:World_War_II :))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"authorized" text:"atomic weapons against Japan". ?x dbo:vicePresident ?uri . }
Our Query	SELECT DISTINCT ?var WHERE{ ?uri dbo:abstract ?a. ?a bif:contains '("Japan" AND "World War II" AND "atomic weapons" AND "against" AND "approved")'. ?uri dbp:vicePresident ?var. ?var a dbo:Agent }
Gold A.	res:Alben_W._Barkley
Our Answer	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall: = 1 Precision: = 1 F-measure: = 1

2. Which anti-apartheid activist was born in Mvezo?	
Entity	Mvezo (res:Mvezo)
Dependency Tree	>born : bear (0|VBN) =>activist : activist (1|NN) ==>Which : which (2|WDT) ==>anti-apartheid : anti-apartheid (3|JJ) =>was : be (4|VBD) =>in : in (5|IN) ==>res:Mvezo (6|NN)
Property Annotation	>born(0|VBN|1(dbo:birthPlace :)) =>anti-apartheid_activist (1CombinedNN0(:)) =>res:Mvezo(6NN0(:))
Gold Query	SELECT DISTINCT ?uri WHERE{ ?uri rdf:type text:"anti-apartheid activist". ?uri dbo:birthPlace res:Mvezo . }
Our Query	SELECT DISTINCT ?var WHERE{ ?var dbo:birthPlace res:Mvezo. ?var ?r ?cmd. FILTER(CONTAINS(str(?cmd),"Anti-apartheid_activist"))}
Gold A.	res:Nelson_Mandela
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

3. How many Golden Globe awards did the daughter of Henry Fonda win?	
Entity	Henry Fonda (res:Henry_Fonda)
Dependency Tree	>did : do (0|VBD) =>awards : award (1|NNS) ==>How many : HOW_MANY (2|NNP) ==>Golden : Golden (3|NNP) ==>Globe : Globe (4|NNP) =>win : win (5|VB) ==>daughter : daughter (6|NN) ===>the : the (7|DT) ===>of : of (8|IN) ====>res:Henry_Fonda(9|NN)
Property Annotation	>Golden Globe awards(1|CombinedNN(res:Golden_Globe_Award :)) >win(5|VB(dbo:award :)) =>daughter(6|NN(dbo:child :)) ==>Henry_Fonda(9|NN(:))
Gold Query	SELECT ?n WHERE { ?uri dbo:parent res:Henry_Fonda. ?uri text:"number of Golden Globe awards" ?n . }
Our Query	SELECT count(DISTINCT ?cmd) WHERE { res:Henry_Fonda ?r ?cmd. res:Henry_Fonda ?r1 ?cmd1. ?cmd ?r2 ?cmd2. FILTER((CONTAINS(str(?cmd2), "Golden_Globe_Award") AND (CONTAINS(str(?cmd1),"daughter") AND (CONTAINS(str(?cmd),"win")). }
Gold A.	2
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

4. Which building owned by the Bank of America was featured in the TV series MegaStructures?	
Entity	Bank of America (res:Bank_of_America)
Dependency Tree	>owned : own (0|VBN) =>building : building (1|NN) ==>Which : which (2|WDT) =>by : by (3|IN) ==>res:Bank_of_America(4|NN) ===>the : the (5|DT) ===>featured : feature (6|VBN) ====>was : be (7|VBD) ====>in : in (8|IN) =====>MegaStructures : MegaStructures (9|NNP) ======>the : the (10|DT) ======>TV : tv (11|NN) ======>series : series (12|NN)
Property Annotation	>owned(0|VBN|1(dbo:owner :)) =>building(1|NN|2(dbp:building : yago:Building102913152)) =>res:Bank_of_America(4|NN|0(:)) ==>featured(6|VBN|1(dbo:starring :)) ===>MegaStructures(9|NNP|1(res:MegaStructures :)) ====>TV series(11|CombinedNN|1(res:Television_program :))
Gold Query	SELECT ?uri WHERE { ?uri rdf:type dbo:Building. ?uri dbo:owner res:Bank_of_America. ?uri text:"featured in" text:"MegaStructures".}
Our Query	SELECT DISTINCT ?var WHERE { ?var dbo:owner res:Bank_of_America. ?var a yago:Building102913152. ?var ?r ?cmd. ?var ?r1 ?cmd1. FILTER((CONTAINS(str(?cmd1), "MegaStructures")) AND (CONTAINS(str(?cmd),"feature")))}
Gold A.	res:Bank_of_America_Tower_(New_York_City)
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

5. Gaborone is the capital of which country member of the African Union?	
Entity	Gaborone (res:Gaborone)
Dependency Tree	>capital : capital (0|NN) =>res:Gaborone (1|NN) =>is : be (2|VBZ) =>the : the (3|DT) =>of : of (4|IN) ==>member : member (5|NN) ===>which : which (6|WDT) ===>country : country (7|NN) ===>of : of (8|IN) ====>Union : Union (9|NNP) =====>the : the (10|DT) =====>African : African (11|NNP)
Property Annotation	>capital(0|NN|2(dbo:capital : yago:Capital108518505)) =>res:Gaborone (1|NN|0(:)) =>member(5|NN|10(res:Member : yago:Member110307234)) ==>country(5|NN|2(dbo:country, res:Country : yago:Country108544813,dbo:Country)) ==>African_Union(8|CombinedNN|1(res:African_Union :))
Gold Query	SELECT DISTINCT ?uri WHERE { ?uri text:"member of" text:"African Union". ?uri dbo:capital res:Gaborone . }
Our Query	SELECT DISTINCT ?var WHERE { ?var dbo:capital res:Gaborone. ?var1 dbo:country ?var. ?var dbo:abstract ?a. ?a bif:contains '"member"'. ?var ?r ?cmd. FILTER(CONTAINS(str(?cmd),"African_Union"))}
Gold A.	res:Botswana
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

6. For which movie did the daughter of Francis Ford Coppola receive an Oscar?	
Entity	Francis Ford Coppola (res:Francis_Ford_Coppola)
Dependency Tree	>did : do (0|VBD) =>For : for (1|IN) ==>movie : movie (2|NN) ===>which : which (3|WDT) =>receive : receive (4|VBP) ==>daughter : daughter (5|NN) ===>the : the (6|DT) ===>of : of (7|IN) ====>res:Francis_Ford_Coppola (8|NN) ==>Award : award (9|NN) ===>an : a (10|DT) ===>Academy : academy (11|NN)
Property Annotation	>did : do (0|VBD) =>movie(2|NN|2(dbp:movie : yago:Movie106613686)) =>receive(4|VBP|1(dbo:award :)) ==>daughter(5|NN|3(dbo:child, dbp:daughter :)) ===>res:Francis_Ford_Coppola (8|NN|0(:)) ==>Academy_Award (9|CombinedNN|1(res:Academy_Awards :))
Gold Query	SELECT DISTINCT ?uri WHERE{ res:Francis_Ford_Coppola dbo:child ?x. ?x text:"receive Oscar for" ?uri . }
Our Query	No answer
Gold A.	res:Lost_in_Translation_(film)
Our A.	No Answer
Accuracy	Recall = 0 Precision = 0 F-measure = 0

7. Which city does the first person to climb all 14 eight-thousanders come from?	
Entity	-
Dependency Tree	>does : do (0|VBZ) |=>city : city (1|NN) |==>Which : which (2|WDT) |=>person : person (3|NN) |==>the : the (4|DT) |==>first : first (5|JJ) |==>climb : climb (6|VB) |===>to : to (7|TO) |===>come : come (8|VBP) |====>eight-thousanders : eight-thousander (9|NNS) |=====>all : all (10|DT) |=====>14 : fourteen (11|CD) |====>from : from (12|IN)
Property Annotation	|=>city(1|NN|2(dbo:city : dbo:Settlement, dbo:City)) |=>person(3|NN|2(dbo:person, res:Person :)) |==>first(5|JJ|0(:)) |==>climb(6|VB|0(:)) |===>come(8|VBP|1(dbo:birthPlace :)) |====>14 eight-thousanders(9|NNS|1(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"is" text:"first person to climb all fourteen eight-thousanders". ?x dbo:birthPlace ?uri. ?uri rdf:type dbo:Settlement. }
Our Query	SELECT DISTINCT ?cmd WHERE { ?var dbo:birthPlace ?cmd. ?var dbo:abstract ?a. ?a bif:contains '"fourteen eight-thousanders" AND "first"'. ?var a dbo:Agent. ?cmd a dbo:Settlement. ?var ?r ?cmd1. FILTER(CONTAINS(str(?cmd1,"climb")))}
Gold A.	res:Brixen
Our A.	res:Brixen, res:Katowice
Accuracy	Recall = 1 Precision = 0.5 F-measure = 0.66

8. At which college did the only American actor that received the César Award study?	
Entity	César Award (res:Cesar_Award)
Dependency Tree	>did : do (0|VBD) |=>At : at (1|IN) |==>college : college (2|NN) |===>which : which (3|WDT) |=>actor : actor (4|NN) |==>the : the (5|DT) |==>only : only (6|JJ) |==>American : american (7|JJ) |==>received : receive (8|VBD) |===>that : that (9|WDT) |===>study : study (10|NN) |====>res:César_Award (11|NN)
Property Annotation	>did(0|VBD|0(:)) |=>college(2|NN|2(dbo:college, res:College :)) |=>only_American_actor(4|CombinedNN|1(res:Actor :)) |==>received(8|VBD|1(dbo:award :)) |===>study(10|NN|3(dbo:almaMater, dbo:university, res:Study :)) |====>César_Award (11|NN|0(:))
Gold Query	SELECT DISTINCT ?string WHERE { ?x text:"is" text:"the only American actor that received the César Award". ?x dbo:almaMater ?uri . }
Our Query	SELECT DISTINCT ?uri WHERE { ?var dbo:almaMater ?uri. ?var dbo:abstract ?a. ?a bif:contains '"César Award" AND "the only American actor" AND "receive"'}
Gold A.	res:Queens_College,_City_University_of_New_York
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

9. Did Napoleon's first wife die in France?	
Entity	Napoleon (res:Napoleon), France (res:France)
Dependency Tree	>die : die (0|VB) |=>Did : do (1|VBD) |=>res:Napoleon(2|NN) |==>wife : wife (4|NN) |===>first : first (5|JJ) |=>in : in (6|IN) |==>res:France (7|NN)
Property Annotation	>die(0|VB|1(dbo:deathPlace :)) |=>res:Napoleon(2|NN|0(:)) |==>first_wife(4|CombinedNN|0(:)) |=>res:France (7|NN|0(:))
Gold Query	ASK WHERE { ?x text:"first wife of" text:"Napoleon". ?x dbo:deathPlace res:France . }
Our Query	ASK WHERE { ?var dbo:abstract ?a. ?var dbo:deathPlace ?cmd1. ?a bif:contains '"first wife" AND "Napoleon"'. FILTER(CONTAINS(str(?cmd1),"France")) }
Gold A.	true
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

10. How old is James Bond in the latest Bond book by William Boyd?	
Entity	James Bond (res:James_Bond), William Boyd (res:William_Boyd)
Dependency Tree	>Bond : Bond (0|NNP) |=>How old : how_old (1|NN) |=>is : be (2|VBZ) |=>res:James_Bond(3|NNP|0) |=>in : in (4|IN) |==>book : book (5|NN) |===>the : the (6|DT) |===>latest : latest (7|JJS) |===>by : by (9|IN) |====>res:William_Boyd(10|NN)
Property Annotation	>Bond(0|NNP|0(:)) |=>HOW_OLD(1|NN|1(dbo:birthYear :)) |=>is(2|VBZ|0(:)) |=>res:James_Bond (3|NNP|1(res:James_Bond :)) |=>book(5|NN|3(dbo:author, dbp:book : yago:Book106410904)) |==>latest(7|JJS|0(dbo:lastAppearance :)) |==>William_Boyd(10|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { res:James_Bond dbo:lastAppearance ?x. ?x dbo:author res:William_Boyd_(writer). ?x text:"age of Bond" ?n . }
Our Query	SELECT DISTINCT ?uri WHERE { res:James_Bond dbo:lastAppearance ?var. ?var dbo:abstract ?a. ?a bif:contains '"Bond"'. ?var dbo:birthYear ?uri. ?var dbo:author ?cmd. FILTER(CONTAINS(str(?cmd),"William_Boyd"))}
Gold A.	45
Our A.	-
Accuracy	Recall = 0 Precision = 0 F-measure = 0

11. What eating disorder is characterized by an appetite for substances such as clay and sand?	
Entity	-
Dependency Tree	>characterized : characterize (0|VBN) |=>disorder : disorder (1|NN) |==>What : what (2|WDT) |==>eating : eating (3|JJ) |=>is : be (4|VBZ) |=>by : by (5|IN) |==>appetite : appetite (6|NN) |===>for : for (8|IN) |====>substances : substance (9|NNS) |=====>as : as (10|IN) |======>such : such (11|JJ) |======>sand :sand (12|NN) |=======>clay : clay (13|NN)
Property Annotation	>characterized(0|VBN|0(:)) |=>eating disorder(1|CombinedNN|1(res:Eating_disorder : yago:EatingDisorders)) |=>appetite(6|NN|1(res:Appetite :)) |==>substances(9|NNS|1(res:Substance : yago:Substance100019613)) |===>sand (12|NN|0(res:sand :)) |====>clay(13|NN|1(res:Clay :))
Gold Query	SELECT ?uri WHERE { ?uri rdf:type yago:EatingDisorders. ?uri text:"characterized by" "appetite for clay and sand"}
Our Query	SELECT ?uri WHERE { ?uri rdf:type yago:EatingDisorders. ?uri dbo:abstract ?a. ?a bif:contains '"characterized" AND "sand" AND "clay" AND "appetite" AND "substances"'}
Gold A.	res:Pica_(disorder)
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

12. What is the native city of Hollywood's highest-paid actress?	
Entity	Hollywood (res:Hollywood)
Dependency Tree	>What : what (0|WP) |=>is : be (1|VBZ) |=>city : city (2|NN) |==>the : the (3|DT) |==>native : native (4|JJ) |==>of : of (5|IN) |===>actress : actress (6|NN) |====>res:Hollywood(7|NN) |=====>'s : 's (8|POS) |====>highest-paid (9|JJ)
Property Annotation	>What(0|WP|0(:)) |=>is(1|VBZ|0(:)) |=>native city (2|CombinedNN|0(res:birthPlace :)) |==>actress(6|NN|1(res:Actor : yago:Actress109767700)) |===>Hollywood(7|NN|0(res:hollywood :)) |===>highest-paid (9|JJ|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"is" text:"Hollywood's highest-paid actress". ?x dbo:birthPlace ?uri . ?x rdf:type dbo:City . }
Our Query	SELECT DISTINCT ?uri WHERE { ?var dbo:abstract ?a. ?var dbo:birthPlace ?uri. ?a bif:contains '"Hollywood s highest-paid actress"'}
Gold A.	res:Los_Angeles
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

13. In which city does the presenter of the Xposé live?	
Entity	-
Dependency Tree	>live : live (0|VB) |=>in : in (1|IN) |==>does : do (2|VBZ) |===>city : city (3|NN) |====>which : which (4|WDT) |===>presenter : presenter (5|NN) |====>the : the (6|DT) |====>of : of (9|IN) |=====>the : the (11|DT) |=====>Xposé : Xposé (12|NNP)
Property Annotation	>live(0|VB|1(dbo:residence :)) |=>does(2|VBZ|0(:)) |==>city(3|NN|2(dbo:city, res:City : dbo:Settlement)) |==>presenter (5|NN|0(:)) |===>Xposé(10|NN|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { ?x text:"is" text:"former presenter of the Xposé girls". ?x dbo:residence ?uri}
Our Query	SELECT DISTINCT ?uri WHERE { ?var dbo:abstract ?a. ?a bif:contains '"presenter"'. ?var ?r ?cmd. FILTER(CONTAINS(str(?cmd),"Xpose")). ?var dbo:residence ?uri}
Gold A.	res:Dublin
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

14. Who plays Phileas Fogg in the adaptation of Around the World in 80 Days directed by Buzz Kulik?	
Entity	Phileas Fogg (res:Phileas_Fogg), Buzz Kulik (res:Buzz_Kulik)
Dependency Tree	>plays : play (0|VBZ) |=>Who : who (1|WP), res:Phileas_Fogg (2|NN), in : in (3|IN) |==>adaptation : adaptation (4|NN) |===>of : of (6|IN) |====>Around : around (7|IN) |=====>World : world (8|NN) |======>in : in (10|IN) |=======>Days : day (11|NNS) |========>80 : 80 (12|CD) |========>directed : direct (13|VBN) |=========>by : by (14|IN) |==========>res:Buzz_Kulik (15|NN)
Property Annotation	>plays(0|VBZ|1(dbo:starring :)) |=>Who(1|WP|0(: dbo:Agent)) |=>res:Phileas_Fogg(2|NN) |=>adaptation(4|NN|2(dbp:adaptation :)) |==>World(8|NN|1(res:World : yago:World102472987)) |===>80_Days(11|CombinedNN|1(res:80_Days :)) |====>directed(13|VBN|0(dbo:director :)) |=====>res:Buzz_Kulik(15|NN)
Gold Query	SELECT DISTINCT ?uri WHERE { ?x rdfs:label ?l. FILTER regex(?l,"Around the World in 80 Days","i") ?x dbo:director res:Buzz_Kulik. ?x dbo:starring ?uri. ?uri text:"as" text:"Phileas Fogg" . }
Our Query	SELECT DISTINCT ?uri WHERE { ?var dbo:director res:Buzz_Kulik. ?var dbo:starring ?uri. ?var ?r ?cmd.?var dbo:abstract ?a. ?a bif:contains '"Phileas Fogg" AND "adaptation"'. FILTER (contains(str(?cmd),"80 Days") and contains(str(?cmd),"World"))}
Gold A.	res:Pierce_Brosnan
Our A.	res:Julia_Nickson-Soul, res:Pierce_Brosnan, res:Eric_Idle
Accuracy	Recall = 1 Precision = 0.33 F-measure = 0.5

15. Who is the front man of the band that wrote Coffee & TV?	
Entity	Coffee & TV (res:Coffee_&_TV)
Dependency Tree	>Who : who (0|WP) |=>is : be (1|VBZ) |=>man : man (2|NN) |==>the : the (3|DT) |==>front : front (4|JJ) |==>of : of (5|IN) |===>band : band (6|NN) |====>the : the (7|DT) |====>wrote : write (8|VBD) |=====>that : that (9|WDT) |=====>res:Coffee_&_TV (10|NNP)
Property Annotation	>Who(0|WP|0(res:agent :)) |=>frontman(2|CombinedNN|1(:)) |==>band(6|NN|2(dbo:bandMember :)) |===>wrote(8|VBD|1(dbo:writer,dbo:musicalArtist :)) |====>res:Coffee_&_TV(10|NNP|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { res:Coffee_&_TV dbo:musicalArtist ?x. ?x dbo:bandMember ?uri. ?uri text:"is" text:"frontman".}
Our Query	SELECT DISTINCT ?uri WHERE { res:Coffee_&_TV dbo:musicalArtist ?var. ?var dbo:bandMember ?uri. ?uri dbo:abstract ?a. ?a bif:contains '"frontman"' }
Gold A.	res:Damon_Albarn
Our A.	Gold Answer ile aynı sonuçlar elde edildi.
Accuracy	Recall = 1 Precision = 1 F-measure = 1

16. Which Chinese-speaking country is a former Portguese colony?	
Entity	-
Dependency Tree	>is : be (0|VBZ) |=>country : country (1|NN) |==>Which : which (2|WDT) |==>Chinese-speaking : chinese-speaking (3|JJ) |=>colony : colony (4|NN) |==>a : a (5|DT) |==>former : former (6|JJ) |==>Portguese : Portguese (7|NNP)
Property Annotation	|=>country(1|NN|2(dbo:country, res:Country : yago:Country108544813 dbo:Country)) |==>Chinese-speaking(3|JJ|0(:)) |=>colony(4|NN|1(res:Colony :)) |==>former(6|JJ|0(:)) |==>Portguese(7|NNP|0(:))
Gold Query	SELECT DISTINCT ?uri WHERE { res:Chinese_language dbo:spokenIn ?uri. ?uri text:"is" text:"former Portuguese colony"}
Our Query	No Query
Gold A.	res:Macau
Our A.	No answer
Accuracy	Recall = 0 Precision = 0 F-measure = 0

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	: ATAÇ, Şenol
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 01.06.1989 Konya
Medeni hali	: Bekar
Telefon	: 0506 780 18 28
E-Posta	: satac@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Y. Lisans	TOBB ETÜ Bilgisayar Mühendisliği	2015
Lisans	TOBB ETÜ Bilgisayar Mühendisliği	2012

İş Deneyimi

Yıl	Yer	Görev
2012-2015	Tübitak Bilgem YTE	Araştırmacı
Mayıs-Ağustos 2012	Tübitak Bilgem YTE	Stajer
Eylül-Aralık 2010	Anel ARGE	Stajer
Ocak-Nisan 2010	TOBB ETÜ Bilişim Sistemleri	Stajer

Yabancı Dil

İngilizce (İyi)
Japonca (Başlangıç Seviyesi)

Yayınlar

M. Akif, Agca, Senol Atac, M. Mert Yucesan, Gokhan Y. Kucukayan, A. Murat Özbayoglu and Erdogan Dogdu. Opinion Mining of Microblog Texts on Hadoop Ecosystem. International Journal of Cloud Computing vol. in press. 2015