NAMED ENTITY DISAMBIGUATION USING LINKED OPEN DATA

Sherzod HAKIMOV

MASTER THESIS
COMPUTER ENGINEERING

TOBB UNIVERSITY OF ECONOMICS AND TECHNOLOGY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

NOVEMBER 2013
ANKARA

Fen Bilimleri Enstitü onayı

Prof. Dr. Necip CAMUŞCU
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

Doç. Dr. Erdoğan DOĞDU
Anabilim Dalı Başkanı

Şerzod HAKİMOV tarafından hazırlanan NAMED ENTITY DISAMBIGUATION USING LINKED OPEN DATA (VARLIK İSİMLERİNİN BAĞLI VERİLER KULLANILARAK (LINKED DATA) ANLAMLANDIRILMASI) adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Doç. Dr. Erdoğan DOĞDU
Tez Danışmanı

Tez Jüri Üyeleri

Başkan	: Yrd. Doç. Dr. Murat Özbayoğlu	
Üye	: Doç. Dr. Erdoğan Doğdu	
Üye	: Yrd. Doç. Dr. Gültekin Kuyzu	

THESIS STATEMENT

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.

I hereby confirm that all the information provided in this thesis was obtained with rules of ethical and academic conduct, the document is written in thesis format and that I have documented all sources used.

Sherzod Hakimov

Üniversite	**: TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitü	**: Fen Bilimleri Enstitüsü**
Ana Bilim Dalı	**: Bilgisayar Mühendisliği**
Tez Danışmanı	**: Doç. Dr. Erdoğan DOĞDU**
Tez Turu ve Tarihi	**: Yüksek Lisans Tezi – Kasım 2013**

Şerzod HAKİMOV

VARLIK İSİMLERİNİN BAĞLI VERİLER (LINKED DATA) KULLANILARAK ANLAMLANDIRILMASI

ÖZET

Doğal dilde yazılmış metinlerde geçen kişi, kuruluş, yer ve benzeri isimlerin belirlenmesi varlık isimlerinin tespit edilmesi (named entity recognition) olarak adlandırılır. Benzer veya aynı isme sahip farklı türde varlık isimlerinin olması belirsizlik oluşturur. Belirsizliğin çözülmesi için aday varlık isimlerinin içinden doğru olanı seçilmelidir yani anlamlandırılmalıdır. İnsanlar bu problemi çözerken okudukları metnin içeriğinden yararlanarak doğru kararı verirler. Yazılımlarla çözüm üretmek için metnin içeriğinin anlaşılması gerekir. Varlık isimlerinin anlamlandırılması (named entity disambiguation) için birçok araştırma yapılmış ve devam etmektedir. Son zamanlarda Wikipedia gibi ansiklopedi verilerinin kullanıldığı projelerin bu alan için elde ettikleri başarı oranı yüksektir. DBpedia, YAGO, Freebase gibi semantic veritabanlarının bu alan için katkı sağlayacağı üzerine araştırmalar yürütülmektedir. Bu tezde kapsamında, varlık ismi anlamlandırılması için bağlı veri (linked data) ve çizge merkezlilik algoritması kullanarak geliştirdiğimiz bir teknik sunulmaktadır. Geliştirdiğimiz sistemin adı NERSO'dur. Bu sistem açık veri kümeleri ile test edilerek değerlendirilmiş, ayrıca diğer varlık ismi tespiti ve anlamlandırılması projeleri ile karşılaştırılmış ve sonuçlar bu tezde sunulmuştur. Elde ettiğimiz sonuçlara göre NERSO diğer araçlara benzer ve daha iyi sonuçlar vermektedir.

Anahtar Kelimeler: Varlık isimleri, varlık isimlerinin anlamlandırılması, çizge merkezlilik algoritması, bağlı veri

University	:	**TOBB University of Economics and Technology**
Institute	:	**Institute of Natural and Applied Sciences**
Science Programme	:	**Computer Engineering**
Supervisor	:	**Associate Professor Dr. Erdogan DOGDU**
Degree Awarded and Date	:	**M.Sc. – November 2013**

Sherzod HAKIMOV

NAMED ENTITY DISAMBIGUATION USING LINKED OPEN DATA

ABSTRACT

Named entity recognition is the task of identifying named entities such as persons, organizations and locations in natural language texts. One of the problems in named entity recognition is that different entities having the same or similar names, that is the ambiguity problem. This problem requires a decision on choosing the right entity among a number of possible entities with the same or similar names. Human readers do this naturally when reading a text, because they are aware of the context. But, for the software to do this, a “named entity disambiguation” task needs to be executed to decide about the right entities. There are many techniques developed for named entity recognition and disambiguation tasks in the literature. A recent approach in these tasks is using open knowledge bases such as Wikipedia, and more recently the structured linked data counterparts that are derived from Wikipedia and similar knowledge bases, such as DBpedia, YAGO, and Freebase. In this thesis, we present a named entity disambiguation technique that is based on using Linked Open Data and a graph centrality algorithm for disambiguation. The system we developed is called NERSO. We evaluated our system using publicly available data sets, and compare its performance with other named entity recognition and disambiguation (or annotation) tools, and present the results in this thesis. Our results show that NERSO performs either similar or better than the other tools.

Keywords: Named entity, named entity disambiguation, graph centrality algorithm, Linked Data

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Assoc. Prof. Dr. Erdogan Dogdu for the patient guidance, encouragement and support throughout my time as his student.

I would like to thank Assist. Prof. Dr. Murat Özbayoğlu and Assist. Prof. Dr. Gültekin Kuyzu for their invaluable reviews on this thesis.

Finally, I would like to express my love and appreciation to my family for their support throughout my education.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Definition
LOD	Linked Open Data
NERSO	Named Entity Recognition and Disambiguation using Semantic Open Data
NER	Named Entity Recognition
NED	Named Entity Disambiguation
NEL	Named Entity Linking
OWL	Web Ontology Language
IE	Information Extraction

1 INTRODUCTION

Web documents and all other digital textual content are mainly produced in natural language for human reading. And the size of digital content on the Web is increasing every day. According to Google, Web pages in English are more than 25 billion today (based on the search of the word "the"). Searching Web content is therefore a difficult task. Current search and ranking techniques are mainly based on keywords using an inverted index. Keyword search is not specifically concerned with the meaning or semantics in text, but just the existence of words in the documents. However, the nature of Web search is changing, as the size of the Web is getting larger.

With the proliferation of digital content on the Web, it is important to automatically extract information from a huge collection of unstructured documents using software. "Information extraction"[1] is mainly concerned with extracting structured knowledge from unstructured documents. And, "named entity recognition"[2] is one of the subtasks in information extraction that is identifying "named entities" such as persons, organizations, locations, etc. in natural language texts. Named Entity Recognition (NER) is a term first coined at the sixth Message Understanding Conference (MUC6) (Grishman and Sundheim, 1996), which was hosted to encourage research for Information Extraction (IE) from unstructured texts (Zhang, 2013). NER is identified as one of the essential steps in information extraction to enable other information extraction tasks. For example, identifying entities is a first step in identifying relationships between entities and to extract structured knowledge from text, or identifying events related to entities, and so on. NER is also used in other applications and research areas such as question answering, semantic search, machine translation, etc. Despite its importance, NER is still an open research problem that is being studied, and there are a number of solutions for NER that are proposed in the literature.

[1]http://en.wikipedia.org/wiki/Information_extraction
[2]http://en.wikipedia.org/wiki/Named-entity_recognition

Another problem related to NER is the "named entity disambiguation" (NED) problem. "Entity names" in text may refer to more than one distinct entity. For example, "Washington" is a short name for the city of "Washington, DC" as well as the state of "Washington". This is called ambiguity, that is entity names referring to multiple entities, and it requires "named entity disambiguation" methods to resolve it.

1.1 Problem Definition

The ability to find the meaning of detected entities (person, location, etc.) is crucial. For instance, the word "Paris" can mean the city in one text and person in the other. The intended meaning can be extracted only from the context. The problem of named entity disambiguation can be defined as finding the intended meaning of entities presented in the text by analyzing the relation among all entities found in the text. For instance, the word "Texas" in the text "former Texas quarterback James Street" refers to University of Texas at Austin, in the text "in 2000, Texas released a greatest hits album" refers to British pop band. Also, the name "Michael Jordan" in the text "Michael Jordan and Karl Malone avoid eye contact in the 1998 NBA Finals (Andrew D. Bernstein/ Getty)" refers to Michael Jordan professional basketball player, in the text "Michael was the father of Ingres and Postgres, two relational database systems developed at Berkeley." refers to Michael Stonebraker the computer scientist. The tool should find the correct meaning of named entities that share the same name by mapping to encyclopedic databases such Wikipedia.

1.2 Our Contribution

Our contributions in this thesis are as follows:

- We present a new method to disambiguate named entities using Linked Data and a graph centrality algorithm. In this method the named entities are scored using the paths or relationships between the entities in the graph and then disambiguated.

- We evaluate our method extensively and compare the results with some well-known annotation tools' performances.

2 BACKGROUND AND RELATED WORK

This chapter first explains the background work about information extraction, semantic Web, RDF Data Model, Linked Data project and DBpedia ontology. Related work about named entity recognition and named entity disambiguation is explained in detail later. Finally, entity annotation systems are explained and a comparison of publicly available entity annotation systems is given in the last part of this chapter.

2.1 Information Extraction

Information extraction (IE) is a task to extract new information from structured, semi-structure or unstructured documents. IE can be considered as a sub discipline of artificial intelligence that focuses on human language texts by means of natural language processing. MUC (Message Understanding Conference) was one of the first conferences to encourage automatic information extraction from text documents. Some of the well-known IE tasks are given below:

- **Named Entity Recognition (NER)**: the task is to recognize proper names of persons, organizations, locations, etc. in texts. These proper names are called named entities.
- **Coreference resolution**: the task is to identify links between previously recognized named entities. It aims to identity if two named entities refer to the same person, location, and organization.
- **Relation extraction**: the task to identify and extract meaning from texts between identified named entities.
- **Named Entity Resolution**: the task is determining which actual person, organization, etc. the use of name refers to.

Recently, Wikipedia based IE has attracted a lot of attention because of its high quality data. Research on NER systems based on Wikipedia data has great success due to its coverage of data. Research in IE tasks lead to construction of knowledge bases that enable to query Wikipedia data. Knowledge bases such as DBpedia [5] and YAGO [6][7][8] exploit semi-structure part of Wikipedia pages such as infoboxes, categories, classes, etc. Such knowledge bases enable users to query encyclopedic data.

In the next section we will explain the Linked Data project that aims to combine all knowledge bases such DBpedia and YAGO and tries to provide links between them. Interlinked knowledge bases would enable users to query data from various sources and ontologies. Using structured data instead of unstructured documents makes it easier to find solution to IE tasks.

2.2 Semantic Web and Linked Data

The Semantic Web is a Web of Data — of dates and titles and part numbers and chemical properties and any other data one might conceive of. The collection of semantic Web technologies (RDF, OWL, SKOS, SPARQL, etc.) provides an environment where applications can query that data, draw inferences using vocabularies, etc. However, to make the Web of Data a reality, it is important to have a huge amount of data on the Web available in a standard format, reachable and manageable by semantic Web tools. Furthermore, not only does the semantic Web need access to data, but relationships among data should be made available, too, to create a Web of Data. This collection of interrelated datasets on the Web can also be referred to as Linked Data [1] [2].

Linked Open Data (LOD) is a project initiated to publish and connect structured data on the Web in a standardized way. The idea behind linked data is to publish large-scale semantic Web data sources that are interconnected with each other. The content of published data sources are made available to use freely using Semantic Web Technologies including RDF(S), OWL and SPARQL. All knowledge bases combined into one big Linked Data Cloud. This cloud has been growing rapidly in the recent years, and now consists of more than 300 datasets on various domains. The community of Linked Data encourages companies, individuals and organizations to publish their data freely in a standard format.

Table 1 Linked Data by domain [3]

Domain	Number of datasets	Triples	%	(Out-)Links	%
Media	25	1,841,852,061	5.82 %	50,440,705	10.01 %
Geographic	31	6,145,532,484	19.43 %	35,812,328	7.11 %
Government	49	13,315,009,400	42.09 %	19,343,519	3.84 %
Publications	87	2,950,720,693	9.33 %	139,925,218	27.76 %
Cross-domain	41	4,184,635,715	13.23 %	63,183,065	12.54 %
Life sciences	41	3,036,336,004	9.60 %	191,844,090	38.06 %
User-generated content	20	134,127,413	0.42 %	3,449,143	0.68 %
	295	31,634,213,770		503,998,829	

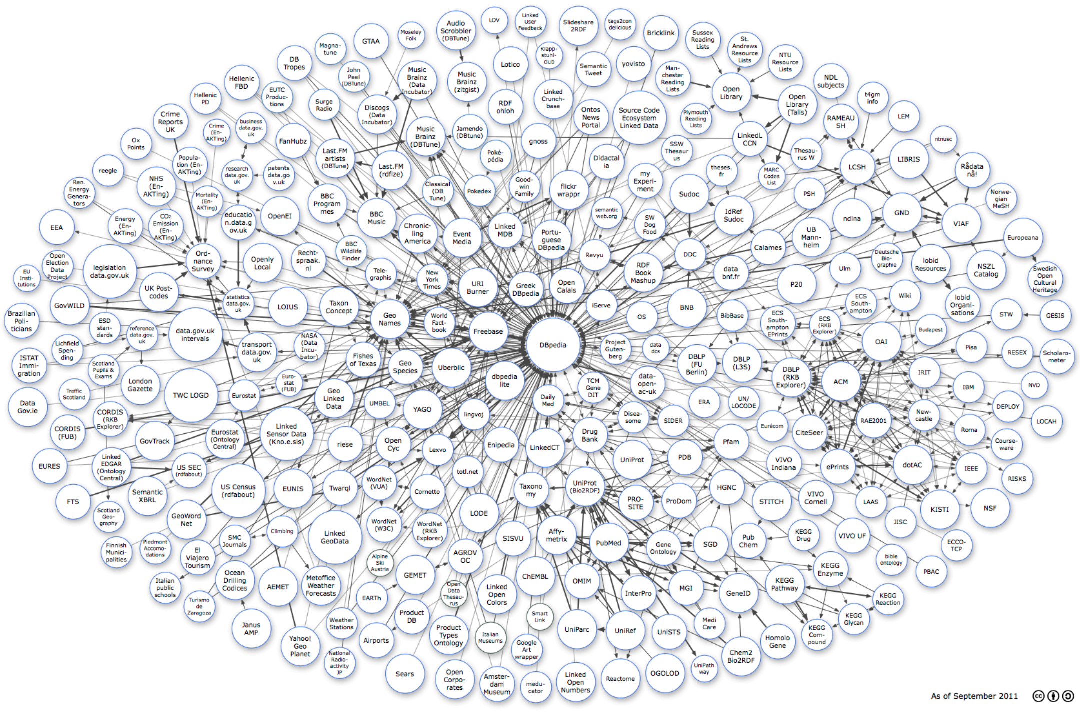


Figure 1 LOD Cloud [4]

A knowledge base is a special kind of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. A knowledge base is an information repository that provides a means for information to be collected, organized, shared, searched and utilized. It can be either machine-readable or intended for human use[3]. Knowledge bases contain information about concepts and links relationships between concepts.

LOD cloud consists of ontologies in various domains. Ontology is a representation of data in knowledge bases. Each ontology has its own specifications and definitions of data. Basically, ontology defines the structure of information in a knowledge base. It defines concepts, rules, specifications and relationships between concepts.

DBpedia [5] is one of the datasets that are available in the LOD. It is mainly produced from structured data in Wikipedia and presented in RDF format. DBpedia has a very central and prominent place in the linked data cloud; many datasets have links to DBpedia resources. It is getting more centralized and developing as a linking hub between other data sources in the LOD. For each entity, DBpedia defines a globally unique identifier that can be referenced over the Web in the RDF description of an entity.

The latest release of DBpedia ontology describes 3.77 million things including 764,000 persons, 573,000 places, 333,000 creative works (including 112,000 music albums, 72,000 films and 18,000 video games), 192,000 organizations (including 45,000 companies and 42,000 educational institutions), 202,000 species and 5,500 diseases.

YAGO [6][7][8] is another linked data knowledge base that contains more than 10 million entities, and more than 120 million facts about these entities. The information is derived from Wikipedia[4] and WordNET[5] and GeoNames[6]. YAGO ontology provides more specific facts about entities extracted from Wikipedia categories combined with WordNet.

[3] https://en.wikipedia.org/wiki/Knowledge_base
[4] http://en.wikipedia.org/
[5] http://wordnet.princeton.edu/
[6] http://www.geonames.org/

2.2.1 Data Model

The data model used in Linked Data Cloud is represented by Resource Description Framework (RDF)[7] statements.

RDF data model is specified by W3C (World Wide Web Consortium)[8] to model concepts and relationships between them in a standard format. RDF data model consists of RDF statements about concepts in subject-predicate-object (SPO) form also called as SPO triples. The subject denotes the resource in a statement, the predicate denotes the relationship between subject and object and the object denotes another resource or a literal text in a statement. For example, the RDF triple representation of the text "Michael Jordan was drafted by Chicago Bulls" as SPO expression is given below.

<Michael Jordan> <drafted by> <Chicago Bulls>

- Subject: Michael Jordan
- Predicate: drafted by
- Object: Chicago Bulls

Resources like people, organizations, locations and etc. are identified by unique addresses called URI (Uniform Resource Identifier)[9] in RDF statements, which enables to reuse the information with corresponding link to a knowledge base or a web page.

URI - is a string used to represent a resource in ontology. Such identification enables interlinking between resources from multiple sources, which is basically the idea behind LOD. URI in a specific ontology can refer to a unique resource or entity. Other URIs in other ontologies can identify the same entity. Each URI must be unique in a defined ontology.

---

7 http://en.wikipedia.org/wiki/Resource_Description_Framework
8 http://www.w3.org/
9 http://en.wikipedia.org/wiki/Uniform_resource_identifier

For example, http://dbpedia.org/resource/Michael_Jordan is a URI of Michael Jordan basketball player in DBpedia ontology.

http://yago-knowledge.org/resource/Michael_Jordan is URI of the same resource in YAGO ontology.

Resources in RDF triples are used with corresponding URIs. In DBpedia ontology the RDF triple of the above statement is:

- Subject: http://dbpedia.org/resource/Michael_Jordan
- Predicate: http://dbpedia.org/property/draftteam
- Object: http://dbpedia.org/resource/Chicago_Bulls

For the rest of the thesis, we will omit URI form of resources. Resources given in this work are DBpedia resources.

RDF statements are used to denote facts about entities. Literals can be used to as object of a statement to denote value representation about the subject. Literals are strings used to represent a value about the subject of a statement such as date, height, age etc. For instance, "Michael Jordan was drafted in 1984" can be represented as:

<Michael_Jordan> <draftyear> "1984".

RDFS - Resource Description Framework Schema[10] is an extension of RDF that allows superclass and subclass relationships. Also, literals can take values of XML schema data types such as xsd:date, xsd:year, etc.

Finally, "Michael Jordan was drafted by Chicago Bulls in 1984" can be represented as:

<Michael_Jordan> <draftteam> <Chicago_Bulls>.
<Michael_Jordan> <draftyear> xsd:yearˆ "1984".

[10] http://en.wikipedia.org/wiki/RDF_Schema

2.2.2 DBpedia Ontology

The DBpedia Ontology is open-domain ontology that has been created from infoboxes within Wikipedia. The most commonly used infoboxes are chosen and manually mapped into ontology. The latest release contains 359 classes, 800 object properties, 859 datatype properties, 116 specialized datatype properties. DBpedia dataset is available in 111 languages. English dataset contains 3.77 million things including 764,000 persons, 573,000 places, 192,000 organizations, etc.

Data model of DBpedia ontology is based on RDF triples such as:

> <Michael_Jordan> <birthPlace> <Brooklyn>.
> <Michael_Jackson> <birthYear> "1958".

The partial representation of the ontology as a graph is given below in Figure 2. Classes have subclass relation between them. Resource has type relation between classes. Predicates such "birthPlace" or "deathPlace" can be used between two resources as shown below.

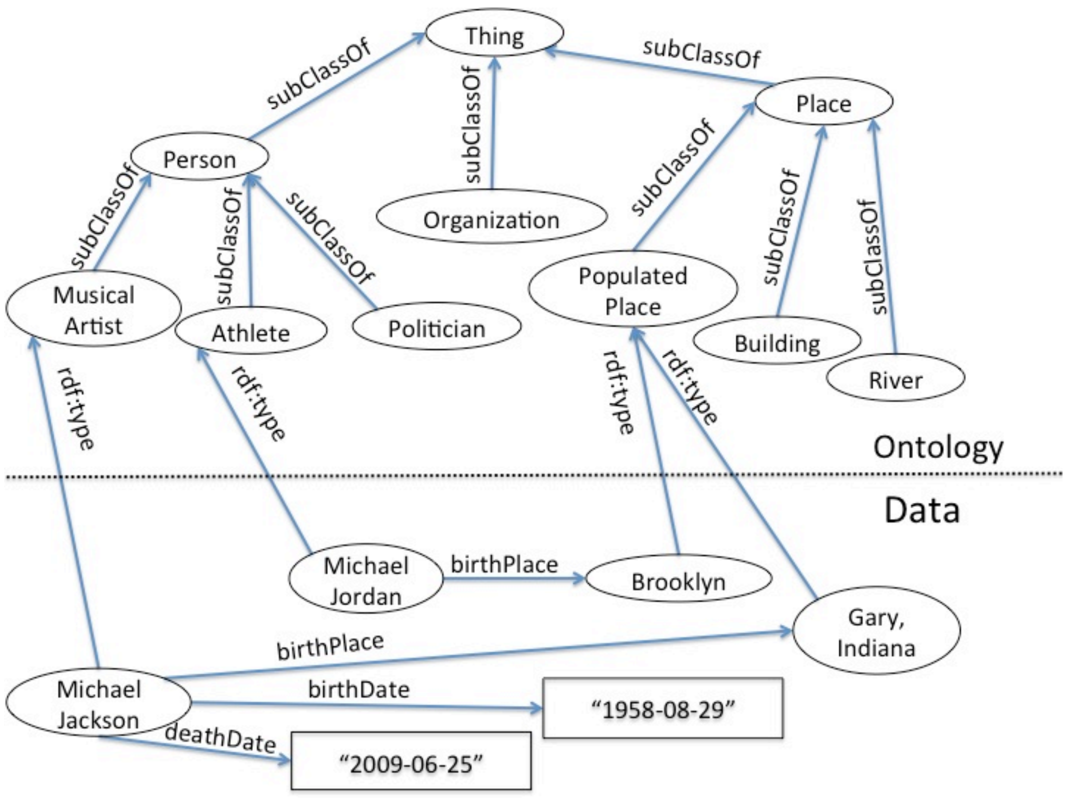


Figure 2 Part of DBpedia ontology

2.3 Named Entity Recognition

Named entity recognition (NER) is a task to extract words in a text that mean predefined categories such as person, location, organization and other categories with names. The term "named entity" was defined in the Sixth Message Understanding Conference (MUC -6)[9]. Research on NER is based on taking input a text and producing an output with annotations of defined categories. For example, the input such as:

Jim bought 300 shares of Acme Corp. in 2006.

is annotated with named entities below:

```
<PERSON>Jim</PERSON> bought <QUANTITY>300</QUANTITY> shares
of <ORGANIZATION>Acme Corp.</ORGANIZATION> in <DATE>2006
</DATE>.
```

State-of-the-art NER systems has high performance values for English texts. One of the best score was recorded in the Seventh Message Understanding Conference (MUC -7) with F1 score of 93.39%. Research on NER systems is generally based on grammatical or statistical model approaches. Grammar-based models obtain better results but lower recall rates as previous research projects suggest. Statistical models require large amount of data to train the system and extract some features that defines named entities [10].

Conference on Natural Language Learning 2003 (CoNLL-2003)[11] is a shared task that evaluated participated NER systems. Datasets include English and German news articles. English dataset contains articles from Reuters Corpus. German dataset contains articles taken from newspaper Frankfurter Rundshau. Evaluation results suggest that systems that use Maximum Entropy Models have higher performance. Other systems used Hidden Markov models, Conditional Markov models and hybrid approaches. The best performance was reached by [11].

The general output of NER systems is based on four categories: PERSON, LOCATION, ORGANIZATION and MISCELLANEOUS (MISC). MISC is used to annotate any entity that is not person, location or organization. More categories can be added to these four such as PERCENT, MONEY and DATE etc.

Stanford NER developed by Finkel et al. [12] is based on based on Conditional Random Field (CRF) classifier 0. Stanford NER classifies recognized named entities based on features extracted from data. The classifier is trained on both American and British texts that make the tool robust. The classifier can be trained on other domains too such as Twitter text, texts in other languages. The software is available online for demo with pre-trained model[12].

[11] http://www.cnts.ua.ac.be/conll2003/ner/
[12] http://nlp.stanford.edu:8080/ner/process

Illinois NER developed by Ratinov et al. 0 uses gazetteers from web sources and Wikipedia to extract non-local features for contextual aggregation, extended prediction history and two-stage prediction history. Non-local features are tokens of word around the candidate token that is likely to be a named entity. The intuition behind extracting non-local features is that the tokens around the candidate word are likely to be similar with tokens around words from the training data. The system is trained on CoNLL 2003 data.

A survey on named entity recognition and classification by Nadeau [41] provides essential information about the research in this area. The majority of studies are devoted to the study of English. Methods based on supervised learning require vast amount of annotated data to train the system. Annotating data in all domains is limited and rare. Recent studies in the field have shown that semi-supervised learning and unsupervised learning techniques are capable of providing better solution for entity types without annotated data in contrast to supervised learning techniques. Algorithms based on unsupervised learning methods rely on lexical resources such as WordNET, lexical patterns and statistics. Semi-supervised learning required a small supervision, a seed to start, to explore the context of the given text and try to find other related information.

2.4 Named Entity Disambiguation

Named entity disambiguation (NED) is a task to find the meaning of entities in a text and link to external knowledge bases. In some papers the task is also referred as Named Entity Linking (NEL). NED problem needs an external source to provide links for named entities. "Entity names" in text may refer to more than one distinct entity. For example, "Washington" is a short name for the city of "Washington, DC" as well as the state of "Washington". This is called ambiguity, that is entity names referring to multiple entities, and it requires "named entity disambiguation" methods to resolve it. For example, given the input text "I am going to <Paris> next week" we mean the city http://en.wikipedia.org/wiki/Paris by <Paris>, and not a person or the 2008 film with the same name.

NED is applied after NER in the information extraction process. NER is used to capture named entities from a list of named entity categories (person, location, etc.). Then, NED is applied to disambiguate and find the actual named entities from an inventory of entities. There are many methods developed to address NED problem as well. Zhang [24] divides these techniques into two categories: learning based and knowledge based methods. Learning based algorithms utilize machine-learning algorithms in both supervised and unsupervised approaches. Knowledge based algorithms utilize a knowledgebase to identify named entities, and also to disambiguate them. Earlier works in this category utilized WordNet and Wikipedia. Recent approaches started to use “linked data” resources such as DBpedia, Freebase and YAGO. These are huge web knowledge bases that include much more entities than earlier resources, but also include more information about the entities as well as relationships between the entities. Therefore, utilizing these knowledge bases is much more promising in NER and NED tasks.

The approaches to disambiguate named entities can be classified as (a) knowledge-based methods, (b) graph-based methods, and (c) hybrid methods such as algorithmic methods combined with crowdsourcing.

2.4.1 Knowledge-based Methods

A method based on Wikipedia data has been proposed by Bunescu & Pasca [15]. They use an article cosine similarity model and Support Vector Machine based on a taxonomy kernel using Wikipedia data.

Cucerzan [16] proposed another approach that also leverages Wikipedia data for identification and disambiguation of named entities. This work preprocesses the Wikipedia collection and extract more than 1.4 million entities with an average of 2.4 terms for each entity and 540 thousand (entity, category) pairs where category is the type of entity that is used to group Wikipedia entities. The knowledge extracted from Wikipedia is then used for the disambiguation process; the vector representation of the processed document is compared with the vector representation of the Wikipedia entities for disambiguation.

Milne & Witten [17] proposed to use Wikipedia articles and the hyperlinks between them to disambiguate named entities. A machine-learning approach based on Wikipedia hyperlinks is trained over millions of entities available. The disambiguation method is based on relatedness of a mention to the surrounding context. Their approach is to select a candidate entity that has more relations (hyperlinks) to other entities among candidate entities. The relatedness between candidates is calculated by taking into account the number of incoming and outgoing hyperlinks. This is similar to our approach but we use linked data counterpart of Wikipedia, that is DBpedia and we use a more sophisticated graph centrality algorithm.

Mihalcea & Csomai [18] proposed an approach to use Wikipedia data for automatic keyword extraction and word sense disambiguation. They show that knowledge-based approach combined with data-driven approach can provide promising results. Knowledge-based approach is based on contextual overlap of Wikipedia pages and definitions of ambiguous words in dictionaries. Data-driven approach is based on feature vector representation of ambiguous words in the given context.

2.4.2 Graph-based Methods

Graph-based methods are introduced to this area of research to analyze the relations between candidate entities and to disambiguate. In this approach graph nodes represent entities and the relations between entities are represented as edges between nodes. Different graph algorithm approaches exist here in determining the right entity among candidate entities for disambiguation.

Malin [25] proposed using random graph walk to disambiguate person names on Web pages. The intuition behind his disambiguation method is that the same names that co-occur with unambiguous names in the same Web pages tend to be the right names. While, Minkov et al. [26] also proposed to use random graph walk method to disambiguate names in emails.

Fernandez et al. [27] proposed to rank entities based on an entity scoring using the PageRank algorithm, which assigns a numerical weight to a node in a graph based on the number and numerical weight of other nodes that link to it. The numerical weight determines the rank of a node in a graph. A node that is linked by many high-ranking nodes is likely to receive a high rank itself. The graph of entities is built using nodes as entity names in news items and the edges between them represent the co-occurrence of entities in news articles. The scores they receive from PageRank algorithm rank entities in a graph. The highest-ranking entity is the disambiguated entity in the given context.

Gentile et al. [19], similar to our work, propose a graph-based model combining features from Wikipedia. They calculate the semantic relatedness over a graph to resolve the problem. In order to compute the semantic relatedness of two entities, they proposed a random graph walk model in combination with features extracted from Wikipedia like the title of a page, top n most frequent words, category words and words from outgoing links (hyperlinks) in a page. Our method on the other hand does not use any features of entities, yet only considers link relations between them. Additionally, we propose a graph centrality algorithm method for disambiguation process instead of a random-walk model.

2.4.3 Hybrid Methods

Methods based on solving disambiguation problem by leveraging crowdsourcing platforms have gained attention recently. CROWDMAP [28] is a model for ontology alignment that uses a crowdsourcing platform by assigning micro tasks to users. The model uses crowdsourcing to improve current automatic ontology alignment algorithm.

Finin et al. [29] introduce a method to annotate named entities in Twitter using crowdsourcing. Participated users are required to select person, location and organization names from presented tweets.

ZenCrowd [30] is a hybrid platform for entity linking using LOD. It takes HTML pages as input and runs algorithmic measures on all named entities. All matching LOD entities are assigned with some probability scores. The decision engine decides whether the probability score of all LOD entities for each named entity is sufficient and then the engine disambiguates named entities based on this decision. If the probability score is not deemed as excellent then users are presented with some information about LOD entities such as abstract pages, data properties, etc. for each named entity and asked to disambiguate named entity manually.

Damljanovic and Bontcheva [31] present an approach to use named entity recognition tool combined with linked data to disambiguate named entities. Their approach is also similar to our work. They use the named entity recognition tool ANNIE to identify named entities of the types PERSON, LOCATION and ORGANIZATION. Identified named entities are queried against DBpedia entities to find matching linked data entities. The scoring method to disambiguate is based on string similarity of identified named entities and DBpedia entities, structural similarity between all DBpedia entities that match recognized named entities and contextual similarity which is the probability of two entities appearing with the same words in the abstracts of DBpedia entities. All three scores define the disambiguation process and the highest score from these three approaches determines the disambiguated entity.

A survey on annotation systems by Andrews et al. [32] present comparable classification of annotation systems. All features of annotation systems are explained in detail. Hachey et al. [22] present full evaluation on three named entity linking systems and found out that co-reference and acronym handling could improve annotation results. A survey by Reeve & Han [34] examines Semantic Web annotation platforms and review architectures, approaches and performances of systems.

2.5 Entity Annotation Systems

Annotation is a process to attach names, attributes or relevant information to a document. Entity annotation systems try to fill the gap of ambiguity in natural language and provide solution using large datasets. These systems link identified items in a document to some encyclopedic data source by analyzing relations between all identified items. These identified items are called entities and the system that provides solution to ambiguity of words in a document is called entity annotation system.

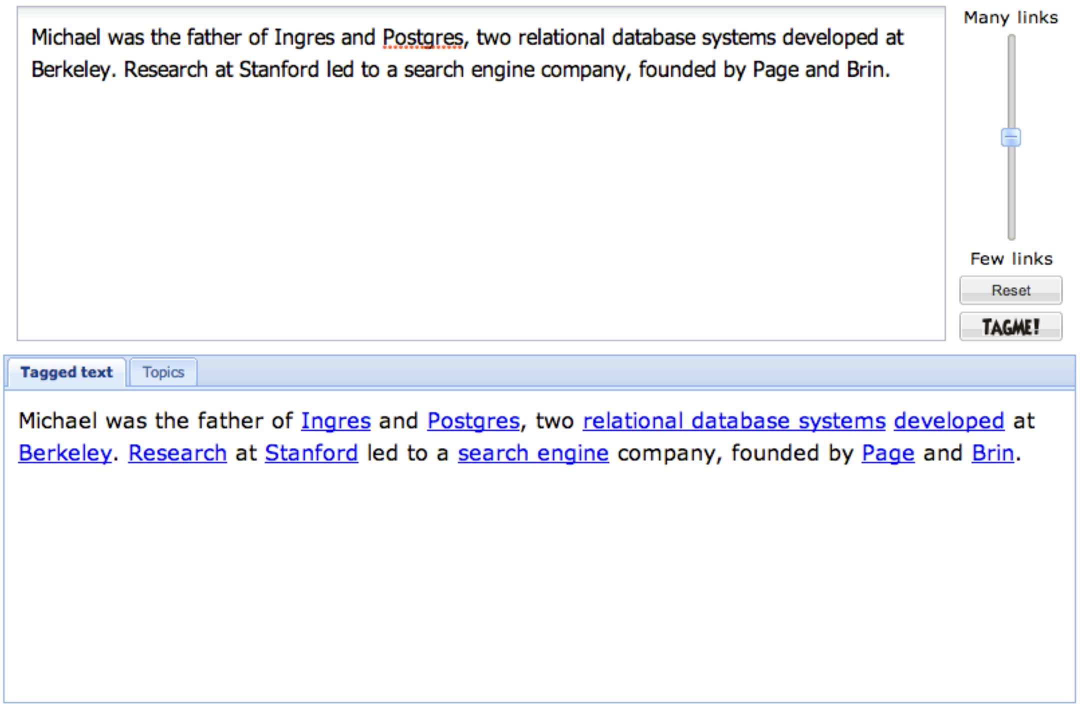
In this thesis we chose a number of publicly available entity annotation systems that we can test. The comparison of methods and data they use is given below.

Table 2 Comparisons of Entity Annotation Systems

Name	Recognition of named entities	Disambiguation method	Data source	Reference
TagMe	Wikipedia data	Collective disambiguation of detected entities	Wikipedia	[35]
Illinois Wikifier	Illinois NER	Support Vector Machine	Wikipedia	[14][36]
Wikipedia Miner	Wikipedia data	Support Vector Machine	Wikipedia	[17]
DBpedia Spotlight	DBpedia data	Vector Space Model of DBpedia entities	DBpedia	[37] [38]
Zemanta[13]	---	---	Wikipedia, etc.	---
AIDA	Stanford NER	Graph algorithm of mentions and linked data entities	YAGO	[39]
NERSO	Stanford NER	Centrality-based graph algorithm	DBpedia	[40]

[13] http://www.zemanta.com

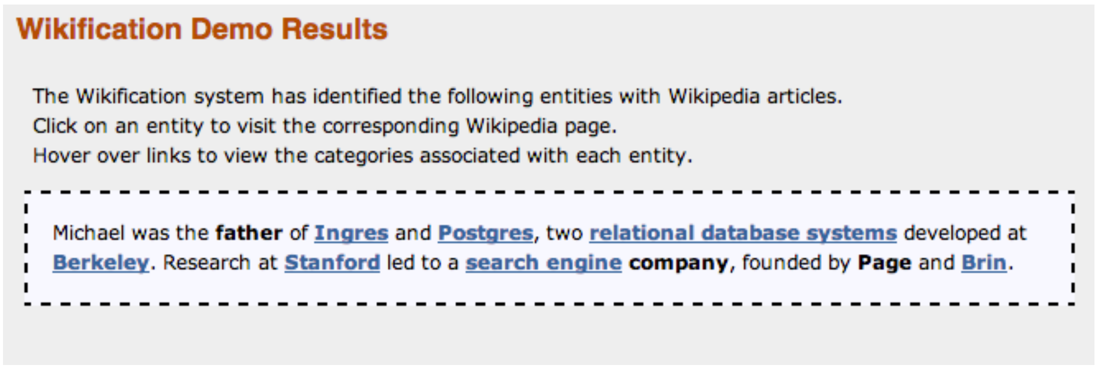
TagMe takes a text as input and searches for mentions using Wikipedia page titles, and redirect pages [35]. The disambiguation is based on calculating a score for each candidate entity of spotted named entities in a text. The score of each candidate entity is a sum of relatedness between all other candidate entities. In a sense, the disambiguation method is collective agreement between candidate entities of all spotted named entities.


Michael was the father of Ingres and Postgres, two relational database systems developed at Berkeley. Research at Stanford led to a search engine company, founded by Page and Brin.

Michael was the father of Ingres and Postgres, two relational database systems developed at Berkeley. Research at Stanford led to a search engine company, founded by Page and Brin.

Figure 3 TagMe sample output

Illinois Wikifier [36] is an annotation system based on recognizing named entities using Illinois NER tool [14]. The link structure in Wikipedia provides vital information about context as stated in the paper.

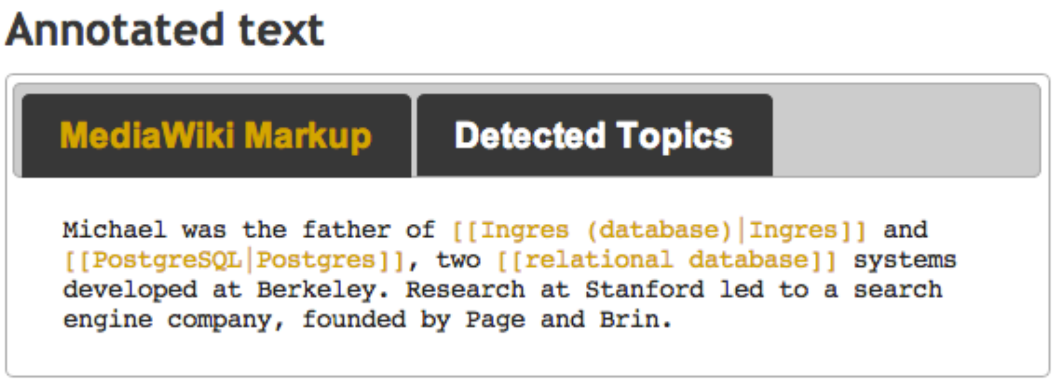

Wikification Demo Results

The Wikification system has identified the following entities with Wikipedia articles.
Click on an entity to visit the corresponding Wikipedia page.
Hover over links to view the categories associated with each entity.

Michael was the **father** of Ingres and Postgres, two relational database systems developed at Berkeley. Research at Stanford led to a search engine **company**, founded by **Page** and Brin.

Figure 4 Illinois Wikifier sample output

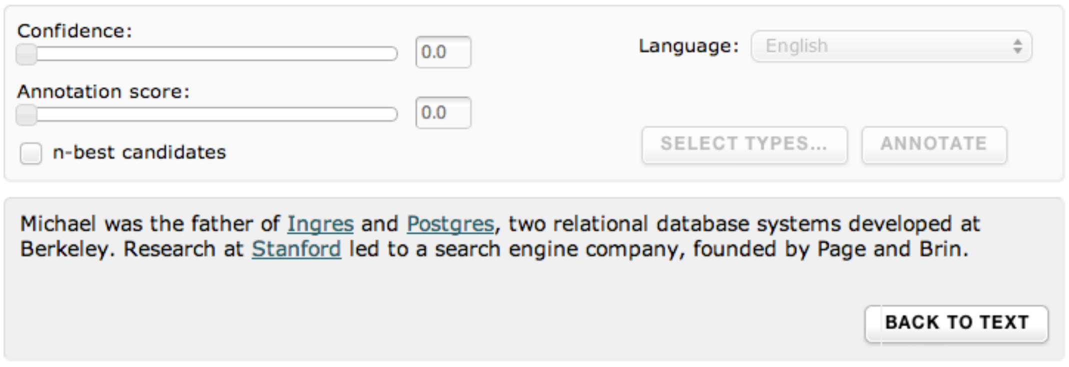
Wikipedia Miner [17] is one of the first systems to use Wikipedia for named entity disambiguation. This method is based on machine learning approach that trains the system by analyzing links in Wikipedia articles.



```
Michael was the father of [[Ingres (database)|Ingres]] and
[[PostgreSQL|Postgres]], two [[relational database]] systems
developed at Berkeley. Research at Stanford led to a search
engine company, founded by Page and Brin.
```

Figure 5 Wikipedia Miner sample output

DBpedia Spotlight [37] [38] is an annotation system that uses DBpedia as a knowledge base to disambiguate named entities. The disambiguation approach is to rank candidate entities according to the similarity score between their context info and the context surrounding a spotted entity. The similarity score is to calculate term frequency and inverse candidate frequency of words in the context of all DBpedia entities.

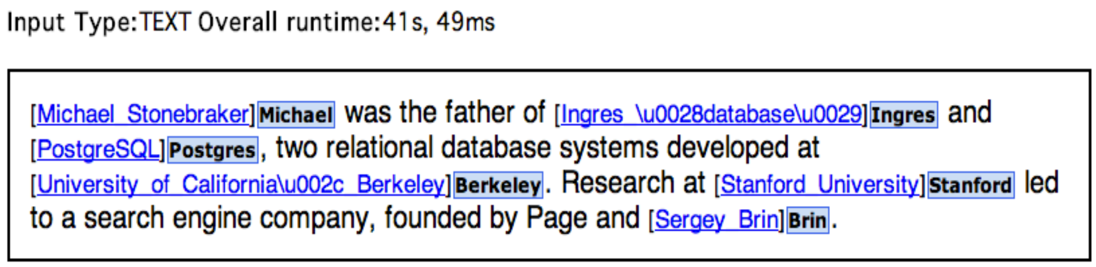


Michael was the father of Ingres and Postgres, two relational database systems developed at Berkeley. Research at Stanford led to a search engine company, founded by Page and Brin.

Figure 6 DBpedia Spotlight sample output

Zemanta is a content and links suggestion plugin for publishers, bloggers and other content creators. It is available online for annotating named entities, finding relevant tags for a given text. It suggests links to other relevant articles depending on the context.

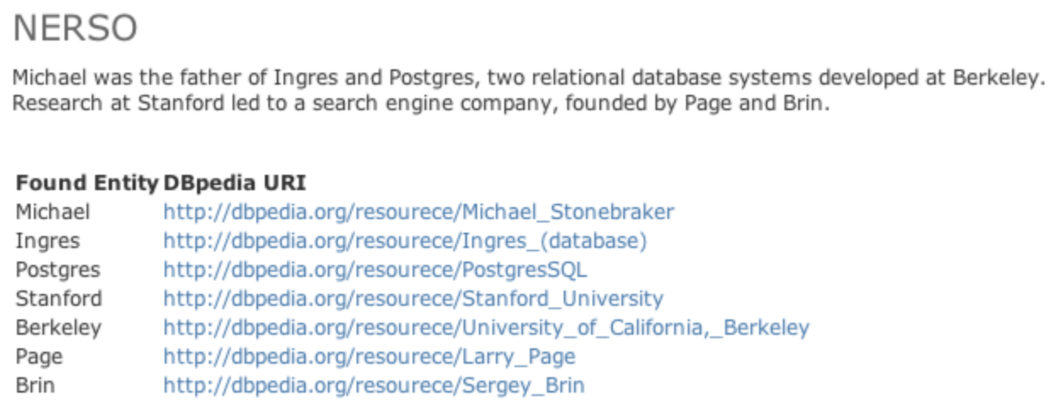
AIDA system [39] uses YAGO as a knowledge base. The recognition process of named entities is done using Stanford NER tool. The approach is based on joint mapping of mentions in the text and candidate entities from YAGO into a graph problem. Weighted edges between nodes in the graph are built on context similarity and coherence of entities and mentions.



Input Type:TEXT Overall runtime:41s, 49ms

[Michael_Stonebraker]**Michael** was the father of [Ingres_\u0028database\u0029]**Ingres** and [PostgreSQL]**Postgres**, two relational database systems developed at [University_of_California\u002c_Berkeley]**Berkeley**. Research at [Stanford_University]**Stanford** led to a search engine company, founded by Page and [Sergey_Brin]**Brin**.

Figure 7 AIDA sample output


NERSO

Michael was the father of Ingres and Postgres, two relational database systems developed at Berkeley. Research at Stanford led to a search engine company, founded by Page and Brin.

Found Entity	DBpedia URI
Michael	http://dbpedia.org/resourece/Michael_Stonebraker
Ingres	http://dbpedia.org/resourece/Ingres_(database)
Postgres	http://dbpedia.org/resourece/PostgresSQL
Stanford	http://dbpedia.org/resourece/Stanford_University
Berkeley	http://dbpedia.org/resourece/University_of_California,_Berkeley
Page	http://dbpedia.org/resourece/Larry_Page
Brin	http://dbpedia.org/resourece/Sergey_Brin

Figure 8 NERSO sample output

3 NERSO: NAMED ENTITY RECOGNITION AND DISAMBIGUATION USING SEMANTIC OPEN DATA

This chapter of the thesis contains details about the method that we developed to disambiguate named entities using DBpedia. Disambiguation method is explained in full detail later. Finally, the algorithm for the method is also provided in the last part of this chapter.

3.1 Method

NERSO is a semantic Web application we developed for entity annotation using linked data, specifically DBpedia. The application consists of the following steps to annotate a text:

1. **Spotting**

   The text is parsed using Stanford NER [12] to find named entities in a text. Recognized named entities are used to find the matching linked data (DBpedia) entities. For each named entity mention (named entity text), there could be more than one matching linked data entity (for example, “Washington” referring to the linked data entities about the State of Washington, President George Washington, Washington DC, etc.)

2. **Constructing the graph of entities**

   A graph is formed using the extracted named entities that are found in the previous step and the relationships (links) between these entities that exist in the linked data source.

3. **Disambiguation**

   The goal of this step to select a single linked data entity in the graph for each group of nodes that are found in the spotting step for specific named entity mentions. The selection process is based on a scoring function for each node in the graph.

3.1.1 Spotting

Stanford NER is a tool that recognizes named entities in a text. This tool outputs named entities with types such as PERSON, LOCATION, ORGANIZATION and MISC. MISC is used for any type that is not person, location or organization. For the text *"Michael was the father of Ingres and Postgres, two relational database systems developed at Berkeley. Research at Stanford led to a search engine company, founded by Page and Brin*." the tool provides the following output:

> Michael (PERSON) was the father of Ingres (MISC) and Postgres (MISC), two relational database systems developed at Berkeley (ORGANIZATION). Research at Stanford (ORGANIZATION) led to a search engine company, founded by Page (PERSON) and Brin (PERSON)

To find the matching DBpedia entities for the recognized mentions, NERSO uses 3 sources in DBpedia in this step:

a. **Data properties**

   We consider rdfs:label and other similar data properties such as name, firstName, lastName, officialName, givenName, birthName, commonName, and alias that might contain recognized entity's text. We chose a short list of data properties that express the title, name, or similar data. SPARQL query for extracting DBpedia entities using data properties is given in Figure 9.

b. **Redirect pages**

   The property *http://dbpedia.org/ontology/wikiPageRedirects* is a predicate in DBpedia that is used to show alternative titles of a given entity. These types of resources have no content itself, only redirects the reader to the base article. For example, "Edison Arantes Do Nascimento" redirects to "Pele", the famous football player. In this example, if the entity label text contains a surface form "Edison Arantes Do Nascimento" then the base page (redirect) "Pele" will be added to the candidate list of that particular named entity. SPARQL for extracting DBpedia entities using redirect pages is given in Figure 9.

c. Disambiguation pages

The property *http://dbpedia.org/ontology/wikiPageDisambiguates* is a predicate in DBpedia that is used to group entities that have various meanings for the same title. For example, “Michael_Jackson” and “Michael_Jordan” are grouped under “Michael_(disambiguation)” entity because they can both be referenced with a common title “Michael”. All entities grouped under these objects will be added to the candidate list of a named entity. SPARQL query for extracting DBpedia entities using disambiguation pages is given in Figure 10.

```
PREFIX dbont: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://www.w3.org/2002/07/owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT distinct ?s from <http://dbpedia.org>  WHERE {
        ?s rdfs:label  “+ searchText + “@en.
        }
        UNION {
                ?redirect dbont:wikiPageRedirects  ?s.
                ?redirect rdfs:label " + searchText + "@en.
        }
        UNION {
                ?s dbpedia-owl:alias " + searchText + "@en.
        }
        UNION {
                ?s dbpprop:alias  " + searchText + ”@en.
        }
        UNION {
                ?s  foaf:givenName   " + searchText + ”@en.
        }
        UNION {
                ?s  foaf:name  " + searchText + ”@en.
        }
        UNION {
                ?s  foaf:surname   " + searchText + ”@en.
        }
        UNION {
                ?s  dbpprop:commonName  " + searchText + ”@en.
        }
        UNION {
                ?s  dbpprop:name  " + searchText + ”@en.
        }
}
```

Figure 9 SPARQL query for data properties and redirect pages

```
PREFIX dbont: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://www.w3.org/2002/07/owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT distinct ?s from <http://dbpedia.org>  WHERE {
            ?dPage dbont:wikiPageDisambiguates ?s.
            ?dPage rdfs:label   " + searchText + ".
}
```
Figure 10 SPARQL query for disambiguation pages

In the following, we provide a formal description of our approach. Formally, let D be a text document, and LOD is a (part of) linked data graph such as DBpedia. Then,

Definition 1: $R_D = \{r_1, \ldots, r_n\}$ is the set of recognized named entities for D.

Definition 2: $V_r = \{v_i, \ldots, v_m\}$ is the set of linked data entities ($\in$ LOD) that match a named entity $r \in R_D$. For example, R_D = {“Michael”, “Ingres”, “Postgres”, “Berkeley”, “Stanford”, “Page”, “Brin”) is a set of named entities found by Stanford NER tool in document D. The matching DBpedia entities for these named entities are found using data properties, redirect and disambiguation pages are as follows:

$V_{Michael}$= { “Michael_Caine”, “Michael_Winner”, “Michael_Stonebraker”, ...}
V_{Ingres} = { “Ingres_database”}
$V_{Postgres}$ = {“PostgresSQL” }
$V_{Berkeley}$ = {“University_of_California,_Berkeley”, “Berkeley,_California”, …}
$V_{Stanford}$ = { “Stanford_University”, “Craig_Stanford”, …}
V_{Page} = {“Larry_Page”, “Anthony_Page”, “Jimmy_Page”, …}
V_{Brin} = {“Sergey_Brin”, …}

Definition 3: G (V_D, E_D) is the graph of all linked data (LOD) entities that match the named entities detected in D and their relationships in LOD. V_D is the union of all LOD entities, $V_D = \cup_r V_r$ for $\forall$ $r \in R_D$. E_D is the set of relationships (edges) between linked data entities that are recognized for D, $E_D = \{(v_i,v_j) \mid <v_i,p,v_j> \in E(LOD)$ for some $v_i,v_j \in V_D\}$. For example, $V_D = \{ V_{Michael} \cup V_{Ingres} \cup V_{Berkeley} \cup \ldots \cup V_n \}$ is the union of all named entities that exist in DBpedia for document D.

Definition 4: $V_s = \{(r, v_i) \mid v_i \in V_i, r \in R_D$ for $i=1\ldots r \}$ is the set of disambiguated linked data entities for each named entity. V_s is defined for our running example as follows:
V_s= {(Michael, Michael_Stonebraker), (Ingres, Ingres_database), (Postgres, PostgresSQL), (Stanford, Stanford_University), (Berkeley, University_of_California,_Berkeley), (Page, Larry_Page), (Brin, Sergey_Brin)}

3.1.2 Constructing Entity Graph

Wikipedia articles have hyperlinks to other articles, and these links are embedded in the text. These page links between articles are denoted in DBpedia with the property "wikiPageWikiLink". For example, DBpedia has the triple in the form of <Michael_Stonebraker, wikiPageWikiLink, Ingres_database>, which indicates that "Michael_Stonebraker" is related to "Ingres_database". In our approach, all links between spotted named entities are queried. For example, if "Michael_Stonebraker" and "Ingres_database" are spotted in the previous step, we search for links between these two entities using the "wikiPageWikiLink"property. If there exists a link between queried pairs, the edge is added between them in the graph. All edge weights are equal and set to 1.

Formally, E_D= {(Michael_Stonebraker, Ingres_database),... }is the set of links between DBpedia entities. As a result, the directed graph G is constructed from vertices V_D and edges E_D as shown in Figure 11.

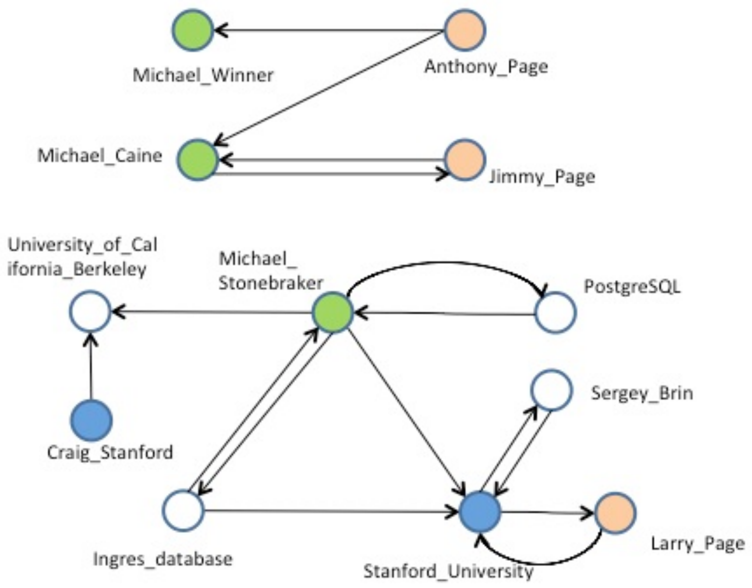


Figure 11 Graph of named entities

3.1.3 Disambiguation

Our disambiguation step is based on scoring each node in a graph with a scoring function explained below. The entity with the highest score is given as disambiguated DBpedia entity for the recognized entity from the previous step. All candidate entities of each recognized entity is added as nodes to a graph, and all links between them are added as edges in the previous step. Since edges are directed, outgoing and incoming links play an important role to score entities.

Centrality algorithms have been proposed to score the importance of a node in a graph. These measures provide information about how a node is central in a graph, which can suggest importance of the node. There are several different centrality algorithms that are widely used such as Degree, Betweenness, Closeness and PageRank. Degree[14] centrality is based on finding the adjacent links of a node. Betweenness[15] centrality is a measure of the number of times a node acts as a bridge

[14]http://en.wikipedia.org/wiki/Centrality#Degree_centrality
[15]http://en.wikipedia.org/wiki/Centrality#Betweenness_centrality

along the shortest path between two other nodes. Closeness[16] centrality is based on finding the overall length of shortest paths to all other nodes. PageRank[17] is introduced by Google to score all nodes in the graph-based on the idea that links to high-scoring nodes contribute more to the score of a node than equal links to low-scoring nodes.

The centrality algorithm we propose in this thesis is similar to Closeness centrality algorithm. It is based on finding reachable nodes in a graph, because the graph is directed and not all nodes are reachable. The sum of lengths of shortest distances to reachable nodes is calculated too. As shown in Figure 11, graph contains all candidates for each mention with all links between them. The graph in Figure 11 is partial representation of the graph of the example we use throughout the thesis.

Nodes in the graph are connected by directed edges that are extracted in the previous step. A node is more central than others if it has more incoming and/or outgoing edges than the others. In Figure 11, "Michael_Stonebraker", "Michael_Caine" and "Michael_Winner" are just three out of many entities for the recognized entity "Michael" in the sample text. The node "Michael_Stonebraker" in the graph has more links to the other entities. In this graph, "Michael_Stonebraker" has 4 outgoing and 2 incoming links in contrast to "Michael_Caine" and "Michael_Winner" with 1 outgoing and 2 incoming links, and 1 outgoing and 0 incoming link respectively. In our approach, these incoming and outgoing links play an important role in the disambiguation process. A node is central if it is connected with all other nodes in a graph with directed edges.

The overall scoring function is given below:

$$score(v) = inlink(v) \times outlink(v) \times cf(v) \times c(v)$$

where *inlink(v)* is the number or total weight of incoming links to the node *v*:

$$inlink(v) = \sum_{a \in V} w(a,v) \text{ s.t. } w(a,v) \in E_D; \; e \text{ is an edge from } a \text{ to } v.$$

[16]http://en.wikipedia.org/wiki/Centrality#Closeness_centrality
[17]http://en.wikipedia.org/wiki/PageRank

And, *outlink(v) is* the number of outgoing links from the node *v:*

$outlink(v) = \sum_{b \in V} w(v, b)$ s.t. *w(v,b)*∈ E_D; *e* is an edge from *v* to *b*.

$c(v)$ returns a constant value for the node v depending on how the node is found in the spotting step. If the node is found using data properties or redirect pages, the value is set to 3, and if the node is found using disambiguation pages then it is set to 1. This is done to lower the effect of the nodes found using disambiguation pages because data properties and redirect pages point the exact matches for the named entities.

$cf(v)$ returns the centrality factor of node *v*. It is calculated by dividing the total number of reachable nodes from node v to the sum of the lengths of the shortest paths to those reachable nodes. The shortest path if found using Dijkstra's shortest path algorithm. *cf(v)* is calculated as follows:

$$cf\ (v) = \frac{\partial_v}{\sum_{c \in V} s(v, c)}$$

where ∂_v is the number of reachable nodes of the node *v*, $\sum_{c \in V} s(v, c)$ is the sum of the lengths of shortest paths to all reachable nodes from the node *v*.

For example, the node "Michael_Stonebraker" has 6 reachable nodes (in Figure 11); the sum of shortest paths is 8. The node has 2 incoming and 4 outgoing links. The centrality factor of this node is 6/8. Then, *score*("Michael_Stonebraker") is 6 which is the product of *inlink*, *outlink*, *cf* and *c* values. The scores of all nodes in the graph are calculated as explained above. Then, the nodes are ranked based on this score. The node with the highest score is regarded as disambiguated named entity. As a result the node "Michael_Stonebraker" is the highest scoring node for the mention of "Michael".

The scoring function given above is based on multiplication of centrality factor of a node in a graph with count of incoming, outgoing links and constant value, which is assigned based on the DBpedia data used to spot that node. The centrality factor of a node is not enough for disambiguating based on the score. In the Figure 11, the node “Michael_Caine” has 1 outgoing and 2 incoming links and the node “Michael_Stonebraker” has 4 outgoing and 2 incoming links. The centrality factor of the node “Michael_Caine” will be 1 and the centrality factor of the node “Michael_Stonebraker” will be 6/8. However, it can be clearly seen that “Michael_Caine” is connected to a smaller cluster of nodes than “Michael_Stonebraker”. Using only centrality factor of the node is not enough to disambiguate named entities. Values such as count of incoming and outgoing links should be considered in the scoring function. The centrality factor given in this thesis is similar to Closeness centrality algorithm. The difference between two algorithms is that our centrality algorithm considers the number of reachable nodes that the node has. Closeness centrality algorithm considers only the length of path to the reachable nodes. The length of path to these reachable nodes does not provide the exact information about the other nodes in the graph. By considering both the number of reachable nodes and the length of path to these nodes we calculate the influence of the node on other nodes in the graph. For example, there must be a distinction between the node that has 4 reachable nodes with length of path to these nodes 8 and the node that has 8 reachable nodes with length of path to these nodes 8. Closeness centrality algorithm would provide the same influence values for both nodes. Our centrality algorithm would consider the second node more influential because it is connected to all its reachable nodes with 1 path length.

In this thesis, the given scoring function is multiplication of all values because it maximizes the score of the node that has many links. We have tried configurations of the scoring function. For example, adding the count of incoming and outgoing links and then multiplying it with other values results in less meaningful scoring. The node with 10 outgoing and 0 incoming links would have the same multiplication value with the node with 5 outgoing and 5 incoming links if add incoming and outgoing links and then multiply with other values. However, the second node is clearly more related with other nodes than the first node because it has both incoming and outgoing links. By multiplying incoming and outgoing links we maximize the chance of the node being selected that has many incoming and outgoing links. The best performance was achieved by multiplying all values. Therefore, in this thesis we decided to use the scoring function as a multiplication of all values explained above.

3.2 Disambiguation Algorithm

The overall disambiguation process is summarized in the following algorithm. The algorithm takes input a text document D and outputs V_s, the set of pairs of linked data entities with their corresponding named entities.

Algorithm 1 Disambiguation Algorithm using Linked Data

```
Input: D: a text document
Output: $V_s$: a set of pairs of linked data entities and corresponding named entities

    $G(V_D, E_D) = \{\}$
    $R_D$ = StanfordNER(D)      //extract named entities from D using Stanford NER
    for each $r$ in $R_D$
      $V_r$ = findDBpediaEntities($r$)      // find DBpedia entities that match $r$ (label etc.)
      $V_D = V_D \cup V_r$                  // add DBpedia entities to the entity graph G
    end
    for each pair of entities $v_1$ and $v_2$ in V
        if $\exists$ link $e$ from $v_1$ to $v_2$ in DBpedia
                $E_D = E_D \cup e(v_1,v_2)$
    end
    $G = (V_D, E_D)$
    for each vertex $v$ in G
        $v$.score = inlink($v$) * outlink ($v$) * cf($v$) * c($v$)
    end
    for each $r$ in $R_D$
        $V_s = V_s \cup (r, v_i)$ where $v_i$ is the highest-scoring vertex in $V_r$
    end
    return $V_s$
```

As explained above, the intuition behind the graph centrality algorithm is to find and select entities that have more links to other entities in the graph. Centrality algorithms are used to find the importance of a node in the graph, and it is based on how the node is linked to the other nodes that are also detected from the text. In other words, the nodes with more links to other nodes are within the context of the text and those that have no or less links are out of the context and therefore can be eliminated for disambiguation.

DBpedia provides the needed information to find candidate entities and constructing the graph of named entities with their relationships. Current version of the method treats all links between candidate entities as equal. The distinction between different object properties in linked data (links between entities) can be utilized to better detect the context among entities and therefore better disambiguate. It is left for future work.

4 EVALUATION

This chapter focuses on the evaluation of our approach and compare with other entity annotation systems. Evaluation metrics for checking correctness of entity annotation systems; public datasets used in the thesis and performance comparisons of entity annotation systems and our tool. Evaluation metrics to test correctness of annotation provided by each system are explained first. Then, datasets used to evaluate the systems are given. Finally, the performance result of each entity annotation system for each dataset is given later.

4.1 Evaluation Measures

Here we introduce evaluation measures to evaluate the performance of our approach against annotated datasets; precision, recall and F1.

4.1.1 Precision, Recall and F1

In information retrieval area precision and recall are defined in terms of retrieved and relevant documents. Retrieved documents are the list of documents that the tested systems return. Relevant documents are the list of documents that are relevant for a certain topic. Precision is computed as the ratio of retrieved documents to relevant documents. Recall is the fraction of documents that are relevant to the successfully retrieved documents. F1 is the harmonic mean of precision and recall values[18]. Gold standard of a dataset is the expected output from the systems. This evaluation metrics are used to test how relevant and successful is the output of the entity annotation systems when compared with gold standard provided by the tested dataset.

Definition 1: Let D be the set of documents, R ⊆ D be the subset of relevant documents for a query, S ⊆ D be the subset of retrieved documents for a query. Then, precision is the fraction of retrieved documents that are relevant.

$$Precision = \frac{|R \cap S|}{|S|}$$

[18] http://en.wikipedia.org/wiki/Precision_and_recall

Definition 2: Let D be the set of documents, R ⊆ D be the subset of relevant documents for a query, S ⊆ D be the subset of retrieved documents for a query. Then, recall is the fraction of documents that are retrieved successfully.

$$Recall = \frac{|R \cap S|}{|R|}$$

We need to check if annotation gold standard given by the dataset matches the output of the entity annotation system. We use the notion of *true positives* as the pairs of named entity and linked data entity that are included in both the output of the system and the gold standard of the dataset. *False positives* are the pairs of named entity and linked data entity that are included in the output of the system but not in the gold standard of the dataset. *False negatives* are the pairs of named entity and linked data entity that are included in the gold standard of the dataset but not in the output of the system. The overall calculation of precision and recall values is given below.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

F1 is the harmonic mean of precision and recall values. It is calculated as follows,

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

Micro average precision, recall and F1 score are calculated for datasets that include many documents. It is calculates as given below.

$$Micro\ Precision = \frac{\sum_{i=1}^{n}(true\ positives)}{\sum_{i=1}^{n}(true\ positives) + \sum_{i=1}^{n}(false\ positives)}$$

$$Micro\ Recall = \frac{\sum_{i=1}^{n}(true\ positives)}{\sum_{i=1}^{n}(true\ positives) + \sum_{i=1}^{n}(false\ negatives)}$$

$$Micro\ F1 = \frac{2 * \ Precision_{micro} * \ Recall_{micro}}{Precision_{micro} + Recall_{micro}}$$

4.2 Datasets

The datasets used in this thesis to test performance of entity annotation systems are listed in Table 3. All 3 datasets include well-written news articles in English and good punctuation. Each dataset contains annotations for each document, constituting the gold standard for each document as shown in Figure 12. AIDA/CoNLL dataset is the largest and most complete dataset; the other two contain fewer entity annotations but longer texts.

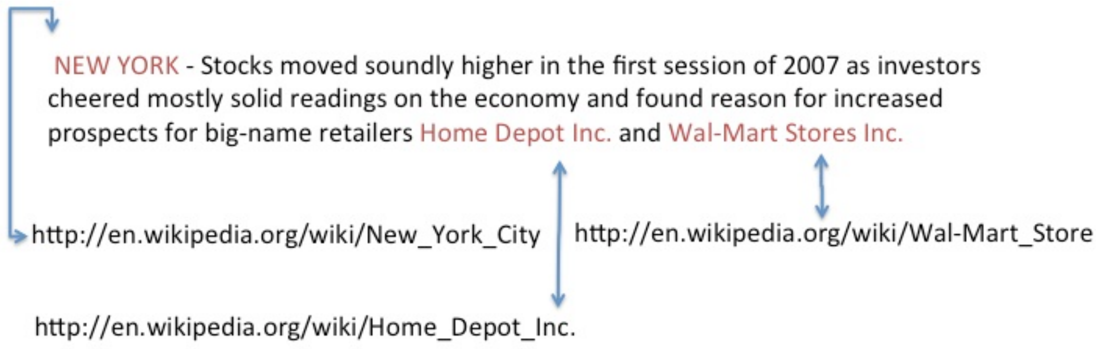

NEW YORK - Stocks moved soundly higher in the first session of 2007 as investors cheered mostly solid readings on the economy and found reason for increased prospects for big-name retailers Home Depot Inc. and Wal-Mart Stores Inc.
http://en.wikipedia.org/wiki/New_York_City
http://en.wikipedia.org/wiki/Wal-Mart_Store
http://en.wikipedia.org/wiki/Home_Depot_Inc.

Figure 12 Part of an example document in MSNBC dataset with its annotations to Wikipedia

Table 3 Public datasets for evaluation

Dataset	**Description**	**Average annotation per document**	**Number of texts**	**Average length**
AIDA/ CoNLL	Contains texts from Reuters news articles, which are part of entity recognition task in CoNLL 2003 [39]	20	1393	1130
MSNBC	Contains texts from MSNBC news articles. Contains 20 news articles for 10 different topics [16]	32.9	20	3316
AQUAINT	Contains texts from different news articles. Not all mentions in articles are annotated [17]	14.5	50	1415

4.3 Setting Up The Experiments

In this section we give brief information about test environment and software version of entity annotation systems tested using datasets explained above.

4.3.1 Software and APIs

The experiments have been done using the up-to-date version of each entity annotation system in August 2013. TagMe has been tested using publicly available API[19]; Wikipedia Miner has been tested using publicly available API[20]; DBpedia Spotlight has been tested using software that was installed on a local machine available at[21]; Zemanta has tested using publicly available API[22]. All these systems don't required any database or knowledge base to be installed to test. NERSO has been tested in August 2013 with DBpedia version 3.6 installed on a local computer. Since AIDA and Illinois Wikifier do not provide an API service, tools or source code to evaluate, we use results by Cornolti et al. [42].

4.3.2 Test Environment

TagMe, Wikipedia Miner, Zemanta were tested on a computer with Core 2 Duo 2.4 GHz microprocessor, 4 GB of RAM using APIs. The latest version of DBpedia Spotlight was installed on a local computer with Core 4 Quad Core 2.6 GHz microprocessor, 16 GB of RAM in August 2013. The system was tested using API that runs locally.

We installed DBpedia version 3.6 to test NERSO on a local computer with 2.4 GHz Core 2 Duo microprocessor, 8 GB of RAM. NERSO was tested on the same computer with TagMe, Wikipedia Miner and Zemanta.

[19] http://tagme.di.unipi.it/tagme_help.html
[20] http://wikipedia-miner.cms.waikato.ac.nz/services/?wikify
[21] http://spotlight.sztaki.hu/downloads/
[22] http://developer.zemanta.com/wiki/

The overall architecture to evaluate entity annotation systems is given in Figure 13. The documents in each dataset are given as an input to the system. The output of the system is disambiguated named entities with links to Wikipedia. The annotation of each document is provided with links to Wikipedia. The output of the systems and the annotation of the document are matched and the resulting precision, recall and F1 values are given as an output.

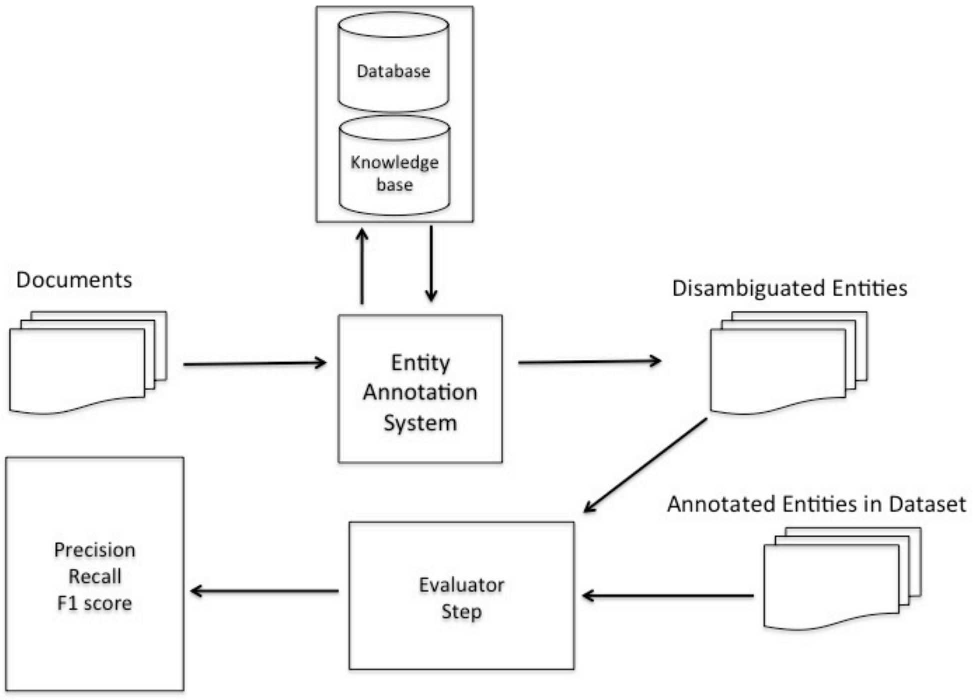


Figure 13 Architecture of Evaluation for Entity Annotation Systems

4.4 Experiment Results

This section gives performance of entity annotation systems for each dataset. We compared results for NERSO, TagMe, Wikipedia Miner, DBpedia Spotlight, AIDA, Illinois Wikifier and Zemanta.

The performance of systems listed in Table 2 is tested on three datasets listed in Table 3. Since AIDA and Illinois Wikifier do not provide an API service, tools or source code to evaluate, the results for the same datasets by Cornolti et al. [42] are given in separate columns to illustrate the performance rates. Results are micro average precision, recall and F1 values of entity annotation systems for each dataset.

TagMe outputs disambiguated named entities with assigned scores. The scores can be used to filter named entities by comparing to a threshold value, which is determined experimentally, choosing the threshold value that gives the best F1 value. Any returned entity with a score lower than the threshold could be discarded. As given in the paper [42] TagMe has higher F1, precision and recall values for AIDA/CoNLL dataset when the threshold value is set to 0.258. Likewise, for datasets MSNBC and AQUAINT the threshold was set to 0.188. The results given in Table 4, Table 5 and Table 6 are based on these threshold values.

Table 4 Evaluation results for AIDA/CoNLL dataset

	Our test results			**Test results in [42]**		
Name	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
NERSO	70	64	**66**	-	-	-
TagMe (t=0.258)	61	58	59	58	52	55
Wikipedia Miner	46	48	47	49	54	52
DBpedia Spotlight	62	64	63	31	40	35
Zemanta	27	34	30	-	-	-
Illinois Wikifier	-	-	-	49	60	54
AIDA – Cocktail party	-	-	-	72	34	47

Table 5 Evaluation results for MSNBC dataset

	Our test results			**Test results in [42]**		
Name	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
NERSO	64	52	**57**	-	-	-
TagMe (t=0.188)	45	40	43	50	51	51
Wikipedia Miner	34	41	37	54	39	46
DBpedia Spotlight	46	37	41	31	35	33
Zemanta	52	21	29	-	-	-
Illinois Wikifier	-	-	-	34	50	40
AIDA – Cocktail party	-	-	-	74	34	46

Table 6 Evaluation results for AQUAINT dataset

	Our test results			**Test results in [42]**		
Name	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
NERSO	18	16	17	-	-	-
TagMe (t=0.188)	26	31	29	47	54	50
Wikipedia Miner	28	47	**36**	36	65	46
DBpedia Spotlight	22	25	23	31	40	35
Zemanta	20	19	20	-	-	-
Illinois Wikifier	-	-	-	28	42	34
AIDA – Cocktail party	-	-	-	35	15	25

Based on results in Table 4 NERSO has the highest F1 score of 66%, the highest precision score of 70% and the highest recall score of 64%. Our test results indicate that the second best performed system is DBpedia Spotlight. NERSO has high precision due to **Spotting** step of the method. This step takes named entities as an input from Stanford NER, which recognizes named entities with high precision as well. After recognition we leverage DBpedia data to find all possible candidates from DBpedia for each named entity. After constructing graph of DBpedia entities we score each node in the graph with scoring function explained in 3.1.3. NERSO is capable of disambiguating named entities with high precision and recall due to Stanford NER, DBpedia data used to spot DBpedia entities for each named entity. Also, the methods to find the right DBpedia entities and construct the graph is important to disambiguate named entities based on the context of the given text. Finally, the disambiguation method that is based on the score of graph centrality algorithm, incoming and outgoing links enables NERSO to disambiguate named entities with promising results as given proven by evaluation results.

Previous tests by Cornolti et al. [42] and our tests suggest that TagMe is capable of disambiguating named entities better than AIDA, Illinois Wikifier, Zemanta and Wikipedia Miner. Our test results show that the latest version of DBpedia Spotlight has improved dramatically. The previous F1 score was 35% and the new F1 score is 63%. The difference between two scores can be explained as the new version handles disambiguation of named entities better. The previous F1 score was obtained using public API of DBpedia Spotlight in 2012. We performed our tests in August 2013. Wikipedia Miner and TagMe have similar evaluation results when compared our test results with previous tests by Cornolti et al. [42]. Zemanta is a system that selects the most important named entities, concepts from the given text and provides them as tags for that text. Zemanta is not the best project for named entity disambiguation. It has evaluation results such as precision score of 27%; recall score of 34% and F1 score of %30.

Based on result in Table 5, NERSO has the highest F1 score, recall and precision. Our test results for TagMe are different from previous tests by Cornolti et al. [42]. Also, DBpedia Spotlight and Wikipedia Miner have different evaluation results than previous tests. It can be explained as different versions, the time of testing. As we explained above, we expect DBpedia Spotlight to perform better from previous version, which is clearly shown by our test results. The high precision score of Zemanta suggest that it provides results that are more relevant with the given annotation of the dataset. It doesn't annotate all named entities in the text but the ones that it annotates are more relevant when we compare with other entity annotation systems. Also, previous tests by Cornolti et al. [42] suggest that AIDA has very high precision, which is 74%. AIDA annotates more relevant named entities than any other system as suggest by the results in Table 5.

Based on results in Table 6, NERSO is one of the worst systems for AQUAINT dataset because the annotation of the dataset provided by publishers is incomplete and not all named entities are included. Identified named entities by NERSO are not included in general and the provided annotation is not based on only named entities. It includes concepts, entities and categories that have corresponding Wikipedia page. For example, the word "cancer" is not considered as a named entity but it is included in the annotation provided by the publishers. Wikipedia Miner is the best performing systems because the system aims to provide links for concepts, entities and named entities that have Wikipedia page. Our test results for TagMe and DBpedia Spotlight are different from previous tests by Cornolti et al. [42].

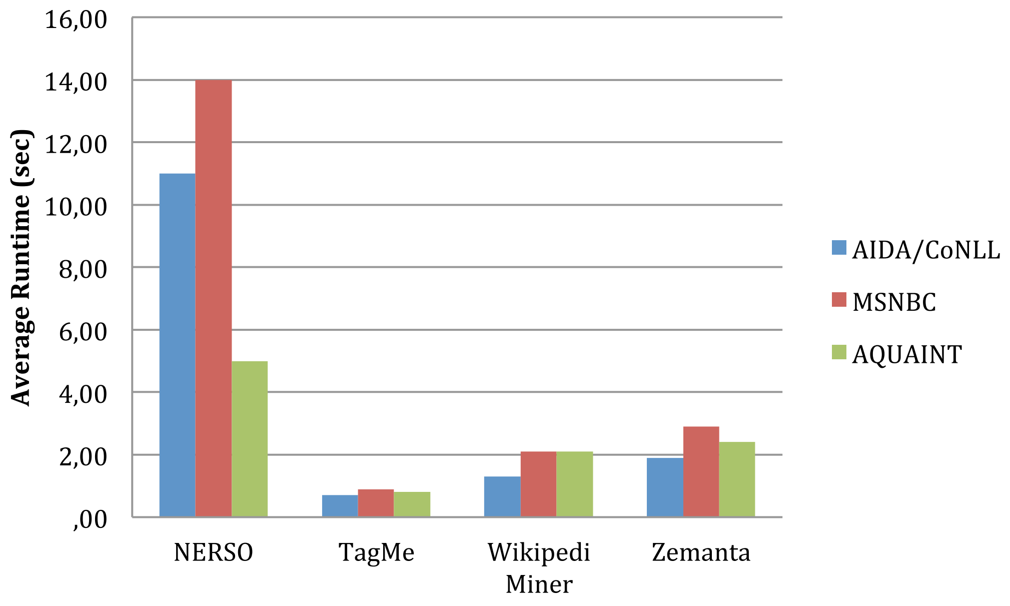


Figure 14 Average runtime of entity annotation systems

Figure 14 lists the average runtime for each document in three data sets. NERSO has the worst running time among compared systems, and TagMe has the best average runtime performance. In contrast to other annotation systems, NERSO runs slower because constructing graph of named entities take much longer time. The current version of NERSO first extracts all links of found DBpedia entities. Then, it searches for overlapping DBpedia entities found in each other's list of links. This step takes the most runtime of the system. The number of named entities and the number of candidate DBpedia entities affect the performance because the more entities mean more links to extract. The effect is consistent with runtime of NERSO for three datasets. The more annotations documents include the more time it takes for NERSO to annotate. AQUAINT has average of 14.5 annotations per document and NERSO annotates each document with average runtime of 5 seconds. AIDA/CoNLL has average of 20 annotations per document and NERSO annotates each document in 11 seconds in average. MSNBC has average of 32.9 annotations per document and NERSO annotates each document in 14 seconds in average. DBpedia Spotlight is not included because the current version of Spotlight runs on a local machine only and runs standalone and therefore there is no network delay and competition for resources where other systems are remotely accessed via an API.

5 CONCLUSION

In this thesis, we presented NERSO, a named entity detection and disambiguation system that is based on detecting entities in text, constructing a graph of linked data entities, and their relationships from linked data, and disambiguating the entities using a graph centrality algorithm. The graph centrality algorithm resolves ambiguities by scoring entities based on their relationships to the other entities in the text and then selecting the highest scoring entities among entities to be disambiguated.

We have compared our system's performance to other similar work, namely TagMe, Wikipedia Miner, DBpedia Spotlight, Zemanta, Illinois Wikifier, and AIDA using three well-known data sets, namely AIDA/CoNLL, MSNBC, and AQUAINT. Our tests show that NERSO frequently performs better than other systems with up to 70% precision. In contrast to other annotation systems, NERSO runs slower because constructing graph of named entities take much longer time. As a future work, the graph of named entities can be constructed using parallel computing and querying of named entity relations.

It is clear that with the proliferation of LOD, linked data based named entity recognition and disambiguation methods will be more effective and more work is expected in this area. Linked data based annotation and disambiguation is used in many Web based applications today, and there will be more work especially in search engines and related applications.

For future work, analyzing and assigning different edge weights in the graph can improve the graph centrality algorithm. Some of the edges (relations) between named entities can have more influence on the graph. The graph centrality algorithm would produce different scores for named entities that are strongly connected through high-weighted edges. The current algorithm assigns equal weight to all edges. Also, the surrounding context around named entities can be considered as a research direction for named entity disambiguation. Entity categories provided by NER systems can be used to filter out some named entities, which can improve runtime performance. Finally, combining linked data based techniques with natural language processing and possible improvements explained above are some of the possible research directions that need to be investigated.

6 REFERENCES

[1] C. Bizer, T. Heath and T. Berners-Lee, “Linked data-the story so far” International Journal on Semantic Web and Information Systems (IJSWIS), 5(3), 1-22

[2] http://www.w3.org/standards/semanticweb/data

[3] C. Bizer, A. Jentzsch, R. Cyganiak, “State of the LOD Cloud. Version 0.3” 09/19/2011. http://lod-cloud.net/state

[4] R. Cyganiak and A. Jentzsch. “Linking Open Data cloud diagram.” http://lod-cloud.net/

[5] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann, “DBpedia - A crystallization point for the Web of Data.” Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 7(3), 154-165, 2009.

[6] F. M. Suchanek, G. Kasneci, G. Weikum, “YAGO - A Core of Semantic Knowledge.” WWW 2007.

[7] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis Kelham, G. de Melo, G. Weikum, “YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages.” WWW 2011

[8] J. Biega, E. Kuzey, F. M. Suchanek, “Inside YAGO2s: A Transparent Information Extraction Architecture.” WWW 2013.

[9] R. Grishman, B. Sundheim, “Message Understanding Conference - 6: A Brief History.” In Proc. International Conference on Computational Linguistics, 1996.

[10] http://en.wikipedia.org/wiki/Named-entity_recognition

[11] R. Florian, A. Ittycheriah, H. Jing and T. Zhang, “Named Entity Recognition through Classifier Combination.” In: Proceedings of CoNLL-2003, Edmonton, Canada, 2003, pp. 168-171.

[12] J. R. Finkel, T. Grenager, and C. Manning, “Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling.” In Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370.

[13] J. Lafferty, A. McCallum and F. CN. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data.", 2001.

[14] L. Ratinov and D. Roth, “Design Challenges and Misconceptions in Named Entity Recognition.” CoNLL, 2009.

[15] R. Bunescu and M. Pasca, “Using encyclopedic knowledge for named entity disambiguation.” In Proceedings of EACL. Vol. 6, pp. 9–16. 2006.

[16] S. Cucerzan, “Large-scale named entity disambiguation based on Wikipedia data.” In Proceedings of EMNLP-CoNLL (pp. 708–716). 2007.

[17] D. Milne, and I. Witten, “Learning to link with wikipedia.” In Proceedings of the 17th ACM conference on Information and knowledge management. 2008.

[18] R. Mihalcea and A. Csomai, “Wikify!: linking documents to encyclopedic knowledge.” In CIKM (Vol. 7, pp. 233–242). 2007.

[19] A. Gentile, Z. Zhang and L. Xia, “Graph-based semantic relatedness for named entity disambiguation.” In Proceedings of International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES. 2009

[20] L. Reeve and H. Han,”Survey of semantic annotation platforms.” In Proceedings of the 2005 ACM symposium on Applied computing (pp. 1634-1638). ACM 2005.

[21] P. Andrews, I. Zaihrayeu and J. Pane, “A classification of semantic annotation systems.” Semantic Web, 3(3), 223-248., 2012

[22] B. Hachey, W. Radford and J. R. Curran, “Graph-based named entity linking with wikipedia.” In Web Information System Engineering– WISE 2011 (pp. 213-226), 2011.

[23] D. Nadeau and S. Sekine, “A survey of named entity recognition and classification.” Lingvisticae Investigationes, 30(1), 3-26, 2007.

[24] Z. Zhang, “Named Entity Recognition – Challenges In Document Annotation, Gazetteer Construction And Disambiguation.” PhD Thesis.The University of Sheffield, 2013.

[25] B. Malin, “Unsupervised name disambiguation via social network similarity.” In Workshop on link analysis, counterterrorism, and security (Vol. 1401, pp. 93-102), 2005.

[26] E. Minkov, R. C. Wang and W. W. Cohen, “Extracting personal names from email: applying named entity recognition to informal text.” In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 443- 450). Association for Computational Linguistics, 2005.

[27] N. Fernández, J. M. Blázquez, L. Sánchez and A. Bernardi, “Identityrank: Named entity disambiguation in the context of the news project.” In The Semantic Web: Research and Applications (pp. 640-654), 2007.

[28] C. Sarasua, E. Simperl and N. Noy, “Crowdmap: Crowdsourcing ontology alignment with microtasks.” In *Proceedings of 12th Internation Semantic Web Conference –ISWC, 2012*

[29] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau and M. Dredze, “Annotating named entities in Twitter data with crowdsourcing.” In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (pp. 80-88), 2010

[30] G. Demartini, D. E. Difallah and P. Cudré-Mauroux, ”ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking.” In *Proceedings of the 21st international conference on World Wide Web* (pp. 469-478). ACM, 2012.

[31] D. Damljanovic and K. Bontcheva, “Named entity disambiguation using linked data” In *Proc. of the 9th Extended Semantic Web Conference (ESWC), 2012.*

[32] P. Andrews, I. Zaihrayeu and J. Pane, “A classification of semantic annotation systems.” *Semantic Web*, *3*(3), 223-248, 2012.

[33] B. Hachey, W. Radford, J. Nothman, M. Honnibal and J. R. Curran, “Evaluating entity linking with Wikipedia.” *Artificial intelligence*, *194*, 130-150, 2013.

[34] L. Reeve and H. Han, “Survey of semantic annotation platforms.” In *Proceedings of the 2005 ACM symposium on Applied computing* (pp. 1634-1638). ACM, 2005.

[35] P. Ferragina and U. Scaiella, “TAGME: on-the-fly annotation of short text fragments (by wikipedia entities).” In *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 1625-1628). ACM, 2010.

[36] L. Ratinov, D. Roth, D. Downey and M. Anderson, “Local and Global Algorithms for Disambiguation to Wikipedia.” In *ACL* (Vol. 11, pp. 1375-1384), 2011.

[37] 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* (pp. 1-8). ACM, 2011.

[38] J. Daiber, M. Jakob, C. Hokamp and P. N. Mendes, “Improving Efficiency and Accuracy in Multilingual Entity Extraction.” In *Proceedings of the 9th*

International Conference on Semantic System, 2013

[39] 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* (pp. 782-792).Association for Computational Linguistics, 2011

[40] 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 (p. 4). ACM 2012

[41] D. Nadeau and S. Sekine, “A survey of named entity recognition and classification.” *Lingvisticae Investigationes*, *30*(1), 3- 26, 2007.

[42] M. Cornolti, P. Ferragina and M. Ciaramita, “A framework for benchmarking entity-annotation systems.” In *Proceedings of the 22nd international conference on World Wide Web* (pp. 249-260), 2013.

CURRICULUM VITAE

Personal Details

Surname, name:	HAKIMOV, Sherzod
Citizen:	TURKMENİSTAN
Birthdate, place of birth:	07.05.1988 TURKMENİSTAN
Marital status:	Single
Phone:	0 (506) 522 50 89
E-mail:	sherzodhakimov@gmail.com

Education

Degree	**University**	**Graduation date**
Bachelor of Science	TOBB ETÜ	2010

Professional Experience

Year	**Institution**	**Job**
2011-	TOBB ETÜ	Research assistant

Foreign Languages

English

Russian

Turkish, Spanish, Uzbek

Publications

- **S. Hakimov**, H. Tunc, M. Akimaliev and E. Dogdu, "Semantic question answering system over linked data using relational patterns." In Proceedings of the Joint EDBT/ICDT 2013 Workshops (pp. 83-88). ACM 2013
- **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 (p. 4). ACM 2012.