Named Entity Recognition and Disambiguation using Linked Data and Graph-based Centrality Scoring

Sherzod Hakimov
TOBB University of Economics and Technology
Computer Engineering
Ankara, Turkey
hakimov@etu.edu.tr

Salih Atilay Oto
TOBB University of Economics and Technology
Computer Engineering
Ankara, Turkey
saoto@etu.edu.tr

Erdogan Dogdu
TOBB University of Economics and Technology
Computer Engineering
Ankara, Turkey
edogdu@etu.edu.tr

ABSTRACT

Named Entity Recognition (NER) is a subtask of information extraction and aims to identify atomic entities in text that fall into predefined categories such as person, location, organization, etc. Recent efforts in NER try to extract entities and link them to linked data entities. Linked data is a term used for data resources that are created using semantic web standards such as DBpedia. There are a number of online tools that try to identify named entities in text and link them to linked data resources. Although one can use these tools via their APIs and web interfaces, they use different data resources and different techniques to identify named entities and not all of them reveal this information. One of the major tasks in NER is disambiguation that is identifying the right entity among a number of entities with the same names; for example "apple" standing for both "Apple, Inc." the company and the fruit. We developed a similar tool called NERSO, short for Named Entity Recognition Using Semantic Open Data, to automatically extract named entities, disambiguating and linking them to DBpedia entities. Our disambiguation method is based on constructing a graph of linked data entities and scoring them using a graph-based centrality algorithm. We evaluate our system by comparing its performance with two publicly available NER tools. The results show that NERSO performs better.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Retrieval models, Selection process

I.2 [**Artificial Intelligence**]: Miscellaneous

I.7.5 [**Document and Text Processing**]: Document Capture – Document analysis

General Terms

Algorithms, Experimentation, Performance

Keywords

Named entity, named entity disambiguation, linked data, DBpedia, semantic databases, centrality algorithms, text annotation, closeness centrality

SWIM 2012, May 20, 2012, Scottsdale, Arizona, USA.

1. INTRODUCTION

Web of Data and Linked Data

Linked data refers to the web of data in contrast to the web of documents. Linked data extends the current web that consists of documents and the links between documents. In the case of linked data or the web of data, not just documents but also data elements (or things) and the links between these data elements exist. Not only that, but the links are meaningful unlike the links in the web of documents; links between data elements have types. Linked data is therefore more structured and machine processable; applications can traverse this data web, easily find useful information and pinpoint the right information [15]. In the web of documents, or the current web, searching and finding information is by way of parsing documents and looking for useful information by matching keywords and terms; therefore it is a dummy search.

One of the central projects towards linked data vision is Linked Open Data (LOD)[1] project. It collects links and pointers to all linked data datasets on the web. There are over 300 datasets listed in LOD currently, consisting billions of information assertions (or triples). Linked data datasets are created using 4 simple rules as outlined by Tim Berners-Lee[2] : (1) use URIs as names for data elements or things, (2) use HTTP URIs so people can look up those names, (3) when a URI is looked up, provide useful information in RDF[3] and SPARQL, (4) include links to other URIs, so that more things can be discovered.

DBpedia

DBpedia[4] is one of the central linked data datasets in LOD. It is created from Wikipedia by converting structured information (such as infobox information) to RDF data model. It currently contains more than 3.5 million things, including 416,000 persons, 526,000 places, 106,000 music albums, 60,000 films, etc. in 15 different languages. All this information is captured in nearly a billion RDF triples.

Named entity recognition and disambiguation

Since the web of documents is not structured, it is difficult to locate the actual data in the documents. Locating the data within documents, which are basically free text documents, depends on lexical analysis and natural language processing techniques. One recent approach to finding relevant information within free text and web documents is associating and annotating the "named entities" within documents to the linked data entities that explain

[1] http://linkeddata.org/
[2] http://www.w3.org/DesignIssues/LinkedData.html
[3] http://www.w3.org/RDF/
[4] http://dbpedia.org

those named things. Once the annotation and the linking are done, then the navigation software can reach other related data and information by following the links from the linked data entities. This process is also called "named entity recognition" and it follows a similar approach as in information extraction techniques. Named entities are identified by matching entity names with the names of the entities in linked data datasets. One major problem in named entity recognition process is that names could be linked to multiple different entities in linked data. This is because some entities are named the same but they refer to different things. For example, "Washington" is a short name for "Washington, DC" as well as the name of "Washington State". Finding the right entity requires finding the context of the text surrounding the named entity; meaning other named entities should be identified and their relationships should be considered to find the right entity. This process is called "named entity disambiguation".

We developed a system called NERSO, short for Named Entity Recognition Using Semantic Open Data. It is in the category of other named entity recognition and disambiguation systems that are using linked data, such as Spotlight, etc. We use a centrality scoring mechanism on the entity graph to disambiguate the similarly named entities.

In section 2 related work is presented. Linked data and DBpedia, the data source we use, and its features are explained in section 3. We describe our named entity recognition and disambiguation method in section 4. Tests we conducted and the evaluation results are presented in section 5 and we conclude in section 6.

2. RELATED WORK

Word sense disambiguation (WSD) is a closely related area to named entity recognition (NER). It is about finding the right sense for words used in sentences and documents. WSD requires linguistic approaches and possibly large linguistic databases such as WordNET[5].

Wikipedia as a source for WSD has been used in a number of works. Fogarolli proposed to use link structure of Wikipedia articles [7]. In their work, words are disambiguated using semantic relatedness by crawling Wikipedia articles using the links between them. Links between Wikipedia articles can provide a way for identifying relationships and understanding how some topics are connected. They found that strongly connected topics belong to the same context in general [7].

Another WSD system using an external source like WordNET to disambiguate was also proposed in [1]. Connection between concepts is calculated by similarity measuring algorithms using WordNET's taxonomy. A graph of relations between words is scored using well known centrality scoring methods. It is shown that centrality scoring methods can be used to disambiguate words with multiple meanings [1].

Regarding NED (Named Entity Disambiguation), systems using linked open data (LOD) for annotating entities have gained popularity recently due to the growth in the area of semantic web and its manifestation on the web, namely linked data. In LOD cloud, DBpedia, DBLP, YAGO, FreeBase are the most known data sources for this purpose.

Hassel et al. create an ontology from DBLP dataset and populate its data by parsing messages from DBWorld mailing list [4]. Using this "populated ontology" they disambiguate the entities such as names of authors written in their papers or domain of interests. They consider text similarity of entities in order to disambiguate. They also use text co-occurrence relationships, such as text proximity.

Most of the recent work on NED is focused using Wikipedia or DBpedia due to their wide coverage of entities. Some of them use BOW (Bag of Words) approach while others use semantic relatedness between concepts.

Bunesco and Pasca [10], Cucerzan [5], Michalcea and Csomai [16] used texts from Wikipedia to annotate. Bunesco and Pasca proposed using encyclopedic data to annotate named entities and disambiguate using ranking algorithms [10]. They used cosine similarity to distinguish between meanings of words as a ranking solution for disambiguation of named entities. Cucerzan proposes a very similar approach [5]. First, he pre-processes the Wikipedia collection and extracted more than 1.4 million entities with an average of 2.4 terms for each entity and 540 thousand (entity, category) pairs. The knowledge extracted from Wikipedia is then used for the disambiguation process; the vectorial representation of the processed document is compared with the vectorial representation of the Wikipedia entities to decide.

DBpedia Spotlight project is a well-known NER tool, it classifies and disambiguates named entities based on DBpedia ontology [9]. Spotlight automatically annotates free text documents with links to DBpedia entities. Spotting function first finds named entities in the text with all corresponding resources. Then the disambiguation step matches named entities to the right DBpedia resources based on the context similarity measures. Their context similarity measure is based on term frequency (TF) and an inverse candidate frequency (ICF) which measure the power of words based on their co-occurrence in articles. The intuition behind ICF is that the discriminate power of a word is inversely proportional to the number of entities it is associated with. Spotlight is configurable with a 'confidence' and a 'support' parameter. Setting the confidence parameter high lets Spotlight to avoid incorrect annotations at the risk of losing correct ones. With support parameter, users can set the minimum number of inlinks a DBpedia resource has to have in order to be annotated so that high precision or high recall can be obtained. Results show that DBpedia as a service for annotating named entities has promising results.

Hoffart et al. present another online tool called AIDA for the disambiguation of named entities leveraging knowledge bases like DBpedia or YAGO [12]. Their method builds the subgraph of entities mentioned in the text with a greedy algorithm that approximates the best joint mention-entity mapping.

Gentiles et al, like 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 on a combination of features extracted from Wikipedia [2]. Our method on the other hand does not extract any features of entities, only considers link relations between them. Additionally, we propose a novel scoring method for disambiguation process instead of a random-walk model.

Han et al proposes similarity measure for annotating entities using Wikipedia as a semantic network [8]. Initially, they construct a large-scale semantic network from Wikipedia so that semantic knowledge can be used efficiently. And then, by leveraging this

[5] http://wordnet.princeton.edu/

semantic knowledge like social relatedness between named entities, they measure the similarity between occurrences of names.

Ni et al propose a method that extracts information from LOD about entities and builds a type-oriented knowledge base [6]. Named entities are scored based on their types and then this score is used in an existing classifier. It is shown that the classifier's performance is increased with the use of the scoring method.

Kulkarni et al. [11] propose the collective disambiguation of entities defined in a context by optimization methods. Authors proposed to use Linear Programming and a Hill-climbing approach for optimization.

Ferragina at al. [13] propose a system called TAGME which uses Wikipedia as a source. The method disambiguates entities based on "collective agreement" which is the sum of votes from other entities detected in the text. These votes are computed using semantic relatedness of entities mentioned in the text [13].

A survey of named entity recognition systems is collected in [3]. They specifically survey many of the existing semantic tagging technologies and services and compare them.

We follow the named entity recognition and disambiguation approach in our work and provide a hybrid solution combining a number of known approaches. Our method is based on graph-based centrality methods and relatedness of entities in the given document.

3. LINKED DATA AND DBPEDIA

Linked data has been growing rapidly, now consisting more than 300 datasets, and DBpedia has a very central and prominent place in this network [14], many datasets have links to DBpedia resources. DBpedia is getting more centralized and developing as a linking hub between other data sources in the linked open data cloud. For each entity, DBpedia defines a globally unique identifier that can be referenced over the Web in the RDF description of an entity.

Surface Forms

Terms that are used to name the entities in the text are called "surface forms". In DBpedia Spotlight project surface forms are identified with a preprocessing step and stored in the DBpedia Lexicalizations Dataset[6] (DLD). Surface form entries in this dataset are identified by entity labels, redirects and disambiguations in DBpedia.

Utilization of DBpedia as a Source for NER

We do not use DLD directly in our work. We however utilize a similar approach. We use three kinds of sources in DBpedia to extract surface forms. These are (1) label and other similar data properties we have chosen, (2) disambiguation and (3) redirect pages (DBpedia ontology properties dbont: wikiPageDisambiguates[7] and dbont:wikiPageRedirects[8]). All entities spotted using these sources are essential in the disambiguation step of our method. Each surface form has a candidate list of entities and if any surface form contains multiple entities then that surface form should be disambiguated.

(1) Data properties

We consider rdfs:label and other similar data properties that might contain surface form data. We chose a short list of data properties that express the title, name, or similar data. In the current setting the list contains these data properties: rdfs:label, foaf:name, dbpprop:officialName, dbpprop:name, foaf:givenName, dbpprop:birthName, dbpprop:alias.

(2) Disambiguation pages

dbont:wikiPageDisambiguates property is a predicate used to group entities that have various meanings for the same title. For example, "Washington.D.C" and "George_Washington" are grouped under "Washington(disambiguation)" entity because they can both be referenced with a common title "Washington". Any object with the type dbont:wikiPageDisambiguates and a label that contains the searched surface form will be selected, meaning all entities grouped under these selected objects will be added to the candidate list of a surface form.

(3) Redirect pages

dbont:wikiPageRedirects property is a predicate that is used to show alternative titles of a given entity. These type of pages 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. Reference pages of objects with a type dbont:wikiPageRedirects and a label that contains the searched surface form will be added to the candidate list. In this example if the 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 surface form.

4. METHOD

We designed a named-entity disambiguation algorithm that consists of 3 main steps. These are:

(1) Spotting algorithm

The text is parsed from beginning to end using a sliding window of a certain maximum number of words to find the surface forms and the matching linked data (DBpedia) entities.

(2) Constructing the graph of entities and relationships

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

(3) Disambiguation

Several nodes (entities) in the constructed graph might match to the same surface form as explained above. Only a single node (entity) needs to be selected to represent each surface form when multiple nodes exist in the graph. This is the disambiguation step.

Below we explain these 3 steps in detail.

4.1 Spotting

In this subsection we present how the text parsing and surface form selection process is carried out. To spot the surface forms in a text, a "sliding window" approach is used. The text is parsed from beginning to end, and in each step the sliding window selects a small number of consecutive words. The sliding window size is set to a maximum size (the maximum number of words to be selected) at the beginning. In each step the selected word set from the window is searched in the database. If a surface form is found then it is added to the list of spotted surface forms, otherwise the window size is decreased by one from the right end of the window

[6] http://dbpedia.org/Lexicalizations
[7] http://dbpedia.org/ontology/wikiPageDisambiguates
[8] http://dbpedia.org/ontology/wikiPageRedirects

and the new surface form candidate is searched again in the database until a match is found. In the case a match is found, the sliding window size set to the maximum size again and slide over the matching words to the next word in line. If no match is found when the window size is shrunk to one word, then the window is again set to the maximum size and slide over to the next word in line.

In this approach, the maximum size of the sliding window must be set to a realistic one. The size should neither be too small so that the entities or surface forms that have long labels are not missed, nor too large so that the processing time is not too high. After some experiments we decided to use a sliding window with the maximum size of 4 words. Consider the following text that we used in our experiments.

Jailbreaking also allows an owner to unlock their phone and switch mobile carriers. Apple's phones, and its iPads, typically come with an exclusive contract with a mobile provider (originally only AT&T in the United States, although Verizon and Sprint versions have been added).

Figure 1 Sample Text from Dataset2 in EVALUATION

Spotting the surface forms in the text follows the steps listed in Table 1 assuming that the sliding window size is set to a maximum of 4 words. Surface form candidates, which are selected by the sliding window, are searched in the target dataset by looking up the 3 resources as explained above. Table 2 lists the queries we run to find the surface forms.

Table 1 Steps for spotting surface forms using sliding window on the sample text

Step	Sliding window	Result
1	Jailbreaking also allows an	not found, reduce window size
2	Jailbreaking also allows	not found, reduce window size
3	Jailbreaking also	not found, reduce window size
4	**Jailbreaking**	**found**, add to the list, move window
5	also allows an owner	not found, reduce window size
6	…	…

Table 2 Queries for spotting surface forms

Query	Source
`SELECT distinct ?s WHERE { ?s rdfs:label "+searchText+"@en." ?s foaf:name "+searchText+"@en." ?s foaf:givenName "+searchText+"@en." … }`	Data properties
`SELECT distinct ?s WHERE { ?redirect dbont:wikiPageRedirects ?s. ?redirect rdfs:label "+searchText+"@en. }`	Redirect pages
`SELECT distinct ?s WHERE { ?disamb dbont:wikiPageDisambiguates ?s. ?disamb rdfs:label "+searchText+". }`	Disambiguation pages

Let $D=\{w_1, \ldots, w_n\}$ be the set of words in a text. Our goal is to find the surface forms in the text. After the spotting step we find $S(D)=\{s_1.....s_m\}$ the spotted surface forms in the text. For the sample text in Figure 1, following is the list of spotted surface forms:

S(*D*)={*"Jailbreaking", "Apple's", "iPads", ...*}

After finding the surface forms, the next step is extracting all the entities that match the surface form.Formally, let $E(s)=\{e_1, \ldots, e_k\}$ be the set of entities for a surface form *s*. For each *s* in *S(D)*, matching entities (URI representations) are extracted from DBpedia triple store and added to *E(s)*. For example, for the surface form "Jailbreaking":

E("*Jailbreaking*")={*Jailbreaking, Jailbreak_(album), Jailbreak_(TV_series), ...*}

Entities spotted are listed with their resource types, which are data properties (such as labels), redirect entities, and disambiguation entities (Figure 2). This information is later used in the scoring; disambiguation entities have a lower constant factor.

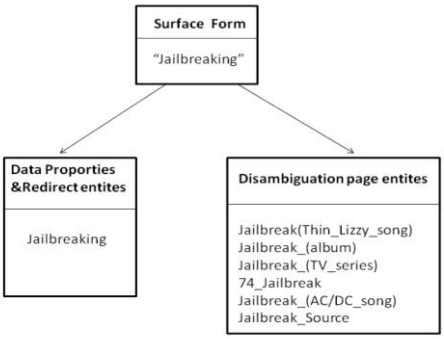

Figure 2 Entities spotted for a surface form "Jailbreaking"

Stopwords

Prepositions, adverbs, verbs, adjectives, pronouns are considered stopwords, not named entities. Words like "also" does not have any meaning on their own; they are meaningful along with a named entity. For example, "I_Am_Not_a_Human_Being" is the name of a play which is only meaningful with all 6 words. The words "am", "not", and "a" are stopwords. Our spotting algorithm skips stop words, meaning it does not search stopwords as surface forms when the sliding window has a single stopword in view. We developed a stopword list which is derived and extended from DBpedia Spotlight project's stopword list. Our stopword list is available for download in project's homepage[9].

4.2 Constructing the graph of entities and relationships

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 called "dbont:wikiPageWikiLink". Not only the regular page links but other tagged links (object and data properties) also have corresponding dbont:wikiPageWikiLinks added. For example, DBpedia has the triple "Washington,_D.C. country United_States" and for this it also has a second triple in the form of "Washington,_D.C. dbont:wikiPageWikiLink United_States" triple indicating that "Washington,_D.C." is related to "United_States".

In our method we search for "dbont:wikiPageWikiLink" links between spotted entities and using entities and the links found between the entities form a graph first. For example, if "Washington,_D.C." and "United_States" are spotted in the

[9] http://wis.etu.edu.tr/nerso/files/stopword.txt

process (4.1), we search for links between these two entities using the "dbont:wikiPageWikiLink" property. All links between spotted entity pairs are queried and a graph is constructed using the entities and the links between entities. Formally, *S(D)* is the set spotted surface forms for the text *D* that is to be annotated:

$$S(D)=\{s_1, s_2, \ldots, s_m\}$$

And, *E(s)* is the set of all entities in the linked dataset for each surface form *s* in S(D):

$$E(s)=\{e_1, e_2, \ldots, e_k\} \quad for\ s \in S(D)$$

The set of all entities for the document *D* is then *E(D)*:

$$E(D) = \bigcup_{s \in S(D)} E(s)$$

If there is a dbont:wikiPageWikiLink link between any two entities in *E(D)*, then the link is added to the relationships set or the link set *R(D)*. We then build a graph of spotted entities and the directed edges between these entities.

The graph is a 3-tuple construct *G*=(*E(D), R(D), sf*), where *E(D)* is the set of nodes representing the entities corresponding to surface forms, *R(D)* is the set of directed edges between any two entities in *E(D)* representing a dbont:wikiPageWikiLink link in the linked data from entity a to entity b, and *sf* is a scoring function that we use to nodes that will be used in the disambiguation step (next).

We used Java Universal Network Graph (JUNG)[10] library to construct and visualize the graph using entities and relationships list. Figure 3 is a partial representation of our sample text using JUNG libraries.

4.3 Disambiguation

In the graph we constructed some nodes are more central than others, they have more incoming and/or outgoing edges than the others. For example in Figure 3, "Apple_Inc.", "Apple" and "Apple_Switch_ad_campaign" are three of the entities spotted for the surface form "Apple's" in the sample text (Figure 1). "Apple Inc." node in the graph has more links to the other entities. Incoming and outgoing links play an important role in the disambiguation process. In this graph "Apple_Inc." has 2 outgoing and 5 incoming links in contrast to "Apple" and "Apple_Switch_ad_campaign" with only 1 outgoing link.

Graph based Centrality Scoring methods have been proposed before [1]. Centrality scoring methods have been successful because they take into account the relationships between nodes. Also in our method, it is crucial to calculate how central a node is in the graph. As our graph is directed, not all nodes are reachable from other nodes. For example, "AT&T" is not reachable from "Apple_Inc.", and "AT&T_Mobility" is reachable through "IPAD", that is a path with length 2 (Figure 3).

Closeness centrality concept in graphs[11] is similar to our customized centrality factor given below. We modified traditional closeness centrality scoring method to take into account the number of all related nodes. For every node the shortest path from that node to all reachable nodes in the graph is calculated using Djikstra's shortest path algorithm.

[10] http://jung.sourceforge.net/
[11] http://en.wikipedia.org/wiki/Centrality#Closeness_centrality

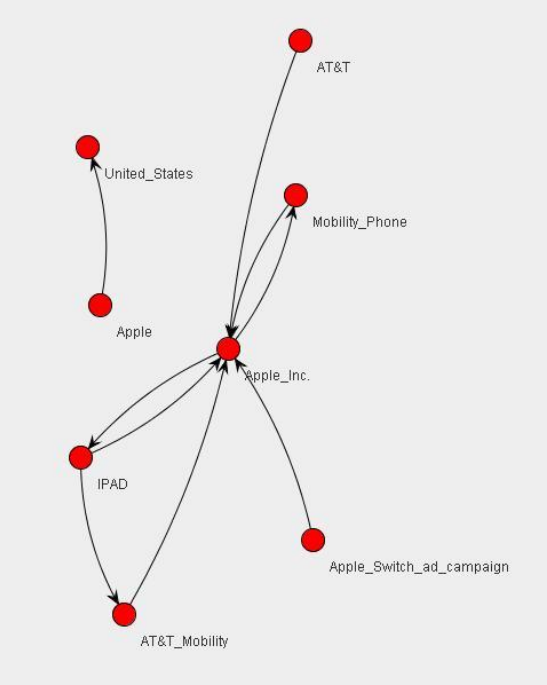

Figure 3 Graph for text above (partially) full image can be obtained from project homepage

Centrality factor for a given node is then calculated by dividing the total number of reachable nodes from the node to the sum of the lengths of the shortest paths to those nodes. Or formally, for node V_a centrality factor can be calculated using the following formula.

$$CF(V_a) = \frac{\partial_{V_a}}{\sum_{V_b \in V} s(V_a, V_b)}$$

$\sum_{V_b \in V} s(V_a, V_b)$ is the sum of the shortest distances between V_a and the reachable nodes from V_a. ∂_{V_a} is the number of all reachable nodes from V_a.

For the partial graph represented in Figure 3, the node *"Apple_Inc."* has three reachable nodes {*AT&T_Mobility, IPAD, Mobility Phone*} and the sum of the shortest distances to these nodes is 4. Centrality factor of the node *"Apple_Inc."* will be 3 divided by 4 or 0.75.

Centrality factor of a node V_a is used to score the node as below:

$$Score(V_a) = CF(V_a) * in\_links(V_a) * out\_links(V_a) * k$$

$CF(V_a)$ is the centrality factor for node V_a, $in\_links(V_a)$ is the number of incoming links for node V_a, $out\_links(V_a)$ is the number of outgoing links for node V_a. And *k* is a constant number depending on the type of node V_a. *k* is set to 6 if the entity is from data properties or redirect pages, and *k* is set to 1 if the entity is retrieved from *disambiguation pages*. *k* is determined by the performance of the disambiguation process in our tests.

As explained in section 4.1 entities are spotted depending on data property attributes, redirect labels or from disambiguation pages. Disambiguation pages collect a high number of unrelated entities

of many different types, which might affect the disambiguation adversely. Therefore, entities from disambiguation pages are given less weight in the scoring function.

Finally, depending on the score of nodes that need to be disambiguated, for each surface form the disambiguation algorithm finds the highest scoring candidate, and selects it to represent and annotate the surface form in the text, and eliminates the other candidate entities. For example, for surface form "Apple's", entity "Apple_Inc." has the highest score in Figure 3, is therefore selected to annotate "Apple's".

5. EVALUATION

We evaluated our approach using two different datasets which are publicly available. Each dataset contains links to DBpedia resources. We named the test datasets as Dataset1 and Dataset2:

- **Dataset1**: This dataset is from DBpedia Spotlight Project [9]. It contains a goldset from 10 different news articles with a total of 251 entities. 217 out of these entities are ambiguous, meaning they match to multiple DBpedia entities.
- **Dataset2**: We also created our own goldset for testing. We selected 10 different news articles from NY Times, Washington Post, and CNN. We included 1 to 4 paragraphs from each article in the dataset. We asked two people to annotate these articles on their own separately. They decided on the most important entities in each article and a common set of entities are created as Dataset2[12]. This dataset contains 157 different entities. Out of these 157 entities, 128 of them are ambiguous.

Both datasets have a high number of ambiguous entities to test the success of disambiguation process meaningfully. Dataset1 has 86% of 251 entities ambiguous and Dataset2 has 81% of 157 entities ambiguous (Table 3). To give an example, "David Beckham" is not an ambiguous entity, it matches to a single entity in DBpedia; yet "Washington" is an ambiguous entity, it matches to "Washington,_D.C.", "Washington_(state)" and other entities in DBpedia.

Table 3 Dataset statistics

	Number of entities	Number of ambiguous entities	Ratio of ambiguous entities
Dataset1	251	217	86%
Dataset2	157	128	81%

We have used 3.6 version of DBpedia, which is open to everyone[13], and performed our tests on it. DBpedia 3.6 was loaded into Virtuoso Database Server[14] which runs on a cluster with four nodes each having 4 GB Memory and 1.3 GHz AMD Phenom(tm) 9950 Quad-Core Processor.

We compared our system NERSO with two other publicly available NER and annotation projects: Zemanta[15] and DBpedia Spotlight[16]. Spotlight was tested with default configurations on its online demo site. Zemanta annotates entities in the following categories only: persons, books, music, movies, locations, stocks, and companies. DBpedia Spotlight annotates all types of entities that are defined in DBpedia dataset. DBpedia Spotlight annotates given text documents with links to DBpedia entities. Zemanta on the other hand annotates given text documents with links to Wikipedia, Amazon, IMDB and others like the homepage of the annotated entity. We took into account only Wikipedia links as all annotated entities contained Wikipedia links. At the time of writing DBpedia Spotlight's v0.5 was released and both datasets were tested on the online version of Spotlight v0.5.

Test results are presented in Table 4.

Table 4 Disambiguation results for Spotlight, Zemanta and NERSO

	Dataset1			*Dataset2*		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
NERSO	42%	**60%**	**50%**	29%	**70%**	**41%**
Spotlight	39%	45%	42%	30%	51%	38%
Zemanta	73%	21%	33%	62%	29%	39%

For Dataset1, DBpedia Spotlight (no configuration) shows a 45% recall rate. Zemanta on the other hand recognizes and disambiguates 21% of entities correctly. Our system NERSO performs higher than Zemanta and Spotlight with a recall rate of 60% which is a promising performance. Also F1 score[17] of NERSO is in range of competition. Evaluation results in [9] does not contain disambiguated entity list for each annotation system. For a better comparison we tested both Spotlight and Zemanta and reported all annotation results with the disambiguated entities and regarding surface forms. In the Spotlight paper F1 score of the Spotlight project with no configuration is reported as 45% [9] whereas we calculated 42%. In the same paper, Zemanta project result is reported with 39% F1 score, whereas we calculated F1 as 33%. We cannot explain these differences since we do not have the annotation results as performed in the paper [9]. But we are guessing the difference could be due to using the different versions of datasets or software. For example, we used the online version of Spotlight 0.5, which is probably newer than the version used in the paper.

For Dataset2, DBpedia Spotlight (no configuration) disambiguates with a recall rate of 51% and Zemanta has a 29% recall rate. Our system NERSO on the other hand outperforms both Spotlight and Zemanta with 70% recall rate. F1 score of NERSO (41%) is also slightly higher than both systems (38% and 39% respectively). These results show that NERSO performs well for disambiguating entities correctly with a success of 70% recall rate.

Precision rates are high for Zemanta while it is lower for Spotlight and NERSO. This is because Zemanta annotates fewer entities (selected categories only) and does it mostly correctly (73% and 62% for Dataset1 and Dataset2 respectively).

NERSO performs better than Zemanta and Spotlight projects in terms of catching the most number of entities from both datasets. Zemanta performed the worst among the three (lowest recall rates).

We also evaluated the success of the disambiguation process by counting the number of successful disambiguations for both datasets. That is for surface forms with multiple entity matches (86% for Dataset1 and 81% for Dataset2 as shown in Table 3) we measure the ratio of matching to the right entities in DBpedia. Results are listed in Table 5. This data is not available for

[12] http://wis.etu.edu.tr/nerso/evaluation.html
[13] http://wiki.dbpedia.org/Downloads36?v=ebv
[14] http://virtuoso.openlinksw.com
[15] http://www.zemanta.com/demo/
[16] http://spotlight.dbpedia.org/demo
[17] http://en.wikipedia.org/wiki/F1_score

Spotlight and Zemanta since we do not have access to the datasets of those systems.

Table 5. Disambiguation success of NERSO

	Dataset1	Dataset2
Ratio of ambiguous surface forms	86%	81%
Recall rate for ambiguous forms	69%	84%

According to the results presented in Table 5, NERSO shows a 69% successful disambiguation rate for the multiple-entity surface forms in Dataset1 (86% of 251 entities), and 84% successful disambiguation for multiple-entity surface forms in Dataset2 (81% of 157 entities). This is consistent with the results in Table 3 for Dataset1 and much better for Dataset2.

6. CONCLUSION

We presented NERSO, a named entity recognition and disambiguation system using graph-based scoring method for annotating named entities in a given text using linked data. Our approach is based on spotting surface forms in the text by mapping them to linked data entities, and then constructing a directed graph of spotted entities using their relationships in the linked data, and then finally disambiguating the multiple entities that match to the same surface forms. We have shown experimentally that our graph-based approach performs better than a bag of words approach such as Spotlight's. Graph-based approaches perform better since they take into account information drawn from the entire graph of semantically related entities. In the disambiguation process, in order to score entities, we used a centrality scoring method (closeness centrality). An entity is selected if it is more central among the other candidates of a surface form since it has more semantic relations with the other entities in the graph.

We compared our system with two well-known and publicly available named entity recognition and annotation services. As shown in the evaluation section our system performs better than the two other systems.

For future work we plan to improve the system in terms of query execution time by using parallel computers, developing new scoring methods for disambiguation, and using multiple linked data resources. Lower precision rates in the results indicate that our system over annotates somehow, and this should also be worked on to increase the precision rates.

All of our test data is available online[18] so that it can be checked and compared against for future works.

7. REFERENCES

[1] Sinha, R., Mihalcea,R. 2007. Unsupervised graph-based word sense disambiguation using measures of word semantic similarity. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007)*.

[2] Gentile, A., Zhang, Z., Xia, L. 2009. Graph-based semantic relatedness for named entity disambiguation. In *Proceedings of International Conference on Software, Services & Semantic Technologies*, 2009.

[3] Gerber, A., Gao, L. 2011. A Scoping Study of (Who, What, When, Where) Semantic Tagging Services. *Research report Public Release February 2011, eResearch Lab, The University of Queensland*

[4] Hassell, J., Aleman-Meza, B. 2006. Ontology-driven automatic entity disambiguation in unstructured text. In *Proc. 5th International Semantic Web Conference (ISWC), volume 4273 of LNCS*, pp. 44–57, Athens, GA, 2006

[5] Cucerzan, S. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of Empirical Methods in Natural Language Processing Conference on Computational Natural Language Learning 2007*, pp. 708–716, 2007.

[6] Ni, Y., Zhang, L., Qiu, Z., Wang, C. 2010. Enhancing the open-domain classification of named entity using linked open data. In *Proc. 9th International Semantic Web Conference (ISWC 2010)*, pp. 566–581, Shanghai, China, 2010.

[7] Fogarolli, A. 2009. Word Sense Disambiguation Based on Wikipedia Link Structure. *IEEE International Conference on Semantic Computing*, pp. 77-82, 2009.

[8] Han, X., Zhao, J. 2009. Named entity disambiguation by leveraging Wikipedia semantic knowledge. In *Proc. of the 18th ACM Conference on Information and Knowledge Management, (CIKM 2009)*, pp. 215–224, 2009.

[9] Mendes, P. N., Jakob, M., García-Silva, A., Bizer, C. 2011. DBpedia Spotlight: Shedding Light on the Web of Documents. *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics).* Graz, Austria, 7–9 September 2011.

[10] Bunescu, R., Pasca, M. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL* , pp. 9–16.

[11] Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S. 2009. Collective annotation of Wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2009)*, pp. 457–466, New York, NY, USA, 2009.

[12] Hoffart, J., Yosef, M., A., Bordino, I., Furstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum. G. 2011. Robust Disambiguation of Named Entities in Text. In *Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 782–792, July 2011.

[13] Ferragina, P., Scaiella, U. 2010. Tagme: on-the-fly annotation of short text fragments (by Wikipedia entities). In *Proc. of the 19th ACM Conference on Information and Knowledge Management, (CIKM 2010)*, 1625–1628.

[14] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S. 2009. 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.

[15] Bizer, C., Heath, T., Berners-Lee, T. 2009. Linked data-the story so far. *Int. Journal on Semantic Web and Information Systems, Special Issue on Linked Data*, 4(2), 1–22, 2009.

[16] Mihalcea, R., Csomai, A. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proc. of the 16th ACM Conference on Information and Knowledge management (CIKM 2007)*, Lisbon, Portugal, pp. 233-242, 2007

[18] http://wis.etu.edu.tr/nerso