Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Computer Science	Spring semester, 2015 Open
Writer: Brage Michael E. Roalkvam	 (Writer's signature)
Faculty supervisor: Krisztian Balog External supervisor(s):	
Thesis title: Effective Entity Linking for News Articles	
Credits (ECTS): 30	
Key words: Entity linking, Freebase, ERD Challenge '14, Named Entity Linking, Java.	Pages: 35 + enclosure: 4 Stavanger, 13/06/2015 Date/year

Front page for master thesis
Faculty of Science and Technology
Decision made by the Dean October 30^{th} 2009

Effective Entity Linking for News Articles

by

Brage Michael E. Roalkvam

A thesis submitted in partial fulfillment for the
degree of Computer Science

in the
Faculty of Science and Technology
Department of Electrical and Computer Engineering

Saturday 13th June, 2015

Declaration of Authorship

I, Brage Michael E. Roalkvam, hereby declare that this thesis titled, 'Effective Entity Linking for News Articles' and all my written work submitted to UiS:

- has not been used in other written submissions to UiS or other institutions in Norway or abroad.
- does not refer to the work of others without citation in the text.
- does not refer to earlier work without citation in the text.
- cites all sources (including web sites) in the literature lists.

I realize that infringement on these rules is to be regarded as cheating while sitting for an exam or test.

Signed: 

Date: 13/06/2015

'I assume that by knowing the truth
you mean knowing things as they really are.'

- Plato

Abstract

This master's thesis presents an entity-linking tool for detecting entity-bearing words in news articles and linking them to the corresponding entries in a knowledge base. Given the vast volume of news articles produced every day, such a tool needs to be not only effective but also efficient. The approach consists of three steps. First, entities are spotted on the basis of capitalized (uppercase) letters in words. Next, entities with surface forms matching the mention are considered. Finally, in the disambiguation phase, a new method based on local relatedness is employed. Using the Entity Recognition and Disambiguation 2014 Challenge platform, we demonstrate that the efficiency of this solution is competitive with other available methods.

Acknowledgements

Foremost, I would like to express my gratitude to my supervisor, Krisztian Balogfor the valuable comments and guidance. He was always ready to help me and provided me with a lot of great information.

I would like to thank Pål Thingbø and Eirik W. Henningsen from Thingpal AS for inspiring meetings and great information.

Last but not least, I would like to give my sincerely thanks to my family and all my friends. Thanks for giving support to me to fulfill my master thesis.

Contents

1

Introduction

When one reads a text – whether a news article, book or a blog post – the brain makes associations between the words and their meaning. For instance, when 'Obama' occurs in the text, an association is immediately made, most likely to the current US President. These associations can be seen as 'links'. Our aim with this work is to create a computer program that can identify such links automatically. For humans, links are established on the basis on the knowledge that the reader in question has. The more knowledge the person has, the more linkages can be made. Initially, these links are based on individual words; then, as one reads further, the context surrounding the word will have an impact on the linkages made. For instance, 'Washington' in isolation may have various meanings, like George Washington, Washington DC, or Washington State. To decide which of these candidates is appropriate, the brain analyses the context surrounding the word. With this information, the brain can then disambiguate between candidates and select the correct one. Figure 1.1 shows a snippet of Wikipedia page about Barack Obama, where links to various entities, such as the United States and Harvard Law School, have been identified by human editors. Obviously, annotating articles manually is a huge effort; we wish to automate this process. For a computer, this is a complicated task; we will refer to it as the problem of *entity linking*.

FIGURE 1.1: Illustrating entity links

Entity-linking approaches typically evaluate the text in three phases: spotting, link detection, and disambiguation. The spotting phase scans the text for words ('mentions') that can potentially refer to entities, i.e. can be linked. The link-detection phase identifies a set of potential entity links that can be associated with the mention. The disambiguation phase considers the collection of potential entity links and selects a single one to map the spotted mentions to a given reference knowledge base. The reference knowledge base used here is Freebase. Freebase covers millions of people, places, and things, grouped in terms of topic, types and properties. Each topic is linked to other related topics, creating a relation graph. Topics are annotated with types and properties that describe or are related to the topic.

This master's thesis presents an entity-linking tool that operates as follows. It uses a technique for spotting words that contain uppercase letters. This method is shown to be extremely efficient and is able to detect most entity-bearing words. The problem is when entity words consist of lowercase letters only: these will not be detected by the spotter. The spotted words or phrases are further sent to the link-detection step, which employs a dictionary of entity surface forms and corresponding links along with statistical information, extracted from a large web corpus. One disadvantage of the dictionary-based approach is that it fails to identify entities if the mentions do not exactly

match one of the known surface forms. To overcome this, we employ a 'window sliding' technique for detecting entities, motivated by Hakimov et al. [5]. In the disambiguation step, the baseline approach involves selecting the entity with the highest 'commonness', meaning the frequency that this entity is linked to the word or phrase. Additionally, we introduce a novel local relatedness score, that measures the relation between entity links for the given mention and links for other mentions in the document.

The development of our tool was motivated by a real-world use-case. The company Thingpal AS has approached us with the need for a high-throughput entity linking system that can be used to annotate a large streaming news collection (with tens of thousands of articles per day). As they could not provide a purpose-built test set, we base our evaluation on a publicly available benchmarking resource.

The evaluation of entity-linking systems is performed by comparing the machine-generated annotations against a 'gold standard' created by humans. F1-measure is used to evaluate effectiveness, which is the micro-average of precision and recall. Precision and recall are calculations of matching annotations. The development of our tool was motivated by a real-world use-case. The company Thingpal AS has approached us with the need for a high-throughput entity linking system that can be used to annotate a large streaming news collection (with tens of thousands of articles per day). As they could not provide a purpose-built test set, we base our evaluation on a publicly available benchmarking resource.

Experimental comparison of our tool against ERD'14 Challenge submissions has shown it to be competitive with the state of the art in terms of efficiency (latency). The effectiveness of our tool is acceptable, but there is still room for improvement.

The remainder of the thesis is organised as follows. Chapter 2 presents the problems and challenges involved in making an entity system. Next, in Chapter 3, we discuss entity linking systems from the literature, their methods and performance. Chapter 4 describes the dataset used in the system, and Chapter 5 presents our entity system. Chapter 6 examines with experimental setup against ERD'14 Challenge platform. Chapter 7 presents the results, with a discussion, and Chapter 8 looks into future development for the system. Finally, we summarize our approach and findings and conclude in Chapter 9.

2

Problem statement

Vast amounts of unstructured data are produced in news articles every day. Entity linking can help to structure and connect such data. This requires a tool that is both efficient and effective. The goal for Entity Recognition and Disambiguation(ERD) is to identify mentions and link them to a knowledge base. *Entity mention* in a document X can be defined as:

$$X \supseteq annotations\{A_1, A_2, \ldots, A_n\}, where A_n = \{B, E, id\}, n \geq 0 \tag{2.1}$$

where annotations is a set of entity annotations contained in document X. Each annotation has set consisting of $beginoffsetB$, $endoffsetE$, and a $knowledge{-}baseidentifierid$. The entity-linking process starts with the document as input to the application; the output is a set of annotations. For instance:

> Hollywood superstar Julia Roberts lit up London with her beaming smile at the gala premiere of her new film, Eat Pray Love.

TABLE 2.1: Set of annotations form above.

Entity word	**Begin offset**	**End offset**	**Knowledge base identifikator**
Hollywood	36	45	/m/0f2wj
Julia Roberts	56	69	/m/046zh
London	77	83	/m/04jpl
Eat Pray Love	145	158	/m/07kdkfj

As noted, the steps in the entity-linking process steps are as follows: spotting, link detection and disambiguation. For the spotting phase there will be some rules for identifying potential entity-links in the text. Can some words be ignored, or it is necessary to search through all words and phrases? Are there some words that stand out as possible entity? The spotter should be able to detect proper nouns and ignore common nouns. Link detection is the collection of candidate links to a spotted word in the text. As a spotted word may have several different meanings, an assembly of candidate links should contain the correct entity link. The number of candidate links will vary, depending on the word itself and the knowledge base. Should all candidate links to a word be collected, or only those above a given threshold? What if the spotted word is not found in the knowledge base – should it be discarded, or should another type of analysis be conducted?

Disambiguation is the evaluation of which candidate link is the correct one, and which words are valid entity links. Words are evaluated so that the correct candidate link can be selected and the correct words chosen and linked up to a database.

Entity linking is not a new discovery. There will always be developments in improving efficiency and effectiveness. Today's tools got a good effectiveness, but suffer in efficientness. For many of the tools it should be possible to save time in the spotting phase through simpler methods of detecting the right word or phrase. The aim is to develop an entity-linking tool that is efficient and effective when applied to news articles. The tool needs to handle huge amounts of new articles to be annotated rapidly. In evaluating the proposed application, it is tested against pre-annotated documents, so-called 'golden standard'. This golden standard was made for the ERD'14 Challenge, and is available for developers to use. It is tested on latency and F1-measure: i.e. the micro-average of precision and recall, as described in detail in Section 6.1.

The aim of this thesis, besides making a fast and effective entity tool that can compete with the state-of-the-art system, is to prepare an entity tool that the company Thingpal AS can use in their service. Thingpal AS host a website called Gumpaper[1] which is a collection of news articles, where user can customized their news feeds based on their

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

specific interests. This is done by collecting RSS feed from the news publisher, and presented as readable text. To give the users better service Thingpal want to provide a pre-filled article with entities. That will enable users to discover entities of interest to them. This report describes an entity service developed with a focus on latency (efficiency) and annotating news articles.

3

Related Work

This chapter offers a brief review of some entity applications on the market today. Techniques that are used in entity recognition and disambiguation are described, and some of the methods are implemented in the proposed method. Applications examined here are Wikipedia Miner, DBpedia Spotlight, NERSO, Tulip, Seznam, as well as ERD'14 Challenge platform.

3.1 Wikipedia Miner

Milne and Witten [12] present Wikipedia Miner, a tool for mining the unstructured data in the rich encyclopedia Wikipedia. The tool is module based program and serves for comparison of words, search, and annotation. The annotation service is an entity-linking tool, one of the first on the market. In the present thesis the annotation process is the focus. Wikipedia Miner uses summary files and the XML dump from Wikipedia, which is stored in the Berkeley Database (BDB). Wikipedia Miner is built up with labels that can contain multiple articles. 'Labels' are words or phrase that may have several meaning and one article is representative of one of them; conversely, one article may refer to many labels. The toolkit tracks how often the labels are used within linking and found in plaintext. Out of this it generates a ratio called 'prior link probability'. The

annotation process starts by gathering all possible labels within a document that have a prior link probability above a given threshold. The disambiguator uses machine learning to match the right article to the right label. The disambiguator is trained on article-to-article links in Wikipedia. When the label has the correct article, all the various labels are weighed up against the context to decide which label is most relevant to the document. Wikipedia involves unstructured text that will entail considerable processing time for analysing. With Wikipedia Miner, this is done by sending phrases as queries to the BDB to find matching Wikipedia labels. However, the BDB is a bottleneck here, since it is a file-based database. With long document with many words or phrases to look up, Wikipedia Miner will be very slow.

3.2 DBpedia Spotlight

Mendes et al. [9] present the DBpedia Spotlight entity tool, which uses the DBpedia ontology. The DBpedia contains Wikipedia Encyclopedia, which is unstructured and converted to a RDF model in a structured form. Wikipedia page titles are stored as labels for the DBpedia resources, which also contains Wikilinks. Wikilinks consists of Wikipedia links to other Wikipedia pages that represent occurrences of the resources. DBpedia Spotlight can be customized for customer use, and has several different spotting methods, as also disambiguation methods. If, say, the user wants to annotate only football players in given documents from the 19th to 20th century, the DBpedia Spotlight can be set up with a special spotter and disambiguator. The link-detection phase collects candidates for the spotted phrase, also working as a rough disambiguator by selecting fewer candidates. The disambiguation phase uses term frequency and inverse candidate frequency. Term frequency measures the relevance of a word in the resource, whereas inverse candidate frequency measures the importance of the word. Mendes et al. [8] propose several different approaches for annotation; were Named Entity Recognition (NER) extended by noun phrase n-grams, had the best score among them.

3.3 NERSO

Hakimov et al. [5] present an entity tool NERSO, which uses the same database as DBpedia Spotlight. The entity tool is specialized for identifying persons, organizations and locations. The spotting phase employs a 'window sliding' approach to find the words, also collecting candidate links. It starts by collecting a certain number of words and looking them up in the database. If there is no match it increases the window from

the right and repeats the operation until a match is found. This technique is the idea behind our proposed method for discovering potential entity links for suspect phrases in the link-detection phase. The disambiguation phase employs a graph method to weight the words spotted in the spotting phrase. The graph uses the Wikipedia page links to construct a relationship graph between the spotted words. From the calculation of centrality nodes and in and out edges, a score is established that is used to elicit the correct meaning of the word. The technique of weighting the words is the basis for calculating the local relatedness used in the proposed method.

3.4 Tulip

Lipczak et al. [7] developed the entity tool called Tulip, the first-place winner in ERD Challenge 2014. They used a combination of several datasets – Wikipedia, Google's Wikilinks corpus, Freebase, DBpedia and Wiktionary – to construct an efficient service. The spotting phase employs Finite State Transducers (FST) as a basis. The input document uses tokenizer on the words, where there words are iterated over. For each token, a cursor is created, which is traversed to FST. One phrase may involve multiple cursors, where the selection of correct cursor is made in the disambiguation phase. Calculation of the spotted words is done with cosine relation, where each word get a score based on how related it is to the document in question. The cosine relation is extracted from the service Sunflower, which is a Wikipedia-based semantic network developed in parallel with Tulip. The Sunflower service/database differs from others in how a surface form is rated. For instance the importance of Barack Obama as the President of United States of America is higher than Barack Obama as a Grammy Award-winning artist.

3.5 Seznam Research

Eckhardt et al. [2] presented their Entity Recognition Based on the Co-occurrence Graph and Entity Probability in the ERD 2014 Challenge. This method is based on DBpedia and uses Wikipedia as a knowledge base. Dbpedia uris are used for identification of entities. Seznam is a model-based system, thus testing of different technique are easily. The spotter phase technique is restricted to detect all words with at least one capitalized letter. Each sub-sequence in the mention has a list of potential candidate links. Disambiguation consists of three evaluations to clarify the correct meaning of the word or phrase: mention probability, co-occurrences and entity probability. Each word or phrase has a mention probability, which is the computed probability that there exists an

entity. Co-occurrence calculates the local relatedness and Wikipedia link connections of two words mentioned in a context. A graph is made, based on the entity's page-rank, where the entity is selected based on the weight of the link between mentions. Entity probability calculates the connection between two entities, that a mention leads to an entity.

3.6 ERD'14 evaluation

The Entity Recognition and Disambiguation (ERD'14) Challenge [1] is a benchmarking evaluation platform where developers could participate and compete to make the best entity linking tool. The winner was invited to the SIGIR 2014 workshop to present the work. Although the competition is over now, the evaluation service has been continued, to enable other developers to evaluate their work. There are two separate evaluation tracks: one for short text, like tweets and search queries, and another one for long text (full documents); our focus in this report is on long text. The ERD evaluation gives developers feedback on their performance and allows for a comparison with the other competing entity systems, based on both effectiveness (F-measure) and efficiency (latency).

4

Datasets

Entity linking needs a reference knowledge base that the spotted entity words can be linked up to. A good knowledge base contains rich information about a given entity, including type information, properties, various name variants, and relationships to other entities. A further important component in entity linking is a dictionary (or lexicon) of surface forms that contains word or phrases with a list of potential candidates for each; the entity linking tool can then utilize this for efficient look-up. In the proposed method, two kinds of dictionary datasets used: one that contains 'surface forms', and another containing 'worthy' entity words. Without this dataset it would not be possible develop an entity tool, with its efficiency performance and effectiveness. MongoDB is used to store the datasets, where each dataset is stored in different database. The entries in the database are in JSON format shown in Figure 4.1.

4.1 Knowledge base

To get an entity link service to work, there must be a knowledge base to link to, but also for spotting proper nouns and disambiguating them. On the market today there are several knowledge bases, including Freebase, DBpedia, Wikipedia, BablNet and Wikidata. All of them contain a rich encyclopedia of knowledge, structured and unstructured.

For the method proposed here Freebase has been chosen, because of its fast-growing knowledge and compatibility with the ERD'14 Challenge. Since the announcement that Freebase was to be retired, shifting to Wikidata has been discussed. However, the already developed dataset with Freebase-MID and ClueWeb content has good efficiency and is omitted. This is dealt with in more detail in Chapter 8, on future development.

4.2 Target entities

The ERD'14 Challenge further restricts the set of entities to be annotated [1]. It is a snapshot of Freebase from 3 December 2014, where only entities with associated English Wikipedia pages are considered. The list is further filtered on the basis of specific Freebase types that are wanted: locations, organizations and persons. The dataset consists of _id, name, wiki as Freebase-MID, surface forms, and Wikipedia page title. The ERD Challenge evaluation has built the golden standard of annotations using this entity snapshot; this is also what we use.

4.3 Surface forms

The dataset 'Surface forms' is an entity surface form dictionary (ESFD) and is the most important dataset that is used. The ESFD has been obtained from Hasibi et al. [6] in their proposal to ERD'14 Challenge for finding entities in short text. In the ESFD each surface form serves as the *key* and has the following information attached as *value*: source, entity identifier, and frequency. The surface forms is a phrase or a word, and is the look-up key. Source may be <rdfs:label>, <foaf:name>, 'facc09' and 'facc12'. Further, <rdfs:label> and <foaf:name> correspond to different DBpedia name variants. 'facc09' and 'facc12' are Freebase annotations of large web corpora, ClueWeb09 or ClueWeb12, created by Google (FACC) [3]. ClueWeb12 is the successor to Clueweb09, and is a dataset of more than 730 million English web pages . The FACC annotations contain the Freebase identity and the frequency value of how many times it was linked to a particular Freebase URI. This can be seen as ESFD links with its frequency value. The Freebase identity, MID(Machine IDentification), is the primary identity for entity link to the knowledge base, as in www.freebase.com/m/0z90c is /m/0z90c the Freebase-MID and reference to eBay. ESFD frequency value gives a rough estimate of which ESFD links(Freebase-MID) that is correct to a given word or phrase. Figure 4.1 shows a snippet from eBay, where 'facc12' object has various entities with their frequency values that may be linked to the surface form 'ebay.'

```
{
    "_id":"eBay",
    "<foaf:name>":{
        "<dbpedia:EBay_(song)>":1
    },
    "facc12":{
        "<fb:m\u002e01btsf>":12457, PayPal
        "<fb:m\u002e0z90c>":2616477, eBay
        "<fb:m\u002e019rl6>":846, Yahoo!
        "<fb:m\u002e01jbs9>":51, Nielsen ratings
        },
    "facc09":{
        "<fb:m\u002e01btsf>":11742, PayPal
        "<fb:m\u002e0z90c>":2567371, eBay
        "<fb:m\u002e01f6pt>":49, For Dummies
        "<fb:m\u002e027m38y>":3, Vermont Country Store
        }
}
```


FIGURE 4.1: Snippet of eBay entity in ESFD

5

Proposed method

This chapter explains our proposed approach for entity linking in news articles. This method is tested up against the ERD Challenge 2014 service and compared with state-of-the-art methods. Our method focuses on news articles and annotating meaningful words. Entity linking is performed in three steps: spotting, link detection and disambiguation. Unlike other methods, the spotter detects all words or phrases containing capitalized letters. This technique offers a means of detecting entity words quickly, thus saving considerable time. Link detection has a window sliding approach presented in Hakimov et al. [5] for detection of entity words in sentences or phrases where one or more letters are capitalized in all the words. A new relatedness measure is introduced in the disambiguator, based on the spotted words FACC. It differs from Eckhardt et al. [2], Mendes et al. [9] which present a word relatedness based on entity knowledge base. Documents are fed into the method one at a time, and final entity-linked document is made. The outcome of the application can be chosen from three different formats: HTML document, where entities are linked to Freebase Figure 5.1; TSV file, where the entity mentions are listed with specifications shown in Figure 5.2; or a plain text document with squared brackets around entity words or phrases shown in Figure **??**.

The three steps in the entity linking process shown in Figure 5.4 are described later, with details of implementation and challenges. First a description of the working process

FIGURE 5.1: Snippet showing the HTML output

textID	Begin offset	End offset	Freebase-MID	Freebase URL	Mention word	Score 1	Score 2
msn-2014-03-28-00098	34	48	/m/050gdt	https://www.freebase.com/m/050gdt	Dermot O'Leary	1	0
msn-2014-03-28-00098	84	98	/m/01pfkw	https://www.freebase.com/m/01pfkw	Simon Cowell's	1	0
msn-2014-03-28-00098	215	218	/m/0687w9	https://www.freebase.com/m/0687w9	OK!	1	0
msn-2014-03-28-00098	268	274	/m/050gdt	https://www.freebase.com/m/050gdt	Dermot	1	0
msn-2014-03-28-00098	320	325	/m/01pfkw	https://www.freebase.com/m/01pfkw	Simon	1	0

FIGURE 5.2: Snippet showing the TSV output

[Dermot O'Leary] has revealed he hopes to follow in [Simon Cowell's] footsteps and become a father soon.
The X Factor host has been married to Dee Koppang for two years and admitted to [OK!] magazine they are thinking of starting a family.
[Dermot] said: "I definitely won't be a father before [Simon], unless there's a miracle on its way! But I hope sometime soon, yeah."

FIGURE 5.3: Snippet showing the plain text output

in this proposal.

5.1 Workflow

This work started with a pre-project for finding an efficient method for spotting entity-bearing words in news article Roalkvam [13]. An examination of potential words and phrases showed that many of the entity words had capitalized (uppercase) letter as a common factor.

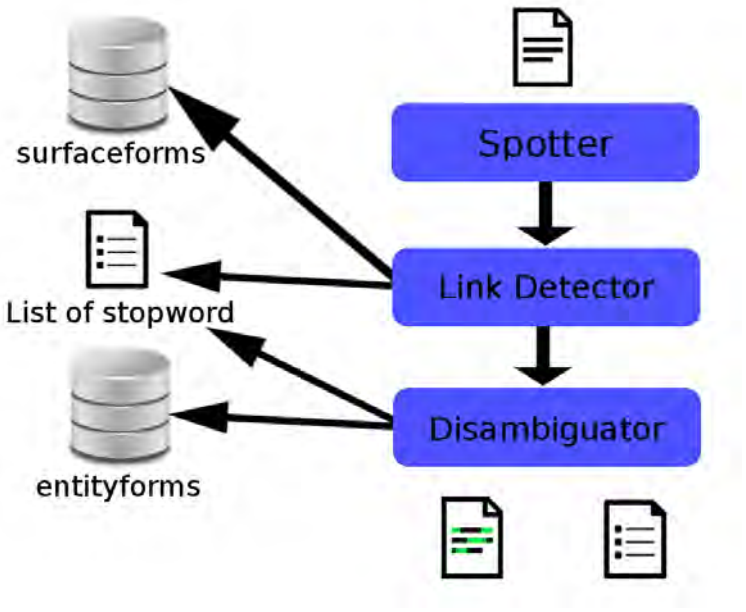

FIGURE 5.4: Illustrating how the components are working together

In this pre-project a method for detecting capitalized letters, and storing them as an annotation was implemented. The pre-project worked well, with good results in efficiency and effectiveness. It was tested against DBpedia Spotlight and Wikipedia Miner spotters, where all three methods analysed ten news articles. The annotation of the three methods was evaluated against a golden set manually constructed on this news articles. The pre-project achieved a score at 57% precision, against DBpedia with 35%, and Wikipedia miner with only 7%. A latency on the spotter was calculated, where the pre-project had an average 27.29ms, as compared with 72.13ms for DBpedia and Wikipedia Miner with 5 minutes average.

The pre-project method for detection of words is used as a basis in the proposal presented in this thesis. A simple link detection and disambiguator on commonalities was implemented to be capable of testing against the ERD'14 service. The F1-score generated from ERD'14 was then at 40%, showing there was some flaw that has to been fixed and improved. It should also be noted that in the pre-project the golden standard and the annotation method were created by the same person, so the golden annotations might be biased. That is no longer a problem, as we use the ERD'14 platform here.

In the earlier version there was a limit on four capitalized word after each other. This has been fixed with a recursive method that detects all capitalized letters in a sequence. Further, we have implemented a line-break detector, a better end-of-sentence detector, and a check if the word is a conjunction word. Common word filters (this, is, etc.) used in the search for capitalized letters have been moved to the disambiguator. In the pre-project each annotation was stored in a hash map with the mention text as key and

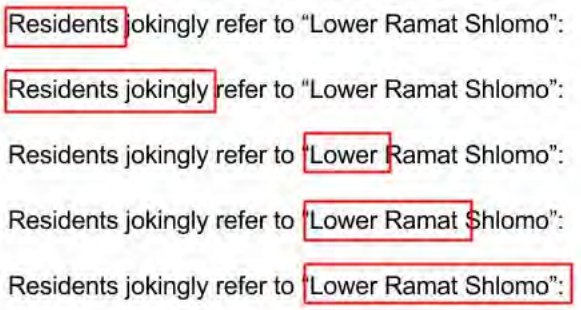

FIGURE 5.5: Image process of finding capitalized word

the begin offset as value. Where in this proposal each annotation is stored in an object, a list contains all the object-annotations for a document. With these improvements and the implementation of better link detection and local relatedness in the disambiguator, the F1-score increased to 64.5%.

5.2 Spotter

In the spotter the text is analysed for words or phrases that may be potential entity links for the document. The text is split up into a list by white spaces or line breaks; this list is looped over in detection of capitalized letters. A recursive function detects capital letters in a word; where there is a match, it check the next word also. In 5.5 the two first line Residents is chosen as a word, because the next word is not capitalized. This method of detecting capital letters has some exception. First it will only check the next word if the analyzed word is not at end of sentence or if it contains a line break. Second is not capitalized word, if a detected capital letter is followed by a conjunction ’and’ or by the preposition ‘of’ it will go on to the next word – as with *Library of Alexandria*, or *Tom and Jerry*. Algorithm 1 presents an overview of the recursive method, where the array list of words is sent in and the word’s position in the array. The detected word or phrase is a potential annotation, and is saved together with its offsets in an array list as an annotation. The class annotation contains information about a spotted word or phrase, as offsets, candidate links and other parameters used in the analysis. Words and phrases are ambiguous at this stage of the process. There may be sentence with all words capitalized. Link detection resolves/disambiguates this problem.

Algorithm 1 countCapitalizedWord

```
1: procedure COUNTCAPITALIZEDWORD(w, i)          ▷ w List of words, i = Offset
2:     set ← 0
3:     if i < w.length then
4:         word ← w(i)
5:         if word ∈ capitalized OR bindingwords then
6:             if startswithsymbol ∃word then
7:                 return set
8:             end if
9:             if linebreak ∃word then
10:                return set+1
11:            end if
12:            if endOfSentenc ∃word then
13:                return set+ 1
14:            end if
15:            if pluralApostrophe ∃word then
16:                return set+1
17:            end if
18:            return set =(1 + countCapitalizedWord(w, i+1))
19:        else
20:            return set
21:        end if
22:    else
23:        return set
24:    end if
25: end procedure
```

5.3 Link Detection

In link detection the task is to connect the spotted word or phrases to a knowledge base. Each annotation detected in the spotter is looked up in the ESFD. Figure 5.6 illustrates the path for a word or phrase in the process identifying knowledgebase candidate links. It starts with a query containing the spotted word or phrase to the Mongo database containing the surface forms dataset. When there is positive feedback, a cursor returns a JSON object with information on the word or phrase. This information is added to the annotation object of the word or phrases. In this information there is an ESFD link-set with the corresponding frequency. Only links with a frequency above a given threshold are selected. This threshold is referred to as the main threshold in this application. A negative feedback returns a null cursor; the word is then cleansed of symbols for another try. If feedback remains negative, the annotation is forwarded to a window sliding function for resolving suspect annotations. Figure 5.7 shows how the method works, where 'VICTORIA CERAMICS' got a match in the data-base. It then starts with the remainder of the phrase, to detect further possible entities. This method of

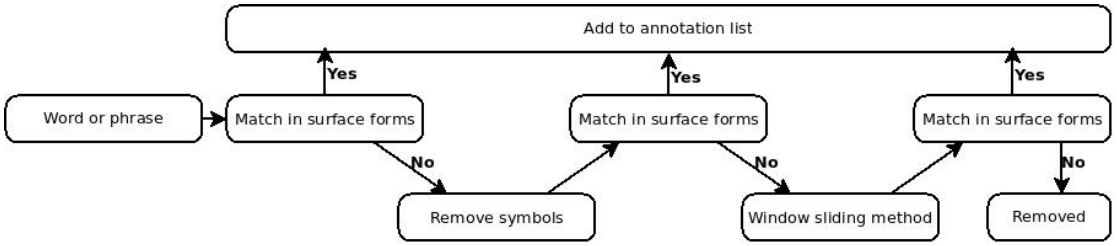

FIGURE 5.6: Link detection word path overview

FIGURE 5.7: Window sliding approach for detecting suspect phrases

solving suspect word is employed by Hakimov et al. [5] for the detection of words or phrases in a document.

Algorithm 2 shows the pseudocode for this process of detecting entities in phrases. When a word is found in 'solveSuspect' the same steps are followed by collecting the ESFD link-set. The offset of the word is still from the original phrase, so a new calculation of offset is needed. On finding the word offset in the original phase, it is added this to the original offset.

5.4 Disambiguation

The words or phrases are now ambiguous, with several knowledge base links from the ESFD that can be the correct ones. The disambiguation step evaluates each word or phrase to find the correct link, based on the context in which it is used. To disambiguate the right meaning of the word or phrase, two methods are used. One is based on the ESFD link frequency, 'commonness', the other is on the local relation between the word or phrases ESFD links. The first method sorts the ESFD link frequency for each annotation in descending order, where the highest score is chosen to be the temporary

Algorithm 2 solvsuspect

```
1: procedure SOLVESUSPECT(s, a)                    ▷ s Spottedword, a = Annotation
2:     w ← split s                                                  ▷ split on white-space
3:     db ← surfaceforms database
4:     j, i, x, y ← 0
5:     annotation ← Annotation list
6:     if w.length ≥ 1 then
7:         while i < w.length do
8:             while j < w.length−y do
9:                 testphrace ← w[j]
10:                j++
11:            end while
12:            i++
13:            y++
14:            if testphrase∃stopWord then
15:                testphrase ← newString
16:                y ← textit0
17:                x ← j
18:                i ← x
19:                Continue
20:            end if
21:            if testphrase∃db then
22:                annotation ← testphrase
23:                y ← 0
24:                x ← j
25:                i ← x
26:            else if testphrase split white-space = 1 then
27:                testphrase ← newString
28:                x++
29:                y ← 0
30:                j ← x
31:                i ← x
32:            else
33:                j ← x
34:            end if
35:        end while
36:    end if
37: end procedure
```

Freebase-MID. Computed as; $\alpha = maxf(x1), \ldots, f(xn)$, where α is the maximum frequency value of ESFD links, and $f(x)$ the frequency in the ESFD link-set. The second method is a novel local-relatedness measure, where the idea is taken from a graph-based disambiguation in Hakimov et al. [5]. It uses the hyperlinks in a Wikipedia article to make relations to other entities. In our proposed method, it uses the annotations ESFD links to measure relatedness. These ESFD links are a collection of entity links that the annotation can refer to, where some of the links are also mentioned in another annotation set of ESFD links. For example, in Figure 5.8 Italy and Italian are two spotted words. The edges mean that Italy has an ESFD link that is mentioned in Italian ESFD link-sets, shown in Figure 5.9. Local relatedness uses this connection in determining whether if there is an entity link that has a lower score on frequency that should be chosen, because of the surrounding context.

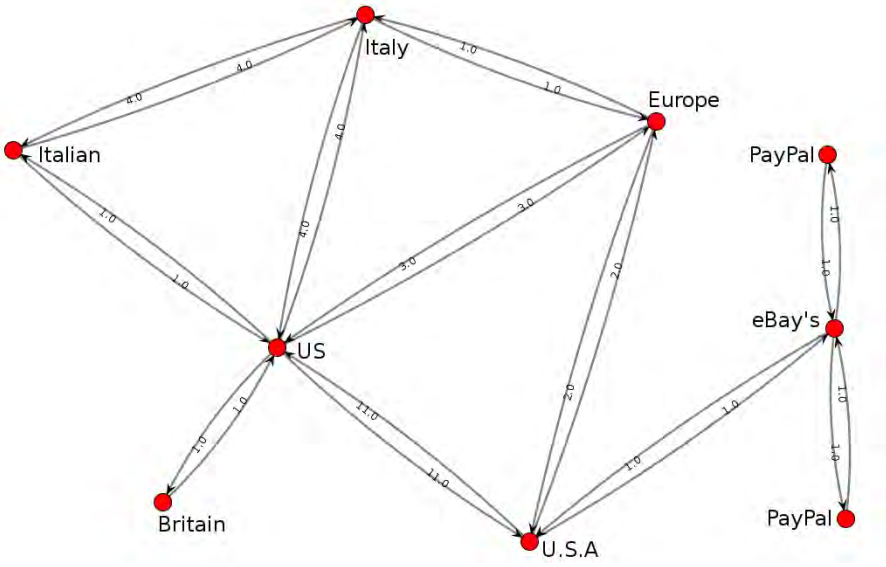

FIGURE 5.8: Relation between different annotations

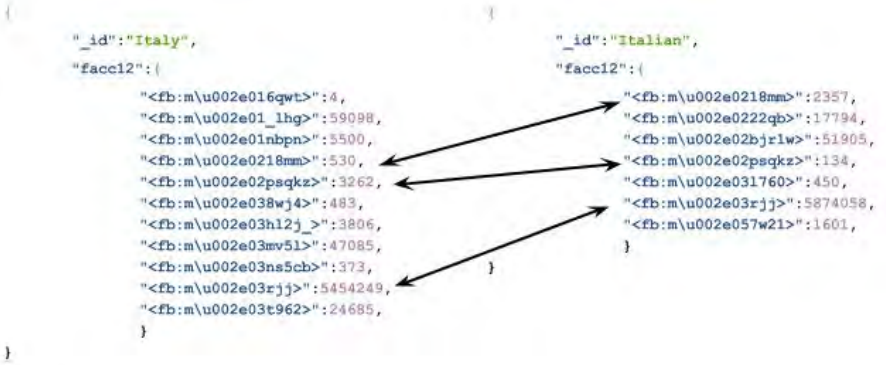

FIGURE 5.9: Common Freebase-MID in ESFD link-set

Take, for instance an article about Clive Palmer and Rupert Murdoch, Table 5.1. When only 'Palmer' is mentioned in the document, the frequency score selects an ESFD link that reference to the town in Alaska. The selection of ESFD link frequency is this case wrong, but can be corrected with local relatedness, where entity link to Palmer is mentioned in the other annotation's ESFD link-set. Likewise, when Rupert Murdoch is mentioned as only 'Rupert', the frequency selection choose Rupert Grint, which is erroneous but is corrected with local relatedness.

TABLE 5.1: Correction of entity link with local relatedness

	Frequency selected		**Local relatedness selected**		
Word	**ID**	**Frequency**	**ID**	**Frequency**	**Mention**
Palmer	/m/01z1_lp	32387.0	/m/07kdvrf	7338	2.0
Rupert	/m/014xzx	9693.0	/m/06hrk	7411	5.0

In formal terms, the local relatedness method can be described as follows; Let $F = \{f1...fn\}$, and Let$\hat{F} = \{\hat{f}1...\hat{f}n\}$ be a set of ESFD links for two different word or phrases. Each fn has a score value equal zero, if $f1$ matches one $\hat{f}n$ in set $\hat{F}$ there the score value for $f1$ increases by one. The one fn with highest score value is selected only if it is higher than given local relatedness threshold. Local relatedness is computed with a percentage of the max-score in ESFD link-set. If it is lower than local relatedness threshold, the Freebase-MID with highest frequency is chosen. Algorithm 3 shows the pseudocode for local relatedness.

In the end of disambiguator right-hand side of Figure 5.10, the annotation list is cleansed of annotations that do not contain a Freebase-MID, remove all common words, or its Freebase-MID is not contained in the dataset entity forms.

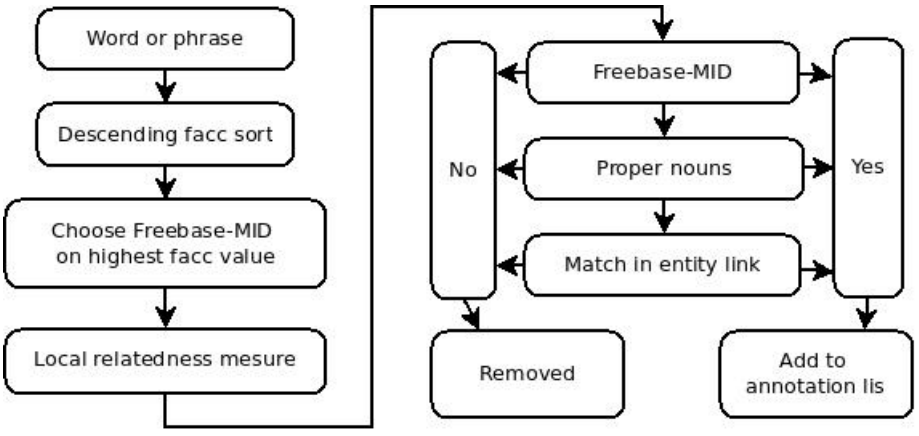

FIGURE 5.10: Disambiguator word path overview

Algorithm 3 localRelatedness

```
1: procedure LOCALRELATEDNESS(a)                                  ▷ a annotations list
2:     i ← 0
3:     j ← 0
4:     localRelatedness(id, score)
5:     for a.length do
6:         F ← a[i].candidatelinks
7:         while F.hasNext do
8:             id_1 ← fn.linkID
9:             if  thenid_1.frequency > local relatedness threshold ∗ max(F.frequency)
10:                for a.length do
11:                    if i == j then
12:                        Continue
13:                    end if
14:                    F̂ ← a[j].candidatelinks
15:                    if id_1 ∈ F̂ then
16:                        localRelatedness ← (id, score + 1)
17:                    end if
18:                end for
19:            end if
20:        end while
21:        a[i] ← localRelatedness
22:    end for
23: end procedure
```

6

Experimental Setup

The system has been tested and evaluated against the ERD'14 Challenge which was held from March to June 2014 [1]. The challenge had two tracks: one for long text, like news articles or web pages, and another for short text, like tweets or search queries. Systems could compete in either or both of the tracks. After the challenge was over, the service was kept available for testing and evaluation of entity linking of documents. This gives the developers the opportunity to develop entity systems and test them against state-of-the-art systems. To use the system, a developer must set up a service where ERD can feed the entity tool with documents. These documents are then processed by the developers system and the annotation words are sent back to ERD for evaluation against the golden standard. The test-set contains 150 documents, equally divided on random web pages and msn news articles. All documents have been stripped of HTML codes; there is no additional process for cleaning up the text before it is annotated. The 'golden standard' is a set of annotation with the best words for a document, given the proper knowledge base link. This golden set is generated from the dataset provided by ERD, given in Chapter 4. The golden standard used in this evaluation contains data from 51 of these documents: 26 from random web pages and 25 from msn news articles.

In the experimental setup the documents from the ERD are sent to a Maven server. With an HTTP POST method on the Marven server the ERD server sends a document,

asking for the annotation to be returned. The annotations are sent back to the EDR service in a TSV format, as shown in Table 6.1, which is required from ERD. Scores 1 and 2 are used in evaluating system confidence.

TABLE 6.1: ERD'14 TSV form

DocID	**Begin-Offset**	**End-Offset**	**Freebase-mid**	**Freebase-url**	**Mention text**	**Score 1**	**Score 2**
mainbody-00003	18	23	/m/03_3d	https://www.freebase.com/m/03_3d	JAPAN	0	0
mainbody-00003	1823	1827	/m/0z90c	https://www.freebase.com/m/0z90c	eBay	0	0

6.1 ERD Evaluation

The program is evaluated on latency and F1-measure combination of precision and recall. Let $\hat{A} = \{\hat{W}1...\hat{W}n\}$ be a set of annotations in a document, where Let $\hat{W}x = \{\hat{B}, \hat{E}, \hat{id}\}$. $\hat{B}$ is the begin-offset, $\hat{E}$ is the end-offset, and $\hat{id}$ is the identifier of the entity in the knowledge base . For the golden set there will be a same set $\hat{A} \rightarrow A$. Since the ambiguous of the word is the challenging task, it is evaluated on the correctness of $\hat{id}$. Thus the offset is correct if $\hat{W}x$ is in the boundaries of W. Precision is calculated on the number of matches A in $\hat{A}$ divided by the number of A, (6.1). Recall is calculated on number of matches A in $\hat{A}$ divided by the number of $\hat{A}$, (6.2). The F1 score is a measure between precision and recall, (6.3), where the final evaluation metric is defined as the micro-averaged F-measure.

$$Precision = \frac{\sum Match(\hat{A}, A)}{\sum A} \tag{6.1}$$

$$Recall = \frac{\sum Match(\hat{A}, A)}{\sum \hat{A}} \tag{6.2}$$

$$F1 - measure = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \tag{6.3}$$

Latency is measured as the average processing time for a document. The timer is started when a document is sent from the ERD'14 service to the user server hosting the entity application. When ERD'14 receives the TSV table the timer is stopped, and run time is calculated. However, as network delays are included in the run time, this does not provide the most accurate estimate of latency.

7

Results and Discussions

In this chapter presents the results based on various different settings given to the system, followed by a discussion. It is tested for threshold for candidate links, solving suspect phrases, and a local relatedness threshold for disambiguation. On all tests, plurals, apostrophes and symbols are removed from the word or phrases. The tests are divided into groups where the test condition shows what setting changes. All tests have been conducted on PC with 8G RAM, Core i7 2.5GHz CPU. The maximum score for F1-measure, precision and recall is 1, so the given number can be translated to percentages. The result is then compared with other submissions to the ERD'14 Challenge. There is also a subsection on error analysis, where the disambiguation method is be analysed for errors in determining the correct entity link.

7.1 Candidate link test

Here the threshold for ESFD link frequency to a word or phrase is tested. (See Table 7.1 for results.) The different thresholds determine which entity link from ESFD link-set should be selected (see Section 5.3). In the ESFD, a word or phrase can contain entity links with a frequency score all the way down to 1. With a variance in threshold, the amount of candidate links will vary. There will also be some word or phrases that will

end up with an empty candidate link-set, and will not be annotated in the document. This will also have an impact on the local relatedness measure in the disambiguator. For this test the local relatedness threshold is set to 5%, from the best F1 score in Table 7.2. Taken into count the latency is higher with 1%, as there is more ESFD links to investigate. Results shows that a threshold of 50 yields the best F1-score. This means that any entity link with a frequency score lower than this is not likely to be a relevant entity link, nor should it have an impact on the local relatedness score. On the other hand if the threshold increases above 50, there are words or phrases that are not annotated, because the candidate link have a frequency score below the given threshold.

TABLE 7.1: Candidate link threshold test

Test condition	**F1 Score**	**Precision**	**Recall**
10	0.6147	0.6057	0.6240
50	**0.6457**	0.6231	**0.6701**
100	0.6403	**0.6235**	0.6582

7.2 Disambiguation test

This involves testing the threshold for local relatedness evaluation. The threshold controls whether an entity link mentioned in other annotations should be chosen (see Section 5.4). Results from test on the local relatedness threshold are shown in Table 7.2, where the main threshold is 50. This test is intended to show which frequency score the mentioned entity link should have, in order to be relevant to the word or phrase. If a word or phrase has an entity link with a low frequency score that has several mentions in other annotation, it is not necessarily the correct entity link. Results shows that with increasing or decreasing the local relatedness threshold has an impact of final score. A threshold of 5% gives the best score, taken into count the higher latency for a local relatedness threshold at 1%. This means that an entity link with a score under this value is to be true, but since our goal is a good latency this is not chosen. When the threshold increases above 5%, the number of entity link correction decreases, and the context-related entity link is not selected. Local relatedness disambiguation raises the F1 score 4.5% from 60% to 64.5%. Precision and recall has a improvement from 59.4% to 62.3% and 61.6% to 67%.

TABLE 7.2: Local relatedness threshold test

Local relatedness threshold	**F1 Score**	**Precision**	**Recall**
1%	**0.6503**	**0.6298**	**0.6721**
5%	0.6457	0.6231	0.6701
10%	0.6330	0.6121	0.6554
Off	0.6052	0.5948	0.6161

7.3 Latency

The latency test is conducted with five consecutive runs, where the conditions of the best results in Table 7.1, and Table 7.2 are used. Running the test several times will yield better estimates on latency. Network delays and first run on the algorithm may yield a higher latency. A first run with a higher latency could also be 'cold start' where common words are loaded into the memory. The differences is negotiable as it is under 0.02 seconds. Results show that the first run of the algorithm takes more time (see row one, Table 7.3). Thereafter it stabilizes to an average of 0.49 seconds.

TABLE 7.3: Latency

Run number	**Latency**
1	0.5054
2	0.4880
3	0.4889
4	0.4922
5	0.4945
Average	0.4938

7.4 Comparison with ERD approaches

7.4.1 Efficiency

Table 7.4 presents a top five list of applications with best latency, where the proposed method is in third place. This is with the best run time of the program. The time 0.48 is 0.19 seconds behind the leader, and 4.36 seconds faster than the overall average 4.85 seconds. This means that it is competitive with the best state-of-the-art entity linking

program on time, which was the main goal of our project. In Appendix A.2 the whole list is shown, in ascending latency.

TABLE 7.4: Ascending latency ERD'14 top 5

Rank	**Team**	**Name**	**Expected F1***	**Latency**
1	MLNS	Tulip 1.01	0.7829	0.2958
2	Rylko	ookb_apriori0.5	0.4744	0.4658
3	**ELNA-B**	**elnab.v4.0.1**	**0.6457**	**0.4880**
4	Neofonie	final	0.6995	0.5318
5	WebSAIL	baseline_mk_IV	0.6873	0.6959

7.4.2 Effectiveness

Comparison of the best results with the official ERD systems in terms of effectiveness, the proposed approach ends up on the 10th place, which is equal with average F1-measure. This shows that there is room for an improvement on the F1-score for the proposed method. Table 7.5 gives a top five list of the best F1-score; the whole list can be found in Appendix A.1.

TABLE 7.5: Ascending F1-score ERD'14 top 5

Rank	**Team**	**Name**	**Expected F1***	**Latency**
1	MLNS	Tulip 1.01	0.7829	0.2958
2	MS_MLI	NEMO v.ERD2014	0.7599	1.4899
3	Seznam Research	lp-graph-w retest	0.7193	2.3297
4	NTUNLP	mg++,2,0.2,0.0,0.3	0.7137	7.6573
5	HITS	E	0.7045	4.9694
10	**ELNA-B**	**elnab.v4.0.1**	**0.6457**	**0.4880**

7.5 Error analysis

Errors have been discovered in the development and testing of the system. Some errors could be fixed; others occur because the system method or golden set from ERD is wrong. Error that was detected concerned the ERD dataset on which the golden standard is based. The golden standard annotations for the document that are tested are available on ERD'14 Challenge webpage, and can be downloaded for analysis. In error analysis this golden standard is used to compare which word is annotated, and which Freebase-MID it has. In analysing the annotations, errors were detected, where annotations in

the golden standard did not occur in the dataset. For instance the word 'Congress' the proposed method recognize and refers to the United States Congress, but its Freebase-MID is not listed in ERD Challenge dataset. Since it is not in the dataset, our proposed method does not annotate it. In the golden standard it is annotated with a reference to the Indian National Congress: thus it cannot be true that the golden standard is based solely on the ERD dataset. In the dataset one finds Freebase-MID for Congress Arizona and Congress Ohio Figure 7.2, but in document Figure 7.3 there is nothing to indicate whether it is Congress Arizona or Congress Ohio.

> *Coming up after a wave of resentment swept the nation in the aftermath of countless scams, Uttar Pradesh, the sprawling epicenter of India's influential cow-belt, was high-priority for* ***Congress*** *more than any other party. The Lok Pal bill related agitations and the Team Anna campaign divided the polity in such a way that that there was no clarity on who stood where and for what. As expected the contentious bill generated much sound and fury but failed to pass muster in Parliament. The* ***Congress*** *tried to project the image that it has actually tried to bring in the legislation, which was thwarted by the Opposition parties. The audience India's grand old party was trying to address during the debate was definitely the Uttar Pradesh electorate, more than anyone else, as Team Anna had made it clear in unequivocal terms that it would campaign against* ***Congress*** *in poll-bound states. Another compelling reason for* ***Congress*** *to go for the kill in UP is Rahul Gandhi, but no prizes for guessing why the younger Gandhi invested so much in UP, which sends the maximum number of lawmakers to Lok Sabha.*

FIGURE 7.1: Snippet of context around CMS

```
> db.entityforms.find({"name":"\"Congress\"@en"})
{ "_id" : "/m/0qr15", "name" : "\"Congress\"@en", "wiki" : "\"/wikipedia/en_title/Congress$002C_Arizona\"\r" }
{ "_id" : "/m/0z355", "name" : "\"Congress\"@en", "wiki" : "\"/wikipedia/en_title/Congress$002C_Ohio\"\r" }
```

FIGURE 7.2: Snippet of target dataset with Congress search

The (US) Centers for Medicare and Medicaid Services also later mentioned, as CMS (see Figure 7.3) is another instance of this error. The system recognizes CMS and references it to the correct Freebase-MID, whereas in the ERD dataset there is no instance on this Freebase-MID.

> *The DRA directed the Department of Health and Human Services (HHS) to establish a Medicaid Integrity Program (MIP). The MIP is designed to provide the* ***Centers for Medicare and Medicaid Services*** *(**CMS**) the resources necessary to combat Medicaid fraud, waste and abuse. Congress appropriated $5 million for FY 2006 with an additional $50 million in each of 2007 and 2008 and $75 million in 2009 and annually thereafter to carry out the operations of MIP.*

FIGURE 7.3: Snippet of context around CMS

The last error detected is when a golden standard for a document contains only lowercase entity words. The system has not been developed for detecting lowercase, and thus does not annotate this. In total of 1166 annotated gold standard words, there are only 17 of cases where the word is in lowercase. In one of these 17 instances, there is a single annotated word in the document; the evaluation score for this document then becomes zero. In Chapter 8 below, Future development, there is a section that explains various difficulties involved in detecting words with only lowercase letters.

8

Future development

This section presents a brief summary of what could be done in further developing the system to make it even better.

The first development should be a relatedness disambiguation that is not based on local entity words. The most common method for calculating word relatedness involves using the Normalized Web Distance, showed in (8.1). Where x and y are two words of interest, $f(x)$ is the number of webpages that contain concept x, likewise for $f(y)$. N is the total number of webpages. This method has shown good results in disambiguation of words, but it has some flaws. When calculating the first word, the other words are still ambiguous, so there is a cyclic problem. Godin et al. [4] use this method to implement a Normalized Freebase Distance, where x and y are the number of concepts links and N is the total number of concepts in Freebase. Milne and Witten [11] propose a similar method, where x and y are two Wikipedia articles of interest, $f(x)$ and $f(y)$ are the number of articles that link to x and y, and N is the total number of Wikipedia articles.

$$relatedness(x,y) = \frac{max\Big(log\big(|x|\big)\, log\big(|y|\big)\Big) - log\big(|x| \bigcap |y|\big)}{logN - min\Big(log\big(|x|\big), log\big(|y|\big)\Big)} \tag{8.1}$$

Another future development would be to exchange the Freebase database with the Wikidata database. Since 16 July 2010 when Google bought Freebase from Metaweb [10] it has been a path of Google's Knowledge Graph, and on 16 December 2014[1], it was announced that Google/Freebase had decided to shut down Freebase and would be transferring its data to Wikidata.

Wikidata, launched in October 2012, is a fast growing knowledge base with an active community of contributors. Vrandečić and Krötzsch [14] explain that every Wikidata entity is identified by a unique URI – such as http://www.wikidata.org/entity/Q42 for item Q42, Douglas Adams. Thus a new database with Wikidata should not entail having to rewrite the whole system.

Further, detection of lowercase entities would increase the effectiveness of the system. Such an implementation will entail some challenges as to efficiency of the system. Hakimov et al. [5] present a method where lowercase entities are spotted, with a window sliding technique that requires more time. They start with a selected window size of word and look it up in a database if there is an entity. If no entity is found , the window size is increased by one word from the right-hand side. This procedure continues until a match is found, and then the window is moved so it starts *after* the spotted entity. This type of searching is time-consuming and will reduce the efficiency of the application.

[1] https://plus.google.com/109936836907132434202/posts/bu3z2wVqcQc

9

Conclusions

This thesis has presented an efficient entity-linking tool for news articles. The work has involved challenges in analysing unstructured texts, and connecting them to a knowledge base. It has been shown that a good Name Entity Recognition(NER) makes it possible to spot the correct words, which provides a good starting point for link detection. Using a window sliding method for resolving the problem of suspect words has been shown to be efficient. Phrases with many uppercase letters are solved by detecting which word is an entity in the given context. Disambiguation of the words based on the frequency mention score works in most cases, but occasionally the wrong entity link is chosen. Here it is shown that using local relatedness between the spotted entity helps in selecting choosing most correct entity links. This local relatedness improves the effectiveness of the system, and the relation between entities will have an positive effect. For further entity applications, relatedness measurement is recommended for system effectiveness of the system, and not over-reliance on the entity link frequency mention.

The method proposed here achieved good latency, outperforming various other solutions. That was the main goal: an efficient approach for entity linking news articles. Our method yielded a latency of 0.48 seconds – an impressive 4.36 seconds faster than the average time of submission to the ERD‘14 Challenge.

A

ERD Challenge '14 Scoreboard

A.1 F1-Mesure

Microsoft Research

Welcome, bme.roalkvam@stud.uis.no | Sign out

Entity Recognition and Disambiguation Challenge

Home | My Team | Related | Servers and Papers | Rules | Datasets | Short Track | Long Track | Discussion

Follow progress on the leaderboard. The leaderboard shows participants' results based on Workshop submissions.

Top systems

All Systems | My Team's Systems

Rank	Team	Name	Expected F1 *	Latency	Comment
1	MLNS	Tulip 1.01	0.782932891466446	0.295766468	Timeout count: 0 Precision: 0.8175 Recall: 0.7512
2	MS_MLI	NEMO v.ERD2014	0.759887616165982	1.489869091	Timeout count: 0 Precision: 0.8332 Recall: 0.6985
3	Seznam Research	lp-graph-w retest	0.719340134577816	2.329690415	Timeout count: 0 Precision: 0.7928 Recall: 0.6583
4	NTUNLP	mg++,2,0.2,0.0,0.3	0.713715605965133	7.657305884	Timeout count: 0 Precision: 0.7571 Recall: 0.6750
5	HITS	E	0.704481489499892	4.969390719	Timeout count: 0 Precision: 0.7740 Recall: 0.6464
6	Neofonie	final	0.699549646150547	0.531759795	Timeout count: 0 Precision: 0.7600 Recall: 0.6480
7	WebSAIL	baseline_mk_IV	0.687343880099917	0.695859909	Timeout count: 0 Precision: 0.7219 Recall: 0.6559
8	Acube Lab	lastHope-14	0.671890303623898	0.861309805	Timeout count: 0 Precision: 0.8756 Recall: 0.5451
9	CI_ERD	i+offset_fix	0.647881170816951	0.716806402	Timeout count: 0 Precision: 0.7192 Recall: 0.5894
10	ELNA-B	elnab.v4.0.1_50_5%x2_1	0.645729605515128	0.488031788	Timeout count: 0 Precision: 0.6231 Recall: 0.6701
11	ExPoSe	dbp_sz_c0.3_s0_single	0.633681343622333	0.710359444	Timeout count: 0 Precision: 0.7390 Recall: 0.5546
12	UBC	UBC_long_86	0.631892015557081	37.288410906	Timeout count: 0 Precision: 0.7449 Recall: 0.5487
13	Acube Lab#2	tagme	0.621006900076667	1.470213044	Timeout count: 0 Precision: 0.8703 Recall: 0.4827
14	MLNS2	test27	0.610479375696767	2.352462021	Timeout count: 0 Precision: 0.6956 Recall: 0.5439
15	NERD	Test3	0.536408864767074	18.027728999	Timeout count: 2 Precision: 0.6226 Recall: 0.4712
16	Rylko	ookb_apriori0.5	0.474422620332398	0.465814121	Timeout count: 0 Precision: 0.5194 Recall: 0.4366
17	C3	C3.10	0.445610965296005	2.087787228	Timeout count: 0 Precision: 0.8377 Recall: 0.3035
18	ABELinkin	SystemTest	0	0	Timeout count: 0 Precision: 0

A.2 Latency

Microsoft Research

Welcome, bme.roalkvam@stud.uis.no | Sign out

Entity Recognition and Disambiguation Challenge

Home | My Team | Related | Servers and Papers | Rules | Datasets | Short Track | Long Track | Discussion

Follow progress on the leaderboard. The leaderboard shows participants' results based on Workshop submissions.

Top systems

All Systems | My Team's Systems

Rank	Team	Name	Expected F1	Latency *	Comment
1	ABELinkin	SystemTest	0	0	Timeout count: 0 Precision: 0 Recall: 0
2	MLNS	Tulip 1.01	0.782932891466446	0.295766468	Timeout count: 0 Precision: 0.8175 Recall: 0.7512
3	Rylko	ookb_apriori0.5	0.474422620332398	0.465814121	Timeout count: 0 Precision: 0.5194 Recall: 0.4366
4	ELNA-B	elnab.v4.0.1_50_5%x2_1	0.645729605515128	0.488031788	Timeout count: 0 Precision: 0.6231 Recall: 0.6701
5	Neofonie	final	0.699549646150547	0.531759795	Timeout count: 0 Precision: 0.7600 Recall: 0.6480
6	WebSAIL	baseline_mk_IV	0.687343880099917	0.695859909	Timeout count: 0 Precision: 0.7219 Recall: 0.6559
7	ExPoSe	dbp_sz_c0.3_s0_single	0.633681343622333	0.710359444	Timeout count: 0 Precision: 0.7390 Recall: 0.5546
8	CI_ERD	i+offset_fix	0.647881170816951	0.716806402	Timeout count: 0 Precision: 0.7192 Recall: 0.5894
9	Acube Lab	lastHope-14	0.671890303623898	0.861309805	Timeout count: 0 Precision: 0.8756 Recall: 0.5451
10	Acube Lab#2	tagme	0.621006900076667	1.470213044	Timeout count: 0 Precision: 0.8703 Recall: 0.4827
11	MS_MLI	NEMO v.ERD2014	0.759887616165982	1.489869091	Timeout count: 0 Precision: 0.8332 Recall: 0.6985
12	C3	C3.10	0.445610965296005	2.087787228	Timeout count: 0 Precision: 0.8377 Recall: 0.3035
13	Seznam Research	lp-graph-w retest	0.719340134577816	2.329690415	Timeout count: 0 Precision: 0.7928 Recall: 0.6583
14	MLNS2	test27	0.610479375696767	2.352462021	Timeout count: 0 Precision: 0.6956 Recall: 0.5439
15	HITS	E	0.704481489499892	4.969390719	Timeout count: 0 Precision: 0.7740 Recall: 0.6464
16	NTUNLP	mg++,2,0.2,0.0,0.3	0.713715605965133	7.657305884	Timeout count: 0 Precision: 0.7571 Recall: 0.6750
17	NERD	Test3	0.536408864767074	18.027728999	Timeout count: 2 Precision: 0.6226 Recall: 0.4712
18	UBC	UBC_long_86	0.631892015557081	37.288410906	Timeout count: 0 Precision: 0.7449

B

Installation Guide

This appendix presents a brief description of the use of the program and how it is set up against the ERD'14 Challenge service. The source code is attached to this paper, where there is one part for the ERD'14 service and another for local runs on the computer. The ERD'14 part is a Maven setup, where server is established that can receive post requests. Attached in the post request should be runID, textID, and text. runID is simply an identifier for which run it is. textID is the filename, and text is the text to be annotated. This will return a TSV file with the annotated word, in the same format as required by ERD'14. The code for setting up this service is taken from a post by Diego Ceccarelli for setting up a service for short track. This is modified to be suitable for long tracks, with changes in the annotated class. The only thing that is used is the code for connection with ERD'14 service: the entity system code is the same as the one that runs on a local computer.

For local computers there is a runnable jar file folder that can be used with the command; $java - jarELNAB.jarfolderpathoutputformat$. This needs two input arguments. First one is the folderpath, second one is; outputformat that will return, HTML document, plain text document, or a TSV file. The folder path must contain a folder named db, and in that folder, a one named original files, where the annotated files are located. After the program has annotated the field, one folder is created inside db folder with the results of the annotations.

Bibliography

[1] David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June (Paul) Hsu, and Kuansan Wang. ERD 2014: Entity recognition and disambiguation challenge. *SIGIR Forum*, 2014.

[2] Alan Eckhardt, Juraj Hreško, Jan Procházka, and Otakar Smrſ. Entity linking based on the co-occurrence graph and entity probability. In *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, pages 37–44, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3023-7. doi: 10.1145/2633211.2634349. URL http://doi.acm.org/10.1145/2633211.2634349.

[3] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0), June 2013. URL http://lemurproject.org/clueweb12/.

[4] Fréderic Godin, Tom De Nies, Christian Beecks, Laurens De Vocht, Wesley De Neve, Erik Mannens, Thomas Seidl, and Rik Van de Walle. The Normalized Freebase Distance. May 2014.

[5] Sherzod Hakimov, Salih Atilay Oto, and Erdogan Dogdu. Named Entity Recognition and Disambiguation Using Linked Data and Graph-based Centrality Scoring. *Proceedings of the 4th International Workshop on Semantic Web Information Management SWIM12*, pages 1–7, 2012. doi: 10.1145/2237867.2237871. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84863477078&partnerID=40&md5=3c880d6e0cac6fe2ea6e6ecde5a9fc94.

[6] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. A greedy algorithm for finding sets of entity linking interpretations in queries. In *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, pages 75–78, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3023-7. doi: 10.1145/2633211.2634356. URL http://doi.acm.org/10.1145/2633211.2634356.

[7] Marek Lipczak, Arash Koushkestani, and Evangelos Milios. Tulip: Lightweight entity recognition and disambiguation using wikipedia-based topic centroids. In *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, pages 31–36, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3023-7. doi: 10.1145/2633211.2634351. URL http://doi.acm.org/10.1145/2633211.2634351.

[8] Pablo Mendes, Joachim Daiber, Rohana Rajapakse, Felix Sasaki, and Christian Bizer. Evaluating the impact of phrase recognition on concept tagging. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

[9] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0621-8. doi: 10.1145/2063518.2063519. URL http://doi.acm.org/10.1145/2063518.2063519.

[10] Jack Menzel. Deeper understanding with metaweb, July 2010. URL http://googleblog.blogspot.no/2010/07/deeper-understanding-with-metaweb.html.

[11] David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 509–518, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-991-3. doi: 10.1145/1458082.1458150. URL http://doi.acm.org/10.1145/1458082.1458150.

[12] David Milne and Ian H. Witten. An open-source toolkit for mining wikipedia. *Artif. Intell.*, 194:222–239, January 2013. ISSN 0004-3702. doi: 10.1016/j.artint.2012.06.007. URL http://dx.doi.org/10.1016/j.artint.2012.06.007.

[13] Brage Michael E. Roalkvam. Approach for spotting entity-bearing words in news articles. Technical report, Department of Electrical and Computer Engineering, University of Stavanger, 2014.

[14] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85, 2014. URL http://cacm.acm.org/magazines/2014/10/178785-wikidata/fulltext.