Entity Linking by Focusing DBpedia Candidate Entities

Alex Olieman[1] Hosein Azarbonyad[1,2] Mostafa Dehghani[1,2] Jaap Kamps[1] Maarten Marx[1]

[1]University of Amsterdam, Amsterdam, The Netherlands
[2]University of Tehran, Tehran, Iran

{olieman|kamps|maartenmarx}@uva.nl {h.azarbonyad|mo.dehghani}@ut.ac.ir

ABSTRACT

Recently, Entity Linking and Retrieval turned out to be one of the most interesting tasks in Information Extraction due to its various applications. Entity Linking (EL) is the task of detecting mentioned entities in a text and linking them to the corresponding entries of a Knowledge Base. EL is traditionally composed of three major parts: i) *spotting*, ii) *candidate generation*, and iii) *candidate disambiguation*. The performance of an EL system is highly dependent on the accuracy of each individual part. In this paper, we focus on these three main building blocks of EL systems and try to improve on the results of one of the open source EL systems, namely DBpedia Spotlight. We propose to use text pre-processing and parameter tuning to "focus" a general-purpose EL system to perform better on different kinds of input text. Also, one of the main drawbacks of EL systems is identifying where a name does not refer to any known entity. To improve this so-called *NIL-detection*, we define different features using a set of texts and their known entities and design a classifier to automatically classify DBpedia Spotlight's output entities as "NIL" or "Not NIL". The proposed system has participated in the SIGIR ERD Challenge 2014 and the performance analysis of this system on the challenge's datasets shows that the proposed approaches successfully improve the accuracy of the baseline system.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process, Selection process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*performance evaluation (efficiency and effectiveness)*

General Terms

Experimentation, Measurement, Performance

Keywords

Entity Linking; DBpedia Spotlight; ERD challenge

1. INTRODUCTION

Entity linking (EL) is the information extraction task of automatically "matching a textual entity mention to a knowledge base entry, such as a Wikipedia page that is a canonical entry for that entity." [2, p. 96] Three key challenges have been identified for EL to deal with: name variation, entity ambiguity, and absence [2, 8]. Name variation entails that an entity can be referred to by multiple different terms. Entity ambiguity refers to the issue that a single name string can match with several distinct entities. The issue with absence is that when no knowledge base (KB) entry exists for the entity that is mentioned in the text, no entity (or NIL[1]) should be returned, rather than the highest-ranking KB entry.

Interest in entity linking is growing, as witnessed by the number of EL tracks that have been initiated at benchmarking conferences in recent years [2]. There are, however, two relevant limitations present in the existing work on EL. Most research focuses explicitly on linking named entities (i.e. entities referred to by proper names), specifically on persons, locations, and organizations [4, 7, 8]. Additionally, many current approaches are evaluated only on English-language texts, with a focus on the news domain [4, 7, 8]. Generally, EL is seen as a method to generate hyperlinks that can be followed by a reader to find more information about the mentioned entities.

EL could be useful in different applications such as Information Retrieval and Expertise Profiling. From the IR view, finding entities of queries and linking them to the entries in KB could help the system to concentrate the search on entities. Finding appropriate entities in queries could resolve the problem of ambiguity which most of search engines suffers from this problem. Moreover, search engines could use these entities to organize their information around entities. In fact, rather than returning related documents, a search engine could return direct information about the entities that the user is looking for and satisfy the users information need.

Recently, with the growth of focuses on EL, different system have been designed and developed for finding entities in a given text. These systems have been composed of three main parts: spotting, candidate generation, and disambiguation. Spotting which refers to detecting all non-overlapping strings in a text that could mention an entity, is the first part of EL systems. The candidate generation is

[1]NIL is a generated entity which represents the hypothesis that the textual mention does not refer to any entity included in the KB [3].

finding all possible candidate entities that may be referred to the spotted string for each spotted string. Finally, after generating the candidates, in disambiguation step, for each spotted string the candidate that is most likely referred to the spotted string is selected as its corresponding entity in KB. Also, it is possible that none of the candidates are correct annotation for the spotted string or the candidate entities are not included in the KB. In this case, the disambiguation module should return NIL for the spotted string. One of the Open Source EL systems is DBpedia Spotlight which is a system for detecting DBpedia entities in a given text. In fact, the target KB of this system is DBpedia which is a rich KB for the EL task. Each entry of DBpedia contains information of an entity such as a description of entity, its corresponding Freebase and Wikipedia URLs, the categories the entity belongs to them, and so on.

This paper focuses on DBpedia Spotlight as a baseline and tries to resolve some of its problems and improve its accuracy. One the main reasons of using DBpedia Spotlight is that it is a very configurable system which makes it a good choice for using it as baseline EL system. In DBpedia Spotlight, spotting can be done by employing NLP techniques such as Named Entity Recognition, Detecting Multi Word Entities, and finding sequences of capitalized words which are main approaches used in most of EL systems. Alternatively, a language-independent method, based on a surface form dictionary, can be used for spotting texts. This step is one of the important parts of EL systems and most of EL systems try to produce all possible surface forms for a given text. However, all detected spots do not necessarily refer to an entity in the target KB. Therefore, one of the important post-processing approaches to improve the accuracy of DBpedia Spotlight could be determining whether a spotted text is really refers to an entity in KB or not. This decision is called "NIL detection". After spotting the given text, all possible candidate entities for the spotted texts should be generated. Since the formatting of the spotted string may be different from the formatting of saved entities in the target KB, some pre-processings should be done in order to match the spotted strings with entities in KB. Therefore, another important step in improving the performance of an EL system could be the pre-processing of the input text and matching its entities with the entities of the KB.

Another dominant factor of the performance of the EL system is to determine, given the context, which of the candidates of a surface form is most likely to be mentioned in this instance. Also, in some cases it is possible that for a surface form there are more than one correct entity. In this case, the EL system should rank the candidates according to their correctness rate. Therefore, after candidate generation, a post-processing on the candidates and disambiguating them could improve the accuracy of the EL system.

The SIGIR ERD 2014 challenge focuses on the recognition and disambiguation of mentioned entities in texts. It consists of two main tracks: the short track in which the goal is finding entities within a short text, such as web search queries, and the long track in which the goal is detecting entities within a document such as a news article. The authors of this paper participated in the ERD 2014 challenge and the proposed system of this paper has been submitted, with good results. In this paper, we focus on three major techniques which could improve the performance of DBpedia Spotlight: pre-processing, NIL detection, and candidate disambiguation. As pre-processing we first tune parameters of DBpedia Spotlight and find its best configuration. Also, we normalize the character encoding and transform all documents formatted in different formats to a unique format. Additionally, since DBPedia Spotlight is case sensitive, in order to find all possible surface forms of texts, in addition to the main text, we capitalize the input text and submit them to the DBpedia Spotlight.

For the NIL detection part of our system, we use two different approaches: filtering candidates which are not included in the target KB and classifying entities as "NIL" or "Not NIL" instances. Since the target KB of DBpedia Spotlight is different from our target KB (the ERD 2014 target KB), it is possible that the detected entities by DBpedia Spotlight do not exist in the target KB. Therefore, we filter out the surface forms that their all candidates do not exist in the KB and consider them as NIL. In the classification approach, we use some texts which their entities are annotated to learn a classifier that classifies candidates as "NIL" or "Not NIL". We extract different types of features from these annotated texts and their entities and train a classifier. Then, we use these classifier such that if the score that the classifier assigns to a candidate is lower than a predefined threshold, we classify it as "NIL".

Finally, in order to disambiguate the candidates, we use the generated scores by the classifier to find the most probable candidate for each surface form. The defined features exploit the context within the text to estimate the correctness rate of an entity. The main idea behind these features is that the entities that are mentioned in a given text are related to each other. Therefore, we could use other mentioned entities in a given text to disambiguate an entity. We use this intuition and define several features. For example, we traverse the categorical structure of Wikipedia to estimate a relatedness score for each entity based on the closeness of its categories to the categories of other entities of the text.

The rest of this paper is organized as follows. In Section 2, we describe the DBpedia Spotlight which is used as the baseline of this paper. Section 3, describes the main methods used in this paper for "NIL detection" and entity disambiguation. In Section 4, the results of different experiments with the discussions on the results are presented. Finally, Section 5 concludes the paper with a brief description of future work.

2. BASELINE: DBPEDIA SPOTLIGHT

DBpedia Spotlight is an open-source system that can annotate any given input text with DBpedia resources (i.e. KB entries), which are based on semantic extraction from Wikipedia articles [5]. Several parameters provide the means to filter annotations according to task-specific requirements [5]. By default, DBpedia Spotlight is not specialized towards specific entity types, but it may be configured to annotate only instances of specific types, either by selection of classes, or by arbitrary SPARQL[2] queries [5].

There are two different branches of DBpedia Spotlight implementation. The original Information Retrieval-based implementation, is characterized by the use of a TF.ICF (Term Frequency . Inverse Candidate Frequency) measure for disambiguation. It ranks disambiguation candidates by

[2]SPARQL 1.1 - `http://www.w3.org/TR/sparql11-overview/`

querying a Vector Space Model (VSM), in which entities are represented by the paragraphs that mention them in Wikipedia, with the context of the observed phrase [5].

As baseline in our experiments, we instead use the newer "Statistical implementation" (version 0.7), which uses a generative probabilistic model for disambiguation. This implementation shows improvements in recognizing phrases that refer to known entities, and in disambiguation performance [3]. It also has several practical benefits, such a significantly higher speed and a smaller memory footprint.

2.1 Statistical Models

The statistical models that DBpedia Spotlight uses in the Entity Linking process are based on a dump of Wikipedia articles. The article dump is processed with PigNLProc to count several kinds of (co-)occurrences within the articles [3]. Most of the raw counts are focused on the hyperlinks between articles on Wikipedia (i.e. *internal wikilinks*).

While each internal wikilink is processed, a count is kept of the Wikipedia entries that are the link targets. This count is used as the prior probability P(e) in disambiguation [3], where

$$P(e) = \frac{|wikilink(e)|}{|wikilink|} \tag{1}$$

Secondly, the co-occurrence of entries and anchor texts is counted. This allows for the likelihood that a given anchor text s is observed, given the entity e, to be estimated as

$$P(s|e) = \frac{|wikilink(e,s)|}{|wikilink(e)|} \tag{2}$$

Thirdly, the context c in which a given entity is mentioned, is represented in terms of the tokens that occur in the vicinity of wikilinks where the article about this entity is the link target. For each paragraph that contains a $wikilink(e)$, the tokens are stemmed and added to a bag-of-words that is specific for e. The tokenizer, stemmer, and stopwords that are used in this operation are language-dependent. A unigram language model is created for each entity, based on these bags-of-words. Hence, the likelihood of e generating the context c is calculated as

$$P(c|e) = \prod_{t \in c} P_e(t), \tag{3}$$

where $P_e(t)$ is estimated as a weighted average of the likelihood of t given the entity's language model and the likelihood of t given P_{LM}: a smoothed unigram language model over all tokens that have been found in the context of any entity mention. See [3] for the details of this definition.

A final count over all Wikipedia articles is needed for phrase spotting. For each distinct string that is used as anchor text the total number of occurrences in the article dump is counted. To avoid a string search in the entire corpus for each known phrase, the counts are instead taken from all n-grams ($n = 5$) in the corpus [3].

The raw counts are finally serialized into data structures that can be efficiently loaded into memory during initialization. In this step, additional processing is needed to deal with Wikipedia *redirects* and *disambiguation pages*, which are included in the article dump as regular entries. Additional DBpedia datasets are used to resolve redirects to their target articles and to find the disambiguation candidates for disambiguation pages.

2.2 Phrase Spotting

Phrase spotting is the task of finding phrases in an input text that should be linked to an entity. Achieving high recall is most important for this task, because any phrases that are overlooked in this step will lead to missing annotations in the final system output [6]. Precision is desirable with regard to computational efficiency, and because false positives in this step will propagate unless a NIL-decision is made during disambiguation.

The experiments described in this paper make use of a lexicon-based phrase spotting approach. The set of anchor texts that has been obtained during model creation serves as the lexicon, and it is used to construct a finite state automaton that encodes all known sequences of tokens. During runtime the Aho-Corasick algorithm is used to simultaneously match all token sequences in the lexicon with the input text. The algorithm is configured for case-insensitive string matching, and it resolves overlapping matches by selecting the longest match [6].

It is also possible to use a language-dependent phrase spotting approach with DBpedia Spotlight. This approach is largely based on existing phrase chunking and named entity recognition models from the Apache OpenNLP library [6]. We do not use this approach in our experiments, because it does not perform significantly better than the lexicon-based approach [3, 6].

After an input text has been searched for entity names in the lexicon, a further selection is made from all candidate phrases. An important feature in making this selection is the prior probability that phrase s in the corpus is used as the anchor text of an internal $wikilink(e,s)$ [3]. Here, case-sensitive matching is used. This feature discriminates between candidate phrases on the basis of whether it is probable that the phrase should be annotated at all. The string "Here", to give an example that is unlikely to be annotated, is found as anchor text 84 times, out of a total of 88,022 occurrences.

There are several types of phrases (e.g. acronyms) that are not used as wikilinks as often as we would like them to be annotated in arbitrary input text. The prior annotation probability is combined in a linear model with several binary features that are meant to correct for these problematic phrase types [3]. A bottom threshold α is applied to the final score to select the phrases for which disambiguation candidates will be found.

Disambiguation candidates for a phrase are found by a case-sensitive search in an anchor text (i.e. *surface form*) - entity mapping. Only in the case that no candidates are found for an exact match, are disambiguation candidates retrieved by a case-insensitive match. Character-case information is thus taken into account whenever possible.

2.3 Disambiguation

The statistical models are used to calculate a disambiguation score for entity e, given the surface form s and its context c, by combining $P(e)$, $P(s|e)$, and $P(c|e)$ [3]. By default the score is calculated as the product of the individual values:

$$P(e|s,c) = P(e) * P(s|e) * P(c|e). \tag{4}$$

This disambiguation probability will be very small for all candidates, due to the negligible value of the priors. Softmax normalization is therefore used to provide a more usable disambiguation score [3], where the sum of the scores of all disambiguation candidates is 1.0 for any given annotation.

As a basic form of NIL-detection, a disambiguation score is also produced for the hypothesis that s does not refer to any known entity [3].

$$P(NIL|s,c) = \frac{1}{|wikilink|} \prod_{t \in s} P_{LM}(t) * \prod_{t \in c} P_{LM}(t) \quad (5)$$

The disambiguation candidates with a score lower than the NIL-score are removed, after which the candidate with the maximum score is selected as annotation. If NIL is the only remaining entity, no annotation is placed at all.

DBpedia Spotlight's API exposes two parameters that can be used to further restrict which disambiguation candidates are seen as valid. The *confidence* parameter, which is provided at runtime, applies as an additional bottom threshold to candidates' disambiguation scores. A second runtime parameter, *support*, specifies the minimum number of Wikipedia inlinks that a candidate resource must have to be further considered [5].

3. METHODOLOGY

The need for modification of a general-purpose EL system, such as DBpedia Spotlight, arises from the difference in the set of target entities, and differing characteristics of the input text. In the ERD 2014 challenge the set of target entities is based on a Freebase dump, from which only entities with a corresponding Wikipedia article are kept. This selection is further filtered by entity types to include only named entities, such as persons, locations, organizations, products, and creative works [1]. In this paper we denote this set of target entities as the vocabulary V.

The input texts that were used to evaluate the systems' performance in the ERD 2014 challenge differ for the two tracks in which we participated. For the short track the texts are (lowercased) web search queries that have been used in earlier TREC workshops. For the long track, approximately half of the input texts are arbitrary web pages originating from a web search engine's index, and the other half consists of press releases that have been collected from various news sources on the web [1].

3.1 Parameter Tuning

The DBpedia Spotlight implementation that we use allows for phrase spotting to be influenced by the parameter α, and for disambiguation to be influenced by the *confidence* and *support* parameters. These three parameters are all bottom thresholds and, as such, low values are expected to result in higher recall and high values are expected to result in higher precision. Due to an implementation issue, α could unfortunately not be assigned a value separately, but instead uses the same value as *confidence*. In our experiments *confidence* thus controls both phrase spotting and disambiguation.

We define a third runtime parameter which chooses between *single* and *multiple candidate filtering*. In single candidate filtering, we take the entity $\hat{e}_s$ that Spotlight has selected as annotation for a surface form and we select it as our disambiguated entity $\hat{e}$ only if it occurs in the vocabulary ($\hat{e} = \{\hat{e}_s\} \cap V$). If $\hat{e}_s$ is not present in V, we consider this annotation to be NIL. This approach to filtering annotations with a custom vocabulary is taken from Mendes et al., who used it for the TAC KBP 2011 Entity Linking task to improve their NIL-detection [4].

In multiple candidate filtering we instead initialize take the candidate vector $\overline{r}_s$, that has been sorted from high to low disambiguation scores, and select as $\hat{e}$ the entity with the highest score that is also present in V. If none of the e in $\overline{r}_s$ are also in V, we consider this annotation to be NIL.

$$\hat{e} = e : e_{min(j)} \in \{e_j : e_j \; in \; \overline{r}_s \; and \; e \in V\} \quad (6)$$

This candidate filtering parameter does not apply to the short track, because the task there is to produce a set of valid interpretations $\hat{E}$ for each spotted phrase. For the short track, we always use the intersection between the candidates generated by Spotlight $\hat{E}_s$ and the vocabulary.

$$\hat{E} = \hat{E}_s \cap V \quad (7)$$

The goal of varying the values of these parameters in our experiments is to examine their influence on the resulting precision, recall, and F-score. While we expect all runtime parameters to influence a precision-recall trade-off, we attempt to find maxima in the F-score in particular.

3.2 Input Text Pre-Processing

The types of input text that are used in the ERD 2014 challenge are in several respects quite different from the Wikipedia articles that have been used as the basis for the DBpedia Spotlight models. None of the types of input text share the encyclopedic style of writing that is adhered to by Wikipedia editors.

The web search queries are the most different, because of their compactness, lack of case information, and informality. The press releases that are used in the long track seem most comparable to Wikipedia articles, because they feature formal language use, and are likely to have a central topic.

3.2.1 Character Encoding Normalization

The documents that are used as input text in the long track are gathered from various sources on the web, and as such they featured several character encodings (e.g. UTF-8, ISO-8859-1). The information of which character set was used to represent the document as a byte-string should have been specified by the content publishers, but in practice this information is not always accurate. This is a problem for our phrase spotting approach, because it relies on exact string matching, and thus will not recognized mentioned entities if any of the characters in their surface form are interpreted incorrectly.

To deal with this issue, our system internally works only with Unicode characters. Upon receiving an input text (as byte-string without character encoding declaration), a simple heuristic is applied to decide how the string will be decoded. First it is assumed that the text is encoded in UTF-8, and it is decoded as such. This decoding occasionally fails, because UTF-8 is a multi-byte character encoding for which many byte sequences are not valid characters. If this is the case, we decode the text as ISO-8859-1 instead, unless the resulting Unicode string contains control characters. The final guess, then, is Windows-1252.

The heuristic can be this simple because all input text is written in English, and nearly all English web pages are encoded in either UTF-8, Windows-1252, or ISO-8859-1 [9].

3.2.2 Capitalizing Each Word

By default, phrase spotting in DBpedia Spotlight relies significantly on correct case information. The main purpose of case-sensitively mapping surface forms to candidate entities is to distinguish between proper names and other kinds of phrases.

Consider, for example, the following set of surface forms from which the case information has been removed:

{*'michelle', 'the tv set', the eagles', 'the turtles'*}

The candidate entities that would be found without taking case information into consideration, would have to include The Beatles' song `Michelle`, the 2006 movie `The_TV_Set`, and the well-known bands `The_Eagles` and `The_Turtles`. Now suppose that these surface forms were found in the sentence:"*Michelle, look at the TV set! The turtles just disappeared into their shells once the eagles arrived. I love this show!*" If case information was taken into account, the surface forms "TV set", "turtles", and "eagles" could be found instead, and they would not need to be mapped to the aforementioned incorrect candidate entities.

The web queries that need to be annotated in the short track, however, completely lack case information. By capitalizing each word in the queries, before they are sent to DBpedia Spotlight, we can test the hypothesis that the majority of entities that are mentioned in the queries are named entities. We expect to see more correct annotations as a result of capitalizing queries, and not too many new annotations that should be NIL.

The long track documents do include case information, although it might be less consistent than in Wikipedia articles. Here, we also experiment with capitalizing each word in the input documents. By doing this, we effectively remove case information, but in favor of generating named entities as disambiguation candidates. We expect the most prominent effect of this processing to be an increase in annotations that should have been NIL, because the entities that their surface forms refer to are actually non-named entities that are not in the vocabulary.

3.3 Spotting and Candidate Generation

During initial tests with the baseline system we came across two issues in the phrase spotting and candidate generation step.

The first issue is caused by the "longest match" criterion for resolving overlapping surface forms, in combination with the vocabulary that has a smaller coverage than DBpedia Spotlight. This problem occurs when a recognized surface form yields candidate entities that are not in the vocabulary, but it has a substring that is a surface form for entities that are in the vocabulary. Consider, for example, the surface form *"California Franchise Tax Board"*. The entity that is referred to is correctly disambiguated by DBpedia Spotlight, but our system decides $\hat{e}$ is NIL because $\hat{e}_s$ is not in V. *"California"*, however, is in V and should be recognized instead.

The second issue is caused by the case-sensitive candidate lookup, in relation with the smaller vocabulary. As described in Section 2.2, a surface form is first matched case-sensitively to retrieve candidate entities, and a case-insensitive lookup is only done if there are no candidates on the first try. This causes valid candidate entities to be overlooked if the case-sensitive match retrieves candidates that are not in V. The lowercase query *"total recall movie"* is an example where this occurs. DBpedia Spotlight finds the resources `Recall_(memory)` and `Eidetic_memory` for the surface form "total recall", but neither entity is in V. A case-insensitive match would have yielded two valid movie entities (that were exclusively mapped to "Total Recall"), but this doesn't occur because some candidates have already been found.

3.3.1 Recreate the Statistical Models from Modified Raw Data

Both described issues can, in principle, be addressed by removing all references to entities that are not in V from the statistical models. To this end we process the files with raw counts (described in Section 2.1) row-for-row, and recreate the statistical models from the modified files. The drawback of using models with only target entities is that they will make less accurate NIL-decisions. We are, however, interested in whether this will be somewhat compensated by a higher recall.

The raw counts of wikilink targets, used for $P(e)$, and the entity-context index, used for $P(c|e)$, are simply filtered to keep only the rows that refer to entities in V. The surface form-entity co-occurrence counts, used for $P(s|e)$ and the surface form-candidates mapping, are filtered similarly, but here we keep a log of the counts that are removed in memory. The counts of surface form occurrence as wikilink and their total occurrence in the Wikipedia corpus, are subsequently updated with the counts that have been logged in the previous step. If a surface form only occurred with entities that have been removed, it too is removed entirely from the model.

3.3.2 Merging Disambiguation Candidates

The main reason to want to combine the disambiguation candidates from systems that are configured differently, is that a system may find a good set of candidates generally, but overlooks candidates in some corner cases. We experiment with combining the candidate entities that are found by giving a system the original and a capitalized version of the input text. We also combine candidates from the original Spotlight models with the candidates that are generated by our modified models.

An additional reason is specific to the Spotlight implementation that we use, and has to do with the spotting score threshold and disambiguation score threshold both being controlled by the *confidence* parameter. By letting the output of a primary system determine which phrases are spotted, an additional system output can add candidate entities for these surface forms with a lower confidence. We have implemented the merging of disambiguation candidates as an asymmetric operation. The surface forms that are produced by the primary system are taken as starting point. The merging algorithm iterates through these surface forms, and, for the short track, takes the union of the candidates from both system outputs, that are also in V, as output entities. For the long track, the algorithm instead iterates through the sorted candidates of the primary system and selects the first candidate that is in V as output entity. If no entity is selected from these candidates, the first entity that is in V is selected from the sorted list of the additional output's candidates.

3.4 Disambiguation

One of the main drawbacks of EL system is existing ambiguity in their detected entities. Therefore, disambiguating candidates could be an effective approach to increase the accuracy of an EL system. In this section, we propose a method for assigning scores to the candidates of entities and disambiguating entities based on these scores. This method, namely "NIL detection classifier" is also used for detecting whether a spotted text really refers to an entity or not. In this view, this method could be considered as a "NIL detector".

3.4.1 *Supervised Disambiguation*

In this section, we describe a supervised approach for the disambiguation part of the ERD task. In this approach, we consider that the spotter finds surface forms of entities in the given text and if the surface form is ambiguous, there are several candidates that should be disambiguated. To do so, we define several features that show the correctness of assigning the entity to the candidate. Then, we use a machine learning approach to learn a complementary model to be used in disambiguation.

3.4.2 *NIL Detection Classifier*

Before the disambiguation it is possible that the spotter determines some surface forms from the given text and generates candidate entities that are not valid in-context. It stands to reason that a filtering approach could be beneficial to remove the incorrect candidates of that surface form. Detecting these types of candidates is called "NIL Detection". Spotters are usually generates a confidence level for each candidate and in a naive approach, the candidates could be removed if their confidence is lower than a threshold. NIL detection could be smarter and in the case of existence of train data, supervised approaches can be used for this purpose.

In our proposed approach for NIL detection, the problem of NIL detection is mapped to a binary classification problem in which the goal is to determine whether a generated candidate is a correct candidate or it should be removed.

To train this classifier, a train data is used containing a set of texts with their annotated entities. We use our spotter to determine the text's surface forms and their corresponding candidates. Then, we label the candidates such that the candidates existing in the annotated train data are labeled as "true" and the others as "false". Different types of features are defined and are used to learn a classification model. This learned model then is used for NIL detection. The defined features will be described in Section 3.5.

3.5 Extracted Features

In this section, we explain the defined features that are extracted to be used in the supervised disambiguation approaches.

3.5.1 *Probability of Entity Surface Form Occurrence in Candidate's Wikipedia Page*

One of the important features for selecting the correct annotation of an entity could be the likelihood of occurrence of entity surface form in the Wikipedia page of the candidate. In fact, if a candidate is a correct annotation for an entity, the surface form of the entity should be occurred frequently in the candidate's Wikipedia page compared to Wikipedia pages of wrong candidates. This feature is estimated as follows:

$$P(e|WP_c) = \frac{count(SF_e, WP_c)}{|WP_c|} \tag{8}$$

where e is the entity, WP_c indicates the Wikipedia page of the candidate, $count(SF_e, WP_c)$ is the frequency of surface form of e in WP_c, and $|WP_c|$ is the length of WP_c in terms of word.

3.5.2 *Probability of Occurrence of Document's Entities in Candidate's Wikipedia Page*

The idea behind this feature is the fact that usually the entities of a document are related to each other. Therefore, we could use other entities of the document as its contextual information for disambiguating an entity and finding its correct annotation. For calculating this feature, we first use the DBpedia Spotlight to find the possible entities of the document. Then, for each candidate annotation of each entity we estimate this feature as follows:

$$P_E(D_e|WP_c) = \frac{\sum_{e' \in E_D - \{e\}} P(e'|WP_c)}{|E_{D_e}|} \tag{9}$$

where D_e is the document, WP_c is the candidate's Wikipedia page, E_D is the entities of the document annotated by DBPedia Spotlight, $|E_{D_e}|$ is the number of entities of the document and $P(e'|WP_c)$ is calculated using Equation 8.

3.5.3 *Cosine Similarity of Document and Candidate's Wikipedia Page*

Intuitively, the context in which the entity is mentioned should be similar to the description of the entity's correct candidates. The description of the candidates could be considered as their Wikipedia pages. Therefore, one of useful features for determining whether a candidate is correct annotation for an entity or not, could be the similarity of the document in which the entity is mentioned and the Wikipedia page of the candidate. We use Cosine similarity for estimating the similarity of the document and Wikipedia page of candidate as follows:

$$Cosine(D_e, WP_c) = \frac{\sum_t w_t(D_e) * w_t(WP_c)}{\sqrt{\sum_i^{|D_e|} w_i(D_e)^2} * \sqrt{\sum_i^{|WP_c|} w_i(WP_c)^2}} \tag{10}$$

where D_e is the document and WP_c is the Wikipedia page of the candidate. $w_t(D_e)$ and $w_t(WP_c)$ are the weight of word t in D_e and WP_c consequently. We consider the frequency of t in document (or Wikipedia page) normalized by the length of document (or Wikipedia page) as the weight of t.

3.5.4 *Reference Similarity of Candidate's Wikipedia Page and Document*

References of a document are a proper sign of its topics. Therefore, one of the useful features for detecting the correct annotation of an entity could be the number of common references between the correct annotation and the document in which the entity is mentioned. We consider the outlinks of Wikipedia page of a candidate as its references and the references of a document as the union of the references of entities mentioned in the document. We first use DBpedia Spotlight to find possible entities in the document and then, we consider the union of outlinks of the Wikipedia pages of entity's candidates as the references of each entity. For estimating the value of this feature, we calculate the Dice similarity of the set of document's references and set of candidate's references as follows:

$$SimCommonRef(D_e, c) = \frac{|Ref_{D_e} \cap Ref_c|}{|Ref_{D_e} \cup Ref_c|} \quad (11)$$

where D_e is the document, c is the candidate annotation, Ref_{D_e} is the set of document's references and Ref_c is the set of candidate's references.

3.5.5 Categorical Distance of Candidate and Document

An interesting feature for disambiguating an entity could be the distance of Wikipedia pages of candidates with the Wikipedia pages of other entities of the document in the Wikipedia graph. The main intuition behind this feature is that the entities of the document usually are related to each other and their corresponding Wikipedia pages are close to each other in Wikipedia graph. In disambiguating an entity, we could use this intuition and consider a candidate that has shortest distance with other entities of the document as the most probable correct annotation for the entity. However, the Wikipedia graph is very large and finding shortest distance between pages of this graph is computationally expensive. Therefore, instead of using Wikipedia graph, we use Wikipedia's category tree for estimating the distance of entities. We first define the category set of a document as the union of the categories of entities mentioned in the document. Also, the category set of an entity is defined as the union of categories of Wikipedia pages of entity's candidates. For each candidate of an entity we define the its distance from other entities of the document as the distance between the categories of the candidate with the categories of the document in the Wikipedia category tree. For calculating the distance of a candidate with other entities, we exclude the categories of the candidate and consider the document's categories as the categories of all entities except the categories of the candidate. We use Gremlin [3] to traverse the Wikipedia category tree. We consider the categorical similarity of a candidate with other candidates of the document as the inverse of their distance and calculate the similarity as follows:

$$Sim_{cat}(D_e, c) = \frac{|Cat_{D_e}| * |Cat_c|}{\sum_{e' \in E_{D_e}} \sum_{cand} dist(cand, c)} \quad (12)$$

where D_e is the document, c is a candidate for entity e, E_{D_e} is the set of entities in document, and *cand* is the set of all candidates of $e' \in E_{D_e}$. $dist(cand, c)$ is the distance of categories of *cand* and c which is defined as the sum of distances of each pair of categories of *cand* and c in the Wikipedia category tree. $|Cat_{D_e}|$ is the number of categories of D_e and $|Cat_c|$ is the number of categories of Wikipedia page of c.

3.5.6 Features extracted using DBpedia Spotlight

The scores that DBpedia Spotlight gives to the entity's candidates could be useful for disambiguating an entity. In this paper, we use some of scores generated by DBpedia Spotlight for constructing the candidate ranking model. One of these features is the final score that the DBpedia Spotlight gives to candidates. This score is product of $P(e)$, $P(s|e)$, and $P(c|e)$ which are described in Section 2.1. Another score generated by DBPedia Spotlight and used as a feature in this paper is DBPedia Spotlight's Contextual score for each candidate. This score is $P(c|e)$ which is described in Section 2.1. The percentage of second rank for a candidate which is the final score of the next best candidate for an entity compared to the final score of the current candidate is another feature extracted using DBpedia Spotlight and used for constructing the ranking model.

[3]Gremlin - `https://github.com/tinkerpop/gremlin/wiki`

4. RESULTS AND ANALYSIS

This section describes the results of our experiments and their subsequent analysis. The presented results are based on our submissions to the ERD 2014 challenge, but we have supplemented this with results that we have obtained outside of the boundaries of the challenge. The challenge organizers have provided an evaluation tool, which compared a submitted system's output to an initial golden standard. This online evaluator was, however, limited to providing an $F_1 - score$ for the short track, and the additional performance measures of precision and recall for the long track.

To enable a more detailed evaluation of our results, we created an error analysis script that followed the same evaluation approach as the online evaluation, but which gave access to document-level statistics. This error analysis could only be done for the long track, because no golden standard was released for the short track. For the long track we have evaluated against a golden standard that was released near the end of the challenge, in which we corrected a small number of mistakes.[4]

The final performance rankings of the ERD 2014 challenge are based on a more extensive golden standard, that has been judged by multiple human annotators [1]. This final golden standard, nor online access to it, have been made available at the time of writing, so the results in this paper could unfortunately not benefit from its superior quality.

As a baseline in our experiments we use DBpedia Spotlight with the default configuration of *confidence = 0.50* and *support = 0*. For the online evaluation we have, however, used an elevated baseline with *confidence = 0.30* in several experiments because of the time cost of additional runs. In each table with results we report on the difference in F-score (ΔF_1) compared to the baseline configuration in that experiment.

4.1 Parameter Tuning

All parameters exhibit a certain influence on the precision—recall trade-off, as was to be expected. The performance statistics from the online evaluation (see Table 1) show that a range of precision and recall values can be obtained, without having a major influence on the resulting F-score. The filtering effect of the confidence parameter on spotted phrases and candidate entities offers the most predictable effect on the precision-recall trade-off.

Support had a more drastic effect on the results in the short track. Because web queries do not contain much context, the ERD challenge guidelines specify that a set of valid interpretations should be annotated for each spotted phrase. If this interpretation set does not contain exactly the same entities as its counterpart in the golden standard, the en-

[4]The modified golden standard: `http://goo.gl/6oWsia`

Table 1: Online Evaluation of Parameter Tuning

		Short Track			
Conf.	Supp.	F_1	ΔF_1		
0.30	30	0.245	-53.24%		
0.20	30	0.518	-1.15%		
0.50	0	0.524	—		
0.40	0	0.533	+1.72%		
0.08	30	0.542	+3.44%		
0.30	0	0.542	+3.44%		
		Long Track			
Conf.	Cand.	F_1	ΔF_1	Precision	Recall
0.15	multi	0.681	-0.44%	0.638	0.729
0.15	single	0.682	-0.29%	0.644	0.726
0.50	single	0.684	—	0.814	0.591
0.35	single	0.694	+1.46%	0.733	0.659
0.30	multi	0.695	+1.61%	0.706	0.685
0.40	single	0.695	+1.61%	0.759	0.641
0.30	single	0.697	+1.91%	0.709	0.686

Table 2: Error Analysis of Parameter Tuning

			Long Track			
Conf.	Supp	Cand.	F_1	ΔF_1	Pr.	Re.
0.10	0	multi	0.484	-27.76%	0.362	0.848
0.10	0	single	0.495	-26.12%	0.377	0.837
0.70	0	both	0.586	-12.54%	0.819	0.453
0.50	50	both	0.620	-7.46%	0.803	0.512
0.50	10	both	0.664	-0.90%	0.808	0.589
0.50	0	both	0.670	—	0.808	0.598
0.30	0	multi	0.714	+6.57%	0.741	0.737
0.30	0	single	0.715	+6.72%	0.743	0.733

tire annotation is marked as incorrect. Because candidate entities that have less Wikipedia inlinks than the support value are filtered out, any annotations that have entities with low prominence as valid interpretations are affected even by modest support values. The performance in the long track (see Table 2) was only subtly influenced by the support parameter, but still had more of a detrimental effect on recall, than a positive effect on precision.

The choice between single and multiple candidate filtering has the most subtle influence on precision and recall. Single candidate filtering assists in NIL-detection, as intended, while multiple candidate filtering allows for a small number of additional valid entities to be found. The influence on the F-score, however, is in all cases more favorable for single candidate filtering.

In our error analysis on the local golden standard we observed several combinations of confidence and support values for which both candidate filtering methods resulted in exactly the same annotations. The corresponding rows in Table 2 are marked with *"both"* in the candidate filtering column. The lists of candidate entities, in these cases, either contained the mentioned entity at the top (i.e. with the highest disambiguation score), or below the highest-ranking entity that is in V, or not at all. The small differences in performance that are caused by the candidate filtering method for other confidence and support values, suggest that this observation is applicable to the majority of sorted candidate entity lists.

DBpedia Spotlight's default values for *confidence = 0.50* and *support = 0* seem very reasonable for the input texts in both tracks. The confidence value of 0.5 does seem to choose precision at the cost of recall, and this is not ideal when precision and recall are deemed equally important, as in the ERD challenge. We have found that a confidence value of 0.3 balances precision and recall much more equally, which results in a higher $F_1 - score$ on the sets of input texts that are used in the challenge.

4.2 Input Text Pre-Processing

The difference in performance between the short and long track in the parameter tuning experiment supports our expectation that the web queries offer more of a challenge to DBpedia Spotlight than the long track documents. The results that are described in this section provide evidence that a large part of this difference is caused by missing character-case information in the short track queries.

There is, however, also a relevant difference in performance between the two kinds of documents that are used in the long track. In our error analyses we calculated the document-level performance, and we have used these statistics to calculate separate averages for the arbitrary web pages and the press releases. For the baseline configuration (*conf. = 0.5*) we observed a nearly identical recall of 0.60 for both kinds of documents, but with a precision of 0.73 for the arbitrary pages, and a precision of 0.89 for the press releases. In the best-performing run (*conf. = 0.3*) we found a precision of 0.67 and recall of 0.76 for the arbitrary pages, and a precision of 0.82 and a recall of 0.70 for the press releases. This is consistent with our expectation that the press releases are a better fit for models based on Wikipedia, although this does not explain the higher recall for arbitrary pages with confidence set to 0.3.

4.2.1 Character Encoding Normalization

The short track queries only contained ASCII characters, so charset normalization was not applicable here. The long track documents did contain several entity mentions where the surface forms included a non-ASCII character. By using the heuristic approach for decoding input text into Unicode characters, we observed an increase in F_1 of 1.3% compared to a baseline where input text was naively decoded (i.e. with errors). However, because character encoding normalization is such a low-level issue, we have included it in all other experiments; also in the baselines.

4.2.2 Capitalizing Each Word

By capitalizing each word in the web queries, we artificially inflated the values of P(s|e) for named entities. If the majority of entities that are mentioned in the queries are named entities, this should lead to a performance increase. The results of this experiment are summarized in Table 3, wherein the first column indicates whether the input text was capitalized.

For most of the short track configurations the capitalization of each word in the queries leads to a relatively high performance increase. From the results that we could obtain from the online evaluation, the increase seems to be approximately 8%, regardless of confidence value. Using a support value greater than zero again leads to a surprising

Table 3: Online Evaluation of Capitalization

		Short Track		
Capw	Conf.	Supp.	F_1	ΔF_1
yes	0.08	30	0.410	-34.35%
no	0.40	0	0.533	-1.66%
no	0.08	30	0.542	0.00%
no	0.30	0	0.542	—
yes	0.40	0	0.577	+6.46%
yes	0.20	0	0.583	+7.56%
yes	0.30	0	0.587	+8.30%
yes	0.25	0	0.598	+10.33%

		Long Track				
Capw.	Conf.	Cand.	F_1	ΔF_1	Precision	Recall
yes	0.30	single	0.364	-47.78%	0.357	0.371
no	0.30	single	0.697	—	0.709	0.686

Table 4: Online Evaluation of the Modified Model

		Short Track		
Model	Conf.	Supp	F_1	ΔF_1
mod.	0.08	30	0.213	-60.70%
mod.	0.20	30	0.350	-35.42%
mod.	0.30	0	0.379	-30.07%
orig.	0.20	30	0.518	-4.43%
orig.	0.08	30	0.542	-0.00%
orig.	0.30	0	0.542	—

result, however. We have applied the same processing to the long track documents. Here, our expectation was a decrease in performance, because we were destroying the existing case information rather than artificially creating it. The results indicate that capitalization indeed caused an increase in annotations that should have been NIL. The accompanying drop in recall, however, was unexpected and cannot be explained solely by the presence of the occasional non-named entity in V.

4.3 Spotting and Candidate Generation

In this section we report on the results of the modified models, and of combining the candidate entities from differently configured systems.

4.3.1 Recreate Statistical Models from Modified Raw Data

We have re-created DBpedia Spotlight's statistical models by filtering the raw counts to include only entities that occur in the vocabulary. The intent of this modification was to address the issues with phrase spotting and candidate generation that are described in Section 3.3. While the modified models indeed seemed to have solved these issues, their resulting performance shows that they also introduced many new errors (see Tables 4 and 5).

It was to be expected that the modified model would be less accurate at NIL-detection than the original model, but this performance difference should not have been much larger than the difference between single and multiple candidate filtering. To get insight into the nature of the newly created problems, we let our error analysis tool make a distinction between two kinds of False Positives (FPs).

Table 5: Error Analysis of the Modified Model

			Long Track			
Model	Conf.	Cand.	F_1	ΔF_1	Pr.	Re.
mod.	0.50	single	0.544	-18.81%	0.711	0.420
mod.	0.30	single	0.559	-16.57%	0.544	0.533
orig.	0.50	single	0.670	—	0.808	0.598
orig.	0.30	single	0.715	+6.72%	0.743	0.733

Table 6: Online Evaluation of the Candidate Merging

		Short Track		
Capw.	Conf.	Supp	F_1	ΔF_1
yes	0.08	0	0.541	-0.18%
no	0.08	30	0.542	—
yes	0.25	0	0.598	+10.33%

The distinction we make is between FPs that should have been NIL, and FPs where an entity should indeed have been found, but where the wrong disambiguation candidate was selected. At a confidence level of 0.5 we found 72% of FPs that should have been NIL with the original model, where the modified model had produced 76% should-be-NIL FPs. At the 0.3 confidence level we found 77% of FPs that should have been NIL with the original model, where the modified model generated 85% should-be-NIL FPs. This suggests that the issue is mostly related to phrase spotting, and could in particular be attributed to the reduction in the total corpus counts of surface form occurrence.

There is however also a decrease in recall that is not explained directly by the removal of entities from the model. The decrease in recall is likely caused by a technical issue. By removing all references to entities that are not in the vocabulary from the raw counts, we may have removed references to redirects and disambiguation pages that were strictly not in V, but that did refer to entities in V. And because these redirects and disambiguation pages are only taken into account in a subsequent model creation step, they would also not be taken into account in the surface form—entity co-occurrence model.

4.3.2 Merging Disambiguation Candidates

The main goal of combining the candidate entities from differently configured system outputs is to obtain a larger set of valid disambiguation candidates, while adding as few as possible invalid candidates. In the following results tables the merged configurations are shown in abbreviated form, with the relevant parameters of the primary system being followed by the configuration of the additional system.

For the short track, we test the effect of adding the candidates from capitalized input text to annotations that have been placed with a low phrase spotting threshold. As Tables 6 and 7 show, this operation added a comparable amount of valid and invalid interpretations to the original annotations.

The long track documents are only evaluated on the basis of individual disambiguated entities, and as such we are interested in adding candidates when none of the primary candidates is in the vocabulary. To do this, we use the multi-

Table 7: Error Analysis of the Candidate Merging

			Long Track			
Model	Conf.	Cand.	F_1	ΔF_1	Pr.	Re.
mod.	0.3	multi	0.632	-5.67%	0.595	0.761
orig.	0.50	multi	0.670	—	0.808	0.598
orig.	0.5	multi	0.670	0.00%	0.806	0.599
orig.	0.3	multi	0.713	+6.42%	0.735	0.740
orig.	0.30	multi	0.714	+6.57%	0.741	0.737

ple candidate filtering method that selects the top candidate that is in the vocabulary as disambiguation. The experiment is done with a well performing configuration as primary system, in order to first get accurately spotted phrases.

By adding the candidates that are generated by the modified model to those of the best run, an increase in recall is achieved, indicating that there are instances where the original model does not produce valid candidates and the modified model selects the correct entity. There are, however, more instances where NIL was correctly selected by the original model, but where an erroneous candidate is now selected by the modified model.

The same approach is taken by taking the spotted phrases from the original model with confidence values of 0.5 and 0.3, and by combining them (in turn) with candidates that can have disambiguation scores as low as 0.10. In both cases this results in a slight increase in recall, but a comparable drop in precision. This indicates that merging candidates can be useful to increase the overall performance, but that the merged candidates need to be re-ranked to reliably select the entity that is actually mentioned.

4.4 NIL detection classifier

In this section, we evaluate the performance of the designed classifier for detecting the NIL entities. We use the SIGIR ERD Challenge 2014 datasets for constructing the train data. For the long track, this dataset contains 70 documents with their annotated entities. For the short track, there are 91 short queries with their corresponding entities. We have extracted the features described in Section 3.5 from these datasets and use them for training classification models. Then, we have used k-fold cross validation to evaluate the performance of the learned models. We train separate models for short and long tracks using their corresponding train data. We use SVM [?] as the classifier.

The results of these sets of experiments are shown in Table 8 (top half). We use the Modified model described in Section 3.3.1 as the baseline. As can be seen the proposed NIL detection classifier improves that baseline method in terms of precision and $F_1 - score$. However, it has a bad effect on recall. That is because the NIL detection classifier determines some correct entities as NIL. Although the recall decreases, the overall performance has been improved. The reason of this improvement is that the classifier successfully detects entities that the spotter has mistakenly considered them as entities. Our analysis on the results also show that the percentage of False-Positives which should be detected as NIL entities without using the classifier is 0.84. However, when we use the classifier this percentage decreases to 0.79. This result indicates that using the NIL detection classifier is effective for determining the wrongly annotated entities.

Table 8: Results of the NIL Detection Classifier

		Long Track		
Method	Pre.	Rec.	F_1	ΔF_1
Modified model	0.54	0.527	0.533	—
Classifier	0.641	0.467	0.541	+2%
		Short Track		
Method	Pre.	Rec.	F_1	ΔF_1
Modified model	0.399	0.349	0.364	—
Classifier	0.405	0.377	0.373	+2%

Table 8 (bottom half) shows the results of using NIL detection classifier for the short track. As can be seen, the classifier successfully detects and filters the NIL entities. Although, in the short track the system improves all of the metrics, like the long track, its main contribution is on the precision. The results show that the designed classifier is effective in both short and long tracks and it could be exploited in EL systems to detect and filter out NIL entities.

5. CONCLUSIONS

In this paper, we proposed a system for automatically detecting entities of a given text in the Entity Linking task. As a baseline, we selected DBpedia Spotlight which is an open source and highly configurable entity recognition system. We focused on the errors that the baseline system made and identified where there is room for improvement, and how it may be achieved.

Our main contribution has been to explore how a general-purpose EL system, such as DBpedia Spotlight, can be focused to produce more accurate results on input texts with specific characteristics. We suggest "focusing" here as a lens-metaphor, where a large lens—the general-purpose EL system—can be complemented with smaller lenses to achieve a better focus for particular purposes, without needing to make structural modifications to the core system.

Since entity mentions that are not recognized can never be disambiguated, we first concentrated on DBpedia Spotlight's spotter and experimented with input text pre-processing to increase its coverage. With the same goal in mind we have examined the effect of different parameters configurations, of modified statistical models, and of merging their generated candidates.

Another important factor of the performance of an EL system is the degree of ambiguity in detecting candidates entities of possible surface forms. As there may be several candidates for a given surface form, the EL system should effectively find the correct candidate which matches with the use of the surface form in its context. To solve this problem, we designed a classifier which automatically assigns a score to each candidate of a surface form and ranks them according to these relatedness scores. Additionally, this classifier has the capability of filtering out the erroneously detected entities, namely NIL entities, which we found to be a large problem with the default DBpedia Spotlight configuration.

The designed system successfully participated in the SIGIR ERD Challenge 2014. The experiments on the challenge's datasets shows the effectiveness of the proposed approaches. The baseline configuration proved to be strong already, but could be made more accurate by parameter tuning. Specifically for the input text category of lowercased web search queries, we found that artificially capitalizing the input tokens had a large positive effect on performance.

The modified models successfully addressed the problems they were intended to solve, but unfortunately also introduced additional issues. We have described these issues because they can provide valuable lessons for similarly-oriented future work. Combining the candidate entities from differently configured system outputs had only a minor effect on the overall performance. This approach to obtaining a more complete set of candidate entities is likely of more use when it is followed up with re-ranking experiments.

The authors hope that the work presented in this paper will encourage more inquiry into the focusing of general-purpose Entity Linking systems for specific purposes. The modifications that we found to have the greatest positive influence on EL performance are, conveniently, relatively simple. This should be beneficial to researchers who are mainly interested in the applications of Entity Linking, rather than its mechanics.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (ExPoSe project, NWO CI # 314.99.108; DiLiPaD project, NWO Digging into Data # 600.006.014).

6. REFERENCES

[1] D. Carmel, M.-W. Chang, E. Gabrilovich, B.-J. P. Hsu, and K. Wang. ERD 2014: Entity Recognition and Disambiguation Challenge. *SIGIR Forum.*

[2] H. Dai, C. Wu, R. Tsai, and W. Hsu. From Entity Recognition to Entity Linking: A Survey of Advanced Entity Linking Techniques. In *The 26th Annual Conference of the Japanese Society for Artificial Intelligence*, pages 1–10, 2012.

[3] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes. Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 3–6, Austria, Graz, 2013.

[4] P. Mendes, J. Daiber, M. Jakob, and C. Bizer. Evaluating DBpedia Spotlight for the TAC-KBP Entity Linking Task. In *Proceedings of the TAC-KBP 2011 Workshop*, Gaithersburg, USA, 2011.

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

[6] P. P. N. Mendes, J. Daiber, R. Rajapakse, F. Sasaki, and C. Bizer. Evaluating the Impact of Phrase Recognition on Concept Tagging. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC*, pages 21–27, 2012.

[7] H. Nguyen and T. Cao. Named Entity Disambiguation: A Hybrid Approach. *International Journal of Computational Intelligence Systems*, 5(6):1052–1067, 2012.

[8] D. Rao, P. McNamee, and M. Dredze. Entity linking: Finding extracted entities in a knowledge base. In T. Poibo, H. Saggion, J. Piskorski, and R. Yangarber, editors, *Multi-Source, Multilingual Information Extraction and Summarization*, pages 93–115. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[9] W3Techs. Usage of character encodings broken down by content languages, 2014.