WebSAIL Wikifier at ERD 2014

Thanapon Noraset, Chandra Sekhar Bhagavatula, Doug Downey
Department of Electrical Engineering & Computer Science, Northwestern University
{nor.thanapon, csbhagav}@u.northwestern.edu, ddowney@eecs.northwestern.edu

ABSTRACT

In this paper, we report on our participation in Entity Recognition and Disambiguation Challenge 2014. We present WebSAIL Wikifier, an entity recognition and disambiguation system that identifies and links textual mentions to their referent entities in Wikipedia and later maps to the ERD set of target entities. WebSAIL Wikifier can handle both short text and long text queries. The system uses hybrid rule-based NER to discover mentions and a supervised machine learning approach to disambiguate their entities. The system achieves F1 of 0.641 and 0.687 on short track and long track respectively.

1 Introduction

Entity linking is the task of identifying and linking phrases (referred as mentions) in text to their entity in a target knowledge base such as Wikipedia and Freebase. The task is an important step to develop a better understanding of an input text by leveraging information of the entities linked in the text. Entity linking systems can benefit many NLP applications such as knowledge acquisition from text, machine translation, and search engines.

Entity linking is a task of recent interest, particularly linking to Wikipedia pages as referent entities (often referred as “Wikification”) as introduced by Bunescu and Pasca 2006 [1]. Given a document, Wikification systems must recognize phrases that can be linked to Wikipedia (referred to as mentions) and then disambiguate the mentions to their referent concepts [4, 10]. The mentions are often discovered using resources derived from Wikipedia (e.g., hyperlinks, page titles, redirects, and disambiguation pages), or an external name entity recognizer (NER). When performing disambiguation, systems also utilize statistics over the Wikipedia link graph to estimate the “coherence” of the output mentions [13, 17].

The system described in this paper focuses on the Wikification task, and performs name entity recognition, disambiguation, and linking. In this paper we report our participation in 2014 Entity Recognition and Disambiguation Challenge (ERD2014) for both the short and long tracks [2]. We present our system, WebSAIL Wikifier, an improved system over that reported in TAC KBP English Entity Linking 2013 [11]. Adapting the WebSAIL Wikifier to ERD required overcoming two new challenges particular to the ERD competition. First, our system covers all the entities in Wikipedia, whereas ERD targets only a subset Freebase entities that are also in Wikipedia. For example, a phrase “Namibian constitution” is a valid mention of Constitution of Namibia, a Wikipedia page. However, this concept does not exist in the ERD target entity database. Thus, we attempt to adapt the WebSAIL Wikifier to output only “Namibian” as mention. Furthermore, in ERD unlike previous competitions, there is a time limit for each query. This prevents our system from using some of its more expensive resources, like models of the context surrounding each entity built from the Wikipedia link graph.

The paper proceeds as follows. We start by explaining the data resources we utilize in ERD2014. These resources are shared for both short and long tracks. Next, we describe the WebSAIL Wikifier’s steps of identifying mentions, disambiguating their referent concepts, and linking for both short and long tracks separately. Finally we discuss our results and preliminary analysis.

2 Resources

This section describes information that we extracted for use in our system. Most of the resources in this section are used for the Wikification task and extracted from English Wikipedia as of November, 2013.

2.1 Entity Surface Forms

An entity surface form is textual phrase that refers to some entity. Collecting entity surface forms is useful for all steps including mention identifying, concept candidate generation and also disambiguation. We extracted entity surface forms from two major resources. First, we used the hyperlinks in English Wikipedia articles, where each hyperlink gives us a surface form (anchor text) mapped to an entity (Wikipedia page). We aggregated the surface forms from more than 100 million hyperlinks to get corpus-level statistics. For example, the phrase “Chicago” appears 16,884 times as an anchor text in Wikipedia, each linked to one of 289 pages including the city (most of the time), the movie, the music band, and etc. To incorporate hyperlinks outside Wikipedia, we also used
the external partition of Google Cross-Lingual Dictionary for English Wikipedia Concepts [14]. The data is already aggregated to yield the same statistics. However, to reduce noise, we removed surface forms that occur less than ten times.

In addition to hyperlinks, as in Cucerzan [4], we use three other sources to extract surface forms to entity mappings: page titles, redirect page titles and disambiguation pages.

Apart from using a page title as a surface mention to itself, we look for page titles in one of the two formats: $X_t, Y$ and $X_t(Y)$. For a page title, $t$ in one of these formats, we extract “$X$” and “$Y$” as additional mentions for the concept $t$. For example, from the title “Baltimore_County,_Maryland”, we extract “Baltimore County” and “Baltimore County Maryland” as additional mentions for the concept “Baltimore_County,_Maryland”. We also extract titles of redirecting pages as mentions for the entity that they redirect to. For example, the title “Chicago_Illinois” redirects to the page “Chicago” - from which we extract “Chicago, Illinois” as a mention for the entity “Chicago”. A disambiguation page on Wikipedia contains a surface-form in the title and a list of concepts that the surface usually refers to. We used the “Disambiguation Links” dataset from DBpedia that contains a list of all links extracted from disambiguation pages from Wikipedia.

For example, from the “Chicago_(disambiguation)” page, we extracted additional mentions for the surface “Chicago” to entities “Chicago_(CTA_Blue_Line_station)”, “Chicago_River” etc. These three sources help us expand our surface concept map, thus increasing recall of the system.

This surface form to concept mapping dataset is utilized for mention extraction and candidate generation. We created a word-level Trie using this set of surface form to entity mappings. Each surface form is inserted into the trie such that the terminal node contains a pointer to the list of Wikipedia entities that the surface form maps to. Thus, this trie helps us find surface mentions as well as get a list of candidates for each mention found in the a given text query.

2.2 Semantic Relatedness

Entities found in the same document are generally related to each other. Automatically estimating the relatedness of entities is useful for disambiguation and linking. Witten and Milne proposed Semantic Relatedness (SR) to capture this relationship [17]. By considering in-links and out-links of Wikipedia articles, SR is defined over a pair of entities. To avoid computation cost during runtime, we precomputed all non-zero SRs between every pair of entities, and stored these in file. Our SR measure is slightly different from Witten and Milne’s, as we give more weight to hyperlinks appearing in the overview section of articles. This implementation developed in Hecht et al. 2012 [6].

2.3 Language Models

Besides hyperlinks and page titles, the vast majority of Wikipedia data is in its article text. To make use of this efficiently, we use SRILM to construct Kneser-Ney backoff n-gram models of all Wikipedia articles [7, 15]. We have two models in our system. The first model is constructed from parsed Wikipedia articles. This model gives us statistics of phrases in Wikipedia and helps us in name entity recognition and linking. For example, we know that “Chicago” is not as common a word as “the”, and the system should pay attention to it. The other model is constructed to aid disambiguation. The model allows us to query the probability of a phrase, $Pr(w_{i-2} \cdot w_{i-1} \cdot t \cdot w_{i+1} \cdot w_{i+2})$, where $w$ is a word surrounding a mention, and $t$ is a concept that we want to disambiguate to. For example, a phrase “I live in the Chicago area.”, $Pr(\text{“in” “the” “t “area” “t”})$ yields higher probability if $t$ is the city than the movie or the train station. To allow this query in an n-gram model, we replace all hyperlinks in the article text with the linked Wikipedia page, and build the n-gram model as the first one. To query an n-gram probability from the language models, we use BerkeleyLM library [12].

2.4 ERD2014 Data

Since WebSAIL Wikifier is designed for Wikification, we need additional resources to map between entities that our system outputs to ERD2014 target entities, and also to filter out mentions that are less likely to be ERD’s annotation. ERD2014 provides an entity map between Freebase and Wikipedia, this solves the mapping issue. To limit our result to the target entities, we construct a set of possible surface forms by selecting surface forms (described earlier) that have at least a link to one of the target entities. The set contains less than 6 millions surface forms. Note that the statistics remains the same because we do not remove other entities from the system.

In addition, we realized person names are difficult mentions to disambiguate because they usually appear only partially in the text (only the first name or last name). We created a resource of person names from Freebase by selecting only entity names of the “person” type. This allows our system to recognize and deal with person names. These resources is mainly used in long track (Section 3.1.2).

2.5 Yahoo! Webscope Dataset

To train a ranker for disambiguation of short text mentions, we used the Yahoo! Webscope Dataset “ydata-search-query-log-to-entities-v1.0” [http://labs.yahoo.com/Academic_Relations] that contains mentions extracted from search queries annotated with Wikipedia entities that they refer to. The details of features used for training the ranker are given in Section 3.2.1.

3 System Description

We approach the entity linking task in 3 major steps. Given an input document, we first extract mentions along with their set of concept candidates (a set of possible entities). For mentions that have more than one concept candidates in the set, we need to disambiguate its referent entity by selecting one concept in the set. Finally we decide whether to link (include) them into the output. Both short and long tracks follow these 3 steps but the details are slightly varied because of the characteristic difference in the input document.

3.1 Mention Extraction and Candidate Generation

In this section we describe our approach to extract mentions, also referred as name entity recognition in ERD2014. Given
an input text, the goal of this step is to output a set of potential mentions, each associated with a set of concept candidates.

3.1.1 Short Track Mention Extractor
For extracting mentions in short text, we use the trie described in Section 2.1 and perform maximal matching of the text query. We start at the root of the trie and traverse through edges matching successive words in the query. If there is no edge matches the current word, we return the longest mention found so far and restart this matching process from the root node at the current word. For example, for the phrase “american airlines baggage size restrictions”, we go through the query one word at a time. We start at the root of the trie and traverse through the edges “american” and “airlines”. We do not find an edge labeled “baggage”, which means there is no link on Wikipedia with the surface “american airlines baggage”. We output “american airlines” as a mention and restart matching from the root node and find an edge for “baggage” and so on. Using our dataset of surface mentions to entity map described in Section 2.1, we obtain candidates for the extracted mentions.

For the example query, we get “american airlines”, “baggage”, “size” and “restrictions” as possible mentions. Clearly, words like “size” are seldom linked in Wikipedia and are indicators of being a “bad” mention. Similar noisy phrases like “of the” are seldom found as links on Wikipedia and adding these to our trie results in noisy output from the maximal matching algorithm. To mitigate this problem, we filter the mentions extracted from maximal matching by the probability of the surface $s$ being linked in Wikipedia. This is defined as:

$$P(s) = \frac{\# \text{s linked}}{\# \text{s occurs in text}}$$  

(1)

It is hard to efficiently compute the count of all possible phrases $s$ in Wikipedia, so we use an estimate of the count using the n-gram probability of the occurrence of a phrase using the language model described in Section 2.3. We use the score defined below and empirically set a threshold of +10.0 to filter out mentions that we do not want to include in our final output.

$$mlScore(s) = \log(\# \text{s linked}) - \log(\text{P}(s \text{ occurs in Wikipedia}))$$  

(2)

3.1.2 Long Track Mention Extractor
Similar to the method in short track, we also use trie and its maximal matching. However, the input text is larger but richer with capitalization and sentences. To fully exploit information in the text, we first use Stanford Core NLP library to break the text into sentences and add part-of-speech tags to each word [16]. Since name entities are usually capitalized and mostly noun or adjective, we start the matching at a capitalized word (ignore leading determiner), and only end at noun, number, or adjective word. For example, a sentence “The Namibian constitution which provides in article 21(1)(j)” will give us “Namibian constitution” because “The” is a leading determiner and “Namibian constitution which” does not have concept (even if it does, “which” is not a valid endpoint). This hybrid NER perform reasonably well in Wikification task, but not in ERD because of the limitation of entities. Thus we only allow a mention whose surface is in the ERD surface form set described in Section 2.4. In the example, “Namibian constitution” is not in the set, and the matching falls back to only “Namibian” which is an only output mention for the sentence.

Many ambiguous mentions arise from partial surface matching such as last names and acronyms where their full form would have been non-ambiguous or at least easier to disambiguate. For example, a sentence “Gandhi is visiting UP”. From earlier method, the system know that “Gandhi” and “UP” are surface forms of mentions, but the system has to consider large amount of concept candidates and might not even consider the correct ones (in this case “Rahul Gandhi” and “Uttar Pradesh”). Fortunately, a document usually contains the full form. This gives us a chance to improve disambiguation result before actually disambiguating the mentions.

To keep ambiguity low, the system tries to expand surface forms of one-word mentions. The system selects extracted mentions such that its surface form is in the person name set (also described in Section 2.4). Then the system creates a set of first names and last names (remove ambiguous ones). Any one-word mention that is in the first name or last name sets, the system expands its surface form to the full form. The same technique is applied to acronyms where an acronym set is created from the first characters of mentions that its surface forms has more than one words. In the previous example, if “Rahul Gandhi” and “Uttar Pradesh” appears anywhere in the input text, the system will output the two full-form mentions instead of “Gandhi” and “UP”.

3.2 Entity Disambiguation
In this section, we describe our disambiguation technique for the extracted mentions from the previous step. A general method to disambiguate is to use a trained machine learning algorithm to rank the concept candidate set. Our system applies pair-wise ranking model where we compare between two concept candidates and predict which one is better as a referent entity for their mention. Our system uses logistic regression classifier from Weka [5, 8]. The goal of this step is to annotate an input mention with a referent entity. We output this by selecting the top-ranked concept candidate from the ranker. Note that the top-ranked is not necessary in the target entities, thus we select the highest concept candidate that is in the target entities.

3.2.1 Short Track Ranker
In Section 3.1.1, we described how we extracted mentions, along with their candidates, from short text queries using a trie maximal matching algorithm. To disambiguate these mentions, we use a machine learning algorithm trained on the Yahoo! Webscope dataset “/data-search-query-log-to-entities-v1L0” [http://labs.yahoo.com/Academic_Relations] described in Section 2.5.

We used a pair-wise Logistic Regression ranking model to rank the candidates of a mention. The top ranked candidate is chosen as the prediction for the mention. Table 1 lists the features used for training the ranker.
3.2.2 Long Track Ranker

For each mention returned by the previous step, we have to extract the same features used in the Short track (Table 1), but the feature extraction process consumes very long time. Since there are many mentions from the previous step and each has many concept candidates. Since the linker will discard many of them anyway (described in Section 3.3.2), we employ an initial pruning to remove mentions unlikely to be included in the output. This can be done by running a fast ranker and a linker that is tuned for recall.

The initial fast ranker is a baseline ranker where it ranks the concept candidates by their probability of entity given the surface form (probabilityRank in Table 1). After the linker discard most of the “bad” mentions, we then proceed to extract features and run the trained machine learning model. We use existing hyperlinks from Wikipedia to train our ranker. Note that one of the submitted system used features not listed in the table. The features are an average and maximum of semantic relatedness between the concept candidate and other top-rank entities from the initial ranker. Unfortunately, we did not have the correct version submitted in time.

3.3 Linking

We now discuss the final step of our system. Given a set of mentions or interpretations, a linker will decide whether to include each of them in the final output. This step is crucial because the actual entity might not exist in the our referent entities, and the output from the previous step is inevitably wrong. This is a common case in TAC KBP where half of the query mentions link to entities outside the list of target entities [9]. In addition, the previous step might have been forced to select unlikely concept candidate that is in ERD’s target entities. For instance, a sentence “he speaks Chinese.”, our system would have ranked Chinese language as a top-rank, but it does not exist in the target entities, Thus China might have been selected instead.

3.3.1 Short Track Linker

An interesting aspect of the ERD short track task was to generate multiple interpretations for a single query. For example, for the query “the secret garden”, following interpretations are possible:

1. The_Secret_Garden
2. The_Secret_Garden_(1949_film)
3. The_Secret_Garden_(1975_TV_series)
4. The_Secret_Garden_(1987_film)
5. The_Secret_Garden_(1993_film)

To generate multiple interpretations, we consider short text queries for which we find less than 5 mentions using the method described in Section 3.1.1. We threshold at 5 mentions because we think that it is unlikely that a combination of 5 mentions could have multiple interpretations and also our interpretation generation system does not scale very well with more number of mentions. In short track linker, we generate different interpretations, classify interpretations as good or bad and output only the interpretations which are good.

After we find and disambiguate mentions as described in Section 3.2.1, we generate multiple interpretations by finding the power set of the set of mentions found in the text. Each set in the powerset is now considered as an interpretations. For sets with less than 3 mentions, we further increase the number of interpretations by considering all combinations of the top 3 ranked candidates according to the mention disambiguation ranker described in Section 3.2.1. We then use a trained linker model that decides whether to output an interpretation or not. This linker model is trained on the features listed in Table 2.

We used Logisitc Regression as the binary classifier and output the ones whose probability of being good is high.

3.3.2 Long Track Linker

There are 2 main linkers in processing long text. The purpose of the first linker is to perform an initial pruning based

<table>
<thead>
<tr>
<th>Feature</th>
<th>Explanation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalPrior</td>
<td>$P(Entity</td>
<td>Surface) from hyperlinks in Wikipedia</td>
</tr>
<tr>
<td>externalPrior</td>
<td>$P(Entity</td>
<td>Surface) from hyperlinks outside Wikipedia</td>
</tr>
<tr>
<td>internalPriorNC</td>
<td>same as internalPrior, but case insensitive</td>
<td></td>
</tr>
<tr>
<td>externalPriorNC</td>
<td>same as externalPrior, but case insensitive</td>
<td></td>
</tr>
<tr>
<td>probabilityRank</td>
<td>A ranked order of candidate concept by a combination of the first 4 features</td>
<td></td>
</tr>
<tr>
<td>normalizedNgram</td>
<td>$Pr(w_{i-2} \ldots w_{i+2}/Pr(t)$</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: List of features used by our system’s rankers.
on the baseline ranking result. This is necessary as we discussed in Section 3.2.2 that we cannot afford to run our machine learning model to every mentions from the mention extraction. This initial linker is a machine learning model trained using the given annotations for long track. The features are varied from the surface form and link information to confidence of the baseline linker (listed in table 3). In our system we use SVM with probability estimation tuned for recall by introducing cost-sensitivity [3,18].

The final linker is to guarantee that we only output mentions linked to an entity in the target set and we are confident about the result. A simple linker for this only include the mention of the selected entity in the top-rank. This works well if the mention extraction generates high quality mentions and the disambiguation performs well. A more sophisticated linker considers more information and uses a trained machine learning model. We use the same set of features in the ranker (Table 1) combined with the previous linker (Table 3), and train the linker using the same dataset as the previous linker.

### Table 2: List of features used by Interpretation Linker

<table>
<thead>
<tr>
<th>Feature</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctRatio</td>
<td>This is the ratio of probability of currently chosen candidate to that of the top-ranked candidate as per the disambiguation model.</td>
</tr>
<tr>
<td>qnMatch</td>
<td>Boolean feature - true if the surface of the mention found and the query match.</td>
</tr>
<tr>
<td>subI12N</td>
<td>Boolean feature - true if there is another interpretation which has all the mentions from the current interpretation.</td>
</tr>
<tr>
<td>sr</td>
<td>Product of the SR between pairs of chosen candidates of mentions.</td>
</tr>
<tr>
<td>mScore</td>
<td>The n-gram score of the mention as defined in Section 3.1.1.</td>
</tr>
<tr>
<td>i12nSetMatch</td>
<td>Boolean feature - true if there is an interpretation which exactly covers the input query.</td>
</tr>
</tbody>
</table>

Table 3: List of features used by the initial linker in long track.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>numCandidates</td>
<td># of candidates</td>
</tr>
<tr>
<td>numSurfaceLinked</td>
<td># of times surface is linked</td>
</tr>
<tr>
<td>numWords</td>
<td># of words in surface</td>
</tr>
<tr>
<td>isStartSentence</td>
<td>whether surface starts sentence</td>
</tr>
<tr>
<td>linkProb</td>
<td>probability of surface is linked</td>
</tr>
<tr>
<td>rank</td>
<td>rank of candidate</td>
</tr>
<tr>
<td>isTopRank</td>
<td>whether candidate is top-rank</td>
</tr>
<tr>
<td>titleMatch</td>
<td>same in ranker’s feature</td>
</tr>
<tr>
<td>internalPrior</td>
<td>same in ranker’s feature</td>
</tr>
<tr>
<td>titleDistance</td>
<td>same in ranker’s feature</td>
</tr>
<tr>
<td>scoreDiff</td>
<td>difference between candidate’s score and the top-rank candidate score</td>
</tr>
</tbody>
</table>

4 Results and Preliminary Analysis

In this section, we describe the performance of our systems for both short and long tracks in the competition. Since we do not have a final annotation, we only provide a preliminary analysis based on what we have observed in our output and the given sample annotations.

4.1 Short Track Result

For short track, we tried two systems:

1. erdSt.recover: This system uses the mention linker described in Section 3.1.1 and the disambiguation model described in 3.2.1. It does not output multiple interpretations.
2. erdSt.recover.i12nLinker: In addition to the mention linker and the disambiguation model, this system also uses the interpretation linking model described in Section 3.3.1.

Table 4 shows the F1 scores of our two short track systems. Our final system is erdSt.recover. It must be noted that the F1 score of erdSt.recover.i12nLinker is calculated on the test set before the final testing phase.

The main challenge for the system was to learn the interpretation linker. There is very little data to train a system to choose an interpretation. It is therefore natural that the performance of erdSt.recover.i12nLinker is lower than that of erdSt.recover.

4.2 Long Track Result

For long track, we have submitted 3 systems in which we incrementally added more methods to improve the performance.

1. mk_H: This system runs mention extraction without person name and acronym surface expansion, then rank the concept candidates with a trained ranker with features in Table 1, and finally the simple linker (not a trained model).
2. baseline_mk_I: We added the surface expansion and a trained linker. However, we had a timeout issue when introducing the SR features, so this system only runs initial pruning (baseline ranker and initial linker).
3. mk_V: This system extends from the previous system. After the initial pruning, we re-rank the concept candidates again using a trained ranker with SR features, then finally a trained linker described at the end of Section 3.3.2. We discovered a technical problem in this system, but we could not get the correct version submitted in time.

Our mention extraction approach yields very high recall of 0.89 on the given annotation, but the precision is 0.44.
This is a desired behavior of the mention extraction. The system inevitably missed mentions that their surface form is not capitalized such as “alendronic acid” or “southern China”. The disambiguation and linking are very difficult as shown in the performance of mk II (F1 of 0.648). We solved many of the ambiguity by introducing the surface expansion, and significantly improved the result in baseline_mk IV (precision of 0.728, recall of 0.757). We noticed that some of the errors are avoidable if we introduced SR features. For instance, “World Cup” is linked to FIFA_World_Cup while the other entities are about cricket. Unfortunately we did not have a working version of this submitted in time (mk V).

Acknowledgments
The authors would like to thank ERD organizers in their effort for managing this contest.

5 References