ABSTRACT
This paper describes Neofonie NERD, our Named Entity Recognition and Disambiguation system submitted to the ERD Challenge 2014. The system uses a vector space model approach for disambiguation, based on the link structure of Freebase, in combination with precomputed statistical measures from Wikipedia and Freebase. It was originally developed for the German language and has now been adapted for English. We achieved 70.0\% F1-score in the final evaluation, which is 5.7 percent points above the average of all participating teams.

Categories and Subject Descriptors
I.2.7 [Artificial Intelligence]: Natural Language Processing—Text analysis

General Terms
Algorithms

Keywords
entity linking, named entity recognition, named entity disambiguation, entity references, knowledge base, Freebase

1. INTRODUCTION
Named Entity Recognition and Disambiguation is an essential step in information extraction and a building block for further text analysis tasks and text understanding. While Named Entity Recognition (NER) has been widely investigated for a long time, Named Entity Disambiguation and Entity Linking came up only the last years with the availability of huge knowledge bases and Linked Open Data sources like Wikipedia, Freebase, and DBpedia [7].

The task of Entity Linking is to annotate mentions of entities in a text with links to some knowledge base. The most challenging part is to detect the appropriate entity for ambiguous mentions depending on the text context.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ERD 14 July 06-11 2014, Gold Coast, QLD, Australia
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3023-7/14/07 ... $15.00.
http://dx.doi.org/10.1145/2633211.2634358

2. THE NEOFONIE NERD SYSTEM
In this section we describe our system in detail. First, there is an offline processing step, that generates the lexica and statistical measures derived from Wikipedia and Freebase dumps, which are then used for disambiguation. The second part describes the actual processing step, in which the input text is analyzed and annotated with named entities.

2.1 Lexicon Generation
2.1.1 Wikipedia Processing
The Wikipedia text corpus is a great source for generating synonyms, common inflections as well as abbreviations...

1http://alexandria.neofonie.de
or alternative spellings. Therefore in a pre-processing step we collect all Wikipedia inter-links \(u \in (s, f, uri)\) with an anchor text as surface form \(s\) linking to a Wikipedia article with an URI \(uri\). In addition to the anchor texts we also gather the Wikipedia title itself and all the article’s redirects to generate further surface forms. As a result we get ambiguous surface forms each linking to several Wikipedia articles.

**SF-URI-Probability.** Based on the Wikipedia corpus we estimate the probability \(P(\text{uri} | s)\) of a surface form \(s\) linking to a Wikipedia article with an URI \(uri\) using the maximum likelihood estimate: We count the links \((s, f, uri)\) pointing to a specific Wikipedia article defined by \(uri\) and divide it by the number of all links \((s, f, *)\) with a surface form \(s\).

\[
P(\text{uri} | s) = \frac{\text{count}((s, f, uri))}{\text{count}((s, f, *))} \quad (1)
\]

**Keyphraseness.** Within the collected surface forms there exist terms that occur very often without linking to any article. As an example the surface form *Over* refers to several places and to other articles. But most likely it occurs as a preposition. In order to identify such terms we estimate the probability that a given surface form \(s\) occurs in a link by again using the maximum likelihood estimate counting the number of occurrences with \(s\) being an anchor text in a wikilink \((s, f, *)\) divided by the total number of occurrences of \(s\) in the whole Wikipedia text corpus. Derived from a similar approach of the Wikify system described in [10], that estimates this probability on the base of extracted keywords, we call it *keyphraseness*.

\[
\text{keyphraseness}(s) = \frac{\text{count}((s, f, *))}{\text{count}(s)} \quad (2)
\]

Surface forms with a very low keyphraseness are discarded from the lexicon. As a result we might lose entities in the process of spotting and disambiguation (lower recall) but on average we gain a higher precision by reducing the number of false positive annotations.

### 2.1.2 Freebase Processing

Taking into account relationships between entities helps to identify correct entities according to a document’s context. For this purpose the Neofonie NERD system evaluates entity relations derived from the Freebase knowledge base.

**Related Entities.** In our system two entities are in a relation if they are connected with a direct link. We neither distinguish the relation type (the Freebase property) nor do we take into account other metrics representing some strength of relatedness. This simple approach has been proven to work well with a huge cross domain knowledge base like Freebase containing millions of entities being well linked. An improvement could be to define a list of Freebase properties that should be ignored.

Freebase uses so called compound value types\(^2\) (CVTs) to model complex relations. In this way each marriage relation is represented by an arbitrary entity that links to both spouses as well as to other optional attributes like a start date, an end date, and the location of ceremony.

\(^2\)https://developers.google.com/freebase/guide/basic_concepts#cvts

To retain such indirectly modeled relations we materialize additional links: We compute the cross product between all entities being connected with the same CVT entity.

To enable a fast lookup the retrieved entity relations are stored in a key-value structure with each key representing an entity and the value referencing a list of related entities.

**Freebase Schema.** The Freebase schema consists of about 80 domains each containing unique types. In comparison to an ontology t-box the hierarchy of two levels is quite flat and there does not exist an inheritance of classes. Entities usually belong to several domains and a lot of types. These type and domain informations serve as a good indicator for the purpose of disambiguation and are therefore stored additionally.

**Inverse Link Frequency.** The entities in Freebase do not all have the same importance. There are some more prominent entities like *USA* or *Angela Merkel* and some rather special entities. A good indicator for the prominence is the number of links of an entity to other entities in Freebase. Similar to the Inverse Document Frequency (IDF), well known in Information Retrieval, we define the *Inverse Link Frequency* (ILF) of an entity \(e\) as follows

\[
\text{ilf}(e) = \log \frac{\max_{\epsilon \in E} |L_{\epsilon}^e| + 1}{|L_{\epsilon}^e|} \quad (3)
\]

with \(L_{\epsilon}^e\) being the set of all entities in Freebase linked to \(e\) and \(F\) the set of all Freebase entities. The ILF is similar to the entity popularity model proposed in [4].

The ILF can also be applied to Freebase types: Specific types with a small number of asserted entities like /basketball/basketball_player help to better identify entities within the same context than a rather general type /people/person.

### 2.2 Document Processing

The input text is processed in five stages. The first stage contains some linguistic preprocessing. In the following spotting step, mentions of entities are detected and linked to potential entity candidates from the lexicon. These candidates are then disambiguated in the next step. After that, a special handling for person names is done with supplementary heuristics. In the last step, a confidence score is computed for each entity candidate, the entities are filtered by lists and thresholds and finally returned as result. In the following, we describe these stages in more detail.

#### 2.2.1 Preprocessing

The input text is preprocessed with a sentence splitter and a tokenizer. Although these steps seem to be not very interesting, they are essential for the further processing and have great impact on the final results. For our first runs we used our own JFlex based tokenizer, that is also used in our German system. This tokenizer recognizes tokens with acronyms as one token, as for example in the term *Motorola’s*. Having annotated tokens like this, it is impossible to detect the entity *Motorola* with a token-based lexicon lookup. After using the OpenNLP\(^3\) sentence splitter and tokenizer with an English model, those problems were resolved. Nevertheless, there are many other of these pitfalls relying on bad tokenizing.

\(^3\)https://opennlp.apache.org/
2.2.2 Spotting

The first step in named entity recognition is spotting. This includes recognizing surface forms of entities in the text, generating entity candidates and particularly resolving overlapping matches.

Since our lexicon contains about 7 million surface forms, an efficient storage and lookup is crucial. After experimenting with some in-memory index structures, we now use Redis\(^4\) for storing the surface form index.

For entity lookup, the index is queried by token with a Longest Match Strategy. This approach results in a reduced set of promising surface forms in comparison to using an all match strategy. It performs very well for the vast majority of spotted entities. Nevertheless, this strategy is not always optimal, since it is possible, that the longest detected lexicon entry is rejected in a later step, whereas a surface form contained therein would have been the expected match. Therefore all rejected candidates are spotted again after the disambiguation phase.

The result of the spotting step is a list of mentions found in the text, together with a list of entity candidates for each mention.

2.2.3 Disambiguation

For resolving ambiguous entities, it is necessary to consider the text context, an entity candidate appears in. The underlying idea is, that an entity usually occurs together with other entities, to which it is somehow related.

If, for example, the mention Michael Jackson appears in a text, the disambiguation component has to decide to which entity to link this mention. If only the precomputed SF-URI-Probability, described in section 2.1.1, is taken into account, this decision is clear, since the probability for Michael Jackson linking to The King of Pop is 0.97. But considering the text in figure 1, it is obvious, that another Michael Jackson is meant in that context. Since most other entities occurring in this text are linked in Freebase directly or indirectly to Michael Jackson, the English beer and whisky expert, it can be concluded that this is the correct entity for that mention. Therefore, Freebase relations are a good indicator for the relatedness of entities and can be used for the disambiguation of ambiguous entities.

Our disambiguation step is similar to the vector space model approach described in [9] but differs in the computation of the context vector and the weights for the vector entries. An entity \(e\) can be represented by a vector \(\vec{v}_e\) containing all its related Freebase entities. Not all entries in this vector have the same relevance. Entities like USA or Europe occur in many contexts and have many related entities, which makes them less suited for disambiguating other entities. Therefore we need to weight these entries by a prominence measure to assign greater weights to less prominent entries.

We use the Inverse Link Frequency computed from Freebase for this purpose. Let \(F = \{e_1, \ldots, e_n\}\) be the set of all Freebase entities. The entity vector \(\vec{v}_e\) for entity \(e\) is then defined as \(\vec{v}_e = (w_{e,1}, \ldots, w_{e,n})\) with

\[
w_{e,j} = \text{if}(e_j) \cdot l(e, e_j)
\]

and

\[
l(e, e_j) = \begin{cases} 
1 & \text{if } e \text{ is linked to } e_j \text{ in Freebase} \\
0 & \text{otherwise}
\end{cases}
\]

The relevance of an entity candidate in the given context is then determined by computing the similarity between the entity vector and a vector for the context. The context vector is based on the entity vectors of all unambiguous entities occurring in the document. An entity mention in the text is considered to be unambiguous if only one Freebase entity exists for that mention in the lexicon. Since most lexicon entries are ambiguous, we augment this set \(U\) of unambiguous entities with almost unambiguous entities, i.e. entities with probability \(P(u(r)\) above a certain threshold (we use 0.7 as threshold). This is the case for an entity like Berlin, which denotes several cities within our lexicon, but also some persons or even works. In most contexts however, Berlin refers to the German capital, and therefore can be used to disambiguate other, more ambiguous entities in the context.

The vector for the document context \(\vec{v}_{doc}(e)\) is computed by adding the vectors of the set \(U\) containing all unambiguous (and almost unambiguous) entities

\[
\vec{v}_{doc}(e) = \sum_{e \in U \setminus \{e\}} \vec{v}_e
\]

For disambiguating a mention in the document, we determine a score for each entity candidate \(c\) linked to that mention by computing the cosine similarity of the candidate vector \(\vec{v}_c\) with the document vector \(\vec{v}_{doc}(e)\). For the computation of the cosine similarity for unambiguous entities \(e \in U\), the vector \(\vec{v}_e\) has to be left out from the computation of the document vector. This is necessary, because it is possible that although \(e\) is considered to be unambiguous, it is not meant in the document context but some other unknown entity not contained in the lexicon. Therefore only the other unambiguous entities should be used for disambiguating \(e\).

Analogous to the described computation, we build a second vector for each entity based on the Freebase types for that entity and using the ILF for types. A second cosine similarity score is computed for the type vectors. Both similarity scores are then combined to form an overall disambiguation score. The candidate with the highest disambiguation score is chosen as the most likely entity for the mention in the given document context.

2.2.4 Person Name Heuristic

Usually, when a text is about a person, in the beginning the full name of that person is introduced, but further on the person is referenced with different, normally shorter surface forms of its full name. Usually the genitive of the last name or only the last name is used. Having this situation our current system would only recognize persons where the full name is given, but would miss the other references if the surface form is not contained in the lexicon. Therefore we applied the following heuristic: For each disambiguated entity of type person variants of its surface forms are generated. If any of that variant occurs in the text, it will be linked to that disambiguated entity. For example the entity with surface form Hams Meyer is expanded to Hans Meyer, Hans and Meyer and linked to Hans Meyer. In the case, where entities produce the same expansions, such that different linkings are possible, we decided to link to the entity...
Figure 1: Computation of the disambiguation score

with the higher disambiguation score. For example, given the sentence "Hillary Clinton and Bill Clinton are politicians. Clinton argued that...", where Hillary Clinton and Bill Clinton were disambiguated, but Hillary Clinton got a higher disambiguation value. Clinton is then mapped to Hillary Clinton.

2.2.5 Scoring and Filtering

At the end, the system computes a weighted score of keyphraseness, SF-URI-Probability and disambiguation score. All entities with a score below a certain threshold are filtered out. By that filtering step, a handling of NIL entities, i.e. mentions that do not refer to any known entity, is possible. Mentions that were misleadingly annotated because the correct entity is not contained in the lexicon will be filtered if their confidence value is below the threshold. This is often the case with person names such as John Smith. Although there are probably several persons named John Smith in the lexicon, the correct one is not contained in the knowledge base, as he is not prominent enough to have a Freebase entry. If the confidence score for all other John Smiths is below the threshold, no entity will be linked.

Since the lexicon is automatically generated from Wikipedia and Freebase, which are based on user generated content, it cannot be avoided that the lexicon contains errors. Therefore an additional filtering step can be used to filter out certain entities. This can be done in three different ways. First there is a stopword filter to filter out words, that are contained in the lexicon and occur often in texts, but denote almost never an entity, such as New. The second filter is a blacklist filter, that filters unwanted surface-URI combinations. Our lexicon contained for instance the entity USA for the surface form California. The third filter is a whitelist filter, that was added for the ERD challenge to restrict the resulting entities to the entities contained on the given entity list.

3. EVALUATION

3.1 Quantitative Analysis

We achieved with our first run, without any modifications and optimizations, 71.9% F1-score on the validation data. This run is denoted as baseline in table 1. The first modification was the filtering of the lexicon. In our system all Freebase entries are used as entity candidates. The ERD Challenge however restricted the set of expected entities to certain types. After removing all entities from the lexicon, that are not on the given entity list, the F1-score fell to 69.9%. This was due to a significant reduction of precision, from 78.3% to 73.8%, whereas the recall even increased slightly. The reason for this effect is, that although the filtered entities did not have any direct effect on the evaluation scores, they were used in the document vector for disambiguating other entities. By leaving out these entities, the disambiguation results probably get worse, which resulted in lower precision. For the further runs we used again the bigger lexicon and filtered the entities only at the end.

After some minor optimizations concerning lexicon generation, tokenizing, and byte offsets we achieved 75.4% F1-score with our final run, which corresponded to the fifth rank of all participants in the long track. In the final evaluation on the test data we achieved the sixth rank with a F1-score of 70.0%, with 76.0% precision and 64.8% recall. The results are above the average of all systems. While the precision is only 0.6 percent points over the
average, the recall is 7.4 points better. This results in a 5.7 percent points higher F1-score than the average.

Since we adapted our system to English quite recently, it has not been evaluated exhaustively. We observed minor issues e.g. with an incomplete stopword list, that, being adapted properly, increases precision with small effort. Another aspect of tuning regards the generation of related entities: Filtering out strong populated but undesired Freebase properties and CVTs could improve the disambiguation of popular entities.

3.2 Performance

Although performance does not matter for the challenge, our system is able to process a large number of documents efficiently. Our NERD was the second fastest system in the challenge, with a measured latency of 0.53. This is probably mostly caused by network latency. On a local machine (8 x Intel(R) Core(TM) i7 CPU 920 @ 2.67GHz, 48 GB RAM), we achieve a throughput of 23 doc/s for average-sized news documents, with a lexicon containing 6.8 million surface forms and using 11 GB RAM.

4. FUTURE WORK

In addition to the steps described above, our German system also includes an open domain NER model for persons and locations, based on Conditional Random Fields (CRF) [6]. The results of the lexicon based approach and the CRF are combined. As a result we also detect entities that are not contained in the knowledge base, which is for instance often the case with persons in regional news. Since the detection of these entities was not required in the ERD task, we left out the CRF models for the challenge. However we plan to train person and location models for English and integrate them into the described system.

Apart from training CRF models, which requires labeled training data, the adaption of our system to a different language is straightforward. For the English version, only the generation of surface forms from an English Wikipedia dump and the computation of the statistical measures on this dump were necessary. Therefore we plan to adapt our system also to other languages. An already ongoing work is the integration of Neofonie NERD into the MIA platform. MIA is a research project within the Trusted Cloud research program, that aims to build a market place for data and analysis [5]. It provides a web crawl of large parts of the German internet, together with analysis modules for analyzing and querying this data. By integrating NERD, it will be possible to execute the named entity disambiguation distributedly on a hadoop cluster and annotate a huge web corpus with entities. These entities can subsequently be used for further custom analysis tasks, for example to determine the number of occurrences of politicians and parties in news before elections by day.

5. CONCLUSIONS

We presented the Neofonie NERD system as a tool for the task of entity recognition and linking. A lexicon of surface forms, containing 7 million entries, is built up from the Wikipedia text corpus. This serves as a good base to detect a wide range of entities in the domain of common knowledge. Moreover the recognition of synonyms, different spellings and abbreviations is supported.

In order to tackle the most challenging part of disambiguating retrieved entity candidates, we leverage the document’s context by applying the vector space model based on unambiguous entities and their related entities. We described our pre-computed measures of SF-URI-Probability, keyphraseness, and Inverse Link Frequency that support the process of disambiguation.

Neofonie NERD proved to be competitive with the other systems, that have been submitted to ERD Challenge. Furthermore our system is capable to efficiently process a large number of documents. Current supported languages are English and German.

6. ACKNOWLEDGMENTS

This work has been partially funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) as part of the Trusted Cloud research program under grant agreement 01MD11017, ‘MIA-Marktplatz für Informationen und Analysen’ (MIA).

7. REFERENCES

