DAI Approaches to the TAC-KBP 2011 Entity Linking Task

Danuta Ploch, Leonhard Hennig, Ernesto William De Luca, Sahin Albayrak
DAI-Labor, TU Berlin
Berlin, Germany
`firstname.lastname@dai-labor.de`

Abstract

This paper describes the DAI approach to the TAC-KBP 2011 Entity Linking Task, which this year introduced the subtask of creating novel knowledge base (KB) entries from name mentions referring to entities unknown to the KB (NIL clustering). For the entity linking task, our system uses disambiguation features that are based on KB information as well as on semantic similarity relations between named entities identified in the document context and in KB entries. To solve the NIL clustering task we implemented a three-stage approach which aims to improve upon an initial name mention string similarity clustering by introducing separate clustering steps for ambiguity and synonymy resolution. We describe implementation details of our system, and present a preliminary analysis of the results.

1 Introduction

The TAC Knowledge Base Population (KBP) track promotes research that supports the automatic discovery of information about named entities (NE) as found in unstructured texts, and the incorporation of this information into a given knowledge base (KB). Entity linking is an important first step towards addressing these goals. It aims to determine the correct KB entry for a name mention (of a person, an organization or a geopolitical entity) found in a document, or to decide that the entity referred to is not a part of the KB (NIL detection). The reference KB made available for the track is based on an October 2008 snapshot of the English version of Wikipedia, and encompasses approximately 800K entries derived from Wikipedia pages containing semistructured infoboxes. The task of entity linking raises several challenges: The name mention strings found in text can be ambiguous because entities may share the same name. Furthermore, the same entity can be referred to by different names, such as nicknames, aliases, and abbreviations. Thus, simple string matching does not suffice to solve this task.

In addition, the TAC-KBP 2011 challenge introduced the subtask of creating novel KB entries from entities determined to be NIL during the entity linking process. Addressing this task required systems to find ways to cluster name mentions referring to the same, unknown entity, in order to augment the KB with valid entries only.

The DAI-Lab of TU Berlin participated in the monolingual entity linking task at TAC 2011, which considers name mentions in English source documents. This paper describes the system we built for this task, our results, and comments on our approaches. We will first describe our approach to entity linking and NIL detection, and subsequently give details of our implementation of NIL clustering.

2 Entity Linking

Our approach to entity linking and NIL detection employs a three-step process. In the first phase, we generate a set of potentially correct KB entries (candidates) given a query, where a query is defined as a specific name mention in a given document. Then, we classify each candidate to determine the likelihood that it is the KB entry corresponding to the query name mention, thus inducing a ranking on the candidates. Finally, we employ another classification step to detect queries referring to NIL entities. Each of these steps is described in more detail below.

2.1 Candidate Generation

Given a query name mention s, the task of the candidate generation component is to retrieve a set of candidate entities $E(s)$ from the KB. Similar to many other systems, this step is geared towards high recall, and prefers a larger candidate set over a smaller one (McNamee et al., 2010; Lehmann et al., 2010).

Our approach to candidate selection is based on an inverted index of surface forms (name variants) and the named entities they refer to. The inverted index allows us to efficiently determine which entities share a word in common with the query name. We construct this index by processing a Wikipedia snapshot from May 2010. For each Wikipedia article describing a concept (i.e. any article that is not a redirect page, a disambiguation page, or any other kind of meta page), we collect a set of surface forms from normalized article titles, redirect page titles, disambiguation pages and the anchor texts of internal Wikipedia links (Cucerzan, 2007). We normalize titles by lower-casing, and removing punctuation as well as appositives. Redirect pages provide alternate spellings, abbreviations and synonymous forms of the named entity's name. Disambiguation pages often specify more general name variants, and list ambiguous acronyms. Anchor texts offer a rich variety of surface forms for a given named entity, but often also contain noise (erroneous links, rare misspellings, uncommon referents). The final index contains separate fields for "title", "redirect" and "form" name variants, which allows us to assign different weights to matches with each of these fields. The index also contains separate, tokenized versions of these fields in order to enable partial matches of multi-word expressions (e.g., 'George W. Bush' and 'George Bush'). The final index covers more than 3 million concepts and 6.7 million distinct name variants.[1]

[1]Note that we do not discard low-frequency surface forms or forms referring with low frequency to a particular entity. Discarding such spurious matches may however be beneficial to reduce the size of the index and to reduce noise when generating the candidate list.

The set of candidates is then determined by performing a search on the name variant fields. We implement a weighted search that assigns a large score to exact title matches, an intermediate score to redirect matches, and finally a low score for all other fields.[2] We limit the candidate set to the N highest scoring results according to the relevance score computed by the index search. We also considered a fuzzy search on the title and redirect fields in order to find KB entries with approximate string similarity to the surface form. However, this lowered the overall accuracy of our approach, and is therefore not included in the submitted system. We found that even though we do not explicitly search for approximate string similarity matches (e.g. misspellings), the rich set of name variants provided by Wikipedia typically still allows us to identify such matches.

The index we utilize to look up candidates is derived from a more recent snapshot of Wikipedia than the one used for constructing the TAC KB. Our candidate generation strategy may therefore return candidates not contained in the reference KB. To eliminate these invalid candidates, we introduce a post-processing step which removes all candidates from the set whose URL does not map to a valid KB entry.

2.2 Candidate Ranking

Given the set of candidates obtained in the previous step, the system needs to identify which of the candidates is the correct referent. Furthermore, in some cases none of the candidates is correct, and thus the system must decide if the query refers to a NIL entity. We cast candidate ranking and NIL detection as two separate classification tasks: The candidate classifier decides for each candidate if it corresponds to the target entity. Each candidate is represented as a feature vector encoding contextual and KB knowledge as well as comparisons of the two (see Section 2.3). The NIL detection classifier is based on features derived from the atomic features all candidates of a given query.

2.3 Features

Our system implements the entity linking approach presented earlier in Ploch (2011). It uses several different disambiguation features that are based on KB

[2]In our experiments, we found the best performance for the weights $title = 9$, $redirect = 6$ and $other = 3$.

Name	Type	Value	Feature
SFP	KB	Real	Probability of candidate entity for a given surface form
CS	KB	Real	TF-IDF weighted score for KB name given query name
BOW	Context	Real	TF weighted cosine similarity score for the document's text and the full Wikipedia article text of a KB entity
EC	Context	Real	TF weighted cosine similarity score for document context NEs and the NEs linked with a KB entity
LC_IN	Context	Real	TF weighted cosine similarity score for incoming links of document context NEs found in Wikipedia, and the incoming links of a KB entity
LC_ALL	Context	Real	TF weighted cosine similarity score for incoming and outgoing links of document context NEs found in Wikipedia, and the incoming and outgoing links of a KB entity
CR	Context	Real	PageRank score of KB entities computed based on a document graph of potential candidates of all NEs in a document

Table 1: Entity linking features

information as well as on semantic similarity relations between named entities identified in a document and the candidates' KB entries. In addition, we investigate a feature which aims to resolve all ambiguous entity mentions in a document at once, a problem which has largely been ignored in NED research. Table 1 lists a summary of the features.

All our features are based on resources created prior to running experiments, i.e. we do not consider any features that require web access during runtime (such as executing a Google query).

2.3.1 KB features

The first group of features focuses on information available in the KB, and ignores document context. The *surface form popularity (SFP)* feature encodes the likelihood with which a particular surface form refers to a given target entity. The entity distribution for a given surface form is determined from the link frequencies of internal Wikipedia anchors, including redirect and disambiguation pages (Han and Zhao, 2009).

The second KB-derived feature is based on our use of an inverted index for candidate generation. The *candidate selection score (CS)* is simply the relevance score of each KB entity as calculated by the weighted index search, which uses a modified tf-idf weighting scheme over index fields.[3] We found this feature to be very useful in our experiments on KBP 2009 and KBP 2010 datasets (see also Ploch (2011)).

[3]We utilize Lucene, for the exact scoring see `http://lucene.apache.org/java/3_4_0/scoring.html`.

2.3.2 Contextual Features

The group of contextual features is based on an analysis of the document context and the KB information. Many of the features in this group utilize the graph structure of Wikipedia to enrich the contextual representation.

We measured the *bag-of-words (BOW)* similarity between the query document and a candidate's KB text using the cosine similarity with TF-weighted bag-of-words representations (Bunescu and Pasca, 2006). We performed stemming and removed words occurring in a stop word list.

The *entity context (EC)* disambiguation feature filters the list of words and considers only named entities, as recognized by the Stanford NE Recognizer (Finkel et al., 2005), to construct the vector representation of the document context. The named entities found in the document context are represented by their corresponding Wikipedia page. If a named entity is ambiguous, we resolve it to the most popular KB entry according to the *SFP* feature described above. Candidates are represented by their (unambiguous) set of incoming and outgoing links. Vector representations of document and candidate context are weighted by the frequency of named entities and respectively link occurrences. The similarity between document and KB vectors is then calculated with the cosine measure.

The *link context feature (LC_IN)* is an extension of the *EC* feature. In some documents, there are very few named entities, and consequently the entity overlap between a document and KB entries can be very low. We therefore expand the document context by including all incoming links of the

Wikipedia pages corresponding to the named entities found in the document context. The incoming links correspond to Wikipedia concepts and named entities related to a named entity found in the document context. The feature hence incorporates notions of semantic similarity between named entities derived from Wikipedia's graph structure, based on the assumption that Wikipedia pages that refer to other Wikipedia pages contain information on the referenced pages or at least are thematically related to these pages. As before, we weight vector entries by their frequency. The representation of KB entries remains unchanged. As a variant of the *LC_IN* feature, we extend the document context by including all incoming and outgoing links of the found named entities (*LC_ALL* feature).

The features described so far disambiguate every surface form in a document separately, and resolve ambiguous context named entities with a simple popularity-based strategy. The *CandidateRank (CR)* feature aims to disambiguate all surface forms found in a document at once. To this end, we represent the document as a graph $D = (E(S), L(E(S)))$ where the nodes $E(S) = \cup_{s \in S} E(s)$ correspond to the union of all candidates of all surface forms in the document and $L(E(S))$ is the set of links between the candidates, as derived from the links found in Wikipedia. Thus, the graph D is a document-specific subgraph of the Wikipedia graph. In our implementation, edges and vertices are unweighted. We then compute the PageRank score (Brin and Page, 1998) for each candidate $c \in E(S)$ and choose for each surface form s the candidate with the highest PageRank score in D.

2.3.3 NIL Features

The features used to represent an ambiguous query for the NIL detection classifier are based on the atomic features computed for the query's candidate set. We calculate several different features, such as the maximum, mean and minimum, the difference between maximum and mean, and the difference between maximum and minimum, of the atomic features, using the feature vectors of all candidates in $E(s)$.

3 NIL Clustering

After the step of entity linking, we cluster all name mentions of identified NIL entities according to their referent.[4] To solve the NIL clustering task we implemented a three-stage clustering approach which first groups queries according to their surface form, then separates queries with ambiguous surface forms and finally aggregates all queries containing synonymous surface forms but referring to the same entity. We decided to introduce steps two and three to have the possibility of tuning the clustering parameters according to the subtask of ambiguity and synonymy resolution. Both steps are implemented using hierarchical-agglomerative clustering (HAC), but differ in their linkage criteria and the similarity threshold indicating whether two queries belong to one cluster. To determine the similarity between two query documents we represent the documents as concept vectors and calculate the cosine similarity between them. We calculate the document similarity for several features (Section 3.4) separately, and average the feature similarities to arrive at a final similarity value. In the following we describe each of the stages in more detail.

3.1 Initial Clustering

The first step of our clustering algorithms serves as an initialization step. We cluster all queries by comparing their surface form, since the surface form appears to be a good feature for distinguishing between different entities. Clustering by surface form (as tested with former TAC datasets) leads to a strong baseline and provides a good starting point for further refinement steps. A difficulty for this approach are unintended and slight name variations, emerging from spelling mistakes or spelling variants. For example different texts may either use the British or the American English way of spelling 'organisation', respectively 'organization'. Making a spelling mistake the German chancellor 'Angela Merkel' could be referenced by 'Angela Markel', or considering case sensitivity, someone could write 'Bank Of America' instead of 'Bank of America'. With high probability, in all examples the authors didn't intend to use a

[4]Since each query annotates only one entity mention, we use the expression query clustering and name mention clustering synonymously.

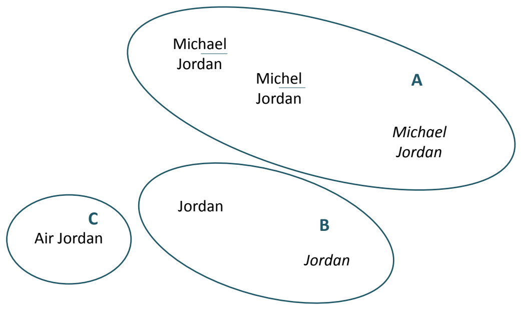

Figure 1: Clustering by surface forms. Surface forms with low Levenshtein distances are considered equal.

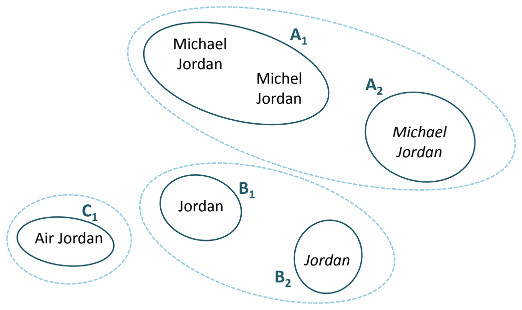

Figure 2: Clustering for ambiguity resolution

"real" synonym. For this reason we treat these kinds of name variants already in this step and consider queries with a low string distance to be equal.

To measure the similarity between two surface forms, we first convert them in a lower-cased form and then calculate the Levenshtein distance. We do not normalize the forms or expand acronyms like (Lehmann et al., 2010) for now, but we will take it into account in our future work. Surface forms with a Levenshtein distance $lv <= k$ are considered equal. Manual optimizations have shown that a suitable value for this parameter is $k = 1$. We cluster names shorter than 4 characters only if they are exactly equals to avoid grouping together surface forms like 'GA' (denoting Georgia) and 'PA' (for Pennsylvania), or acronyms like 'USA' and 'UPA' (e.g. for United Productions of America).

Figure 1 illustrates the resulting clusters for an example set of queries with the names 'Michael Jordan', 'Michel Jordan', 'Jordan' and 'Air Jordan'. The query names form three clusters A, B and C regardless whether the names in one cluster refer to the same entity.

3.2 Resolving Ambiguity

Surface forms may be ambiguous. For example the surface form 'Michael Jordan' may denote the basketball player and the researcher in machine learning. Looking at one result cluster after the first step, the (approximately) equal surface forms located in one cluster, may therefore denote different entities. For this reason we aim to slightly correct the clustering result of the first step by separating queries referencing different entities.

Our approach for resolving ambiguity employs a hierarchical-agglomerative clustering (HAC) algorithm. We apply HAC to each of the clusters of the first step separately, so that each cluster is split up into several clusters if its queries are ambiguous – for each surface form sense an own cluster. We initialized the algorithm by assigning each query its own cluster and merged clusters as long as two clusters exceeded a similarity threshold t_a. Since our approach is based on the assumption that there is one dominant entity, we used a relative low threshold $t_a = 0.1$ to favor bigger clusters and to only separate "obviously different" queries from the rest. In addition, we decided to perform single-linkage clustering. Because of its chaining phenomenon we assume to obtain larger clusters.

In Figure 2 the three clusters A, B and C of the first step were each separated into smaller clusters. Altogether, the queries were divided into 5 clusters A_1, A_2, B_1, B_2 and C_1 in this ambiguity resolving step.

3.3 Resolving Synonymy

An entity often can be referenced by different surface forms, for example the basketball player 'Michael Jordan' may be denoted by his nickname 'Air Jordan'. Up to here our clustering algorithm only considered surface form similarity and resolved their ambiguity. The goal of this step is to identify synonyms and to cluster them.

Each cluster of the second step contains entries for one entity. However, some of these clusters may refer to the same entity, as for example the cluster A_1 and C_1 in Figure 2. To resolve synonymous queries we again employ HAC. We initialize our algorithm

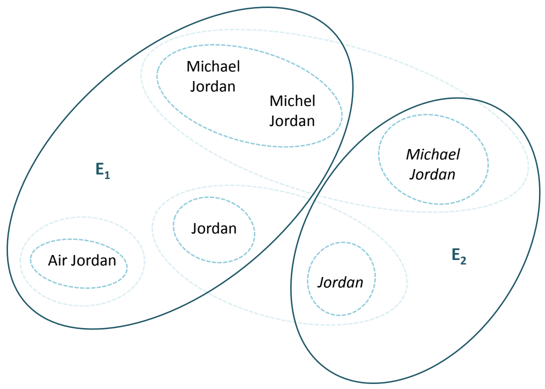

Figure 3: Clustering for synonymy resolution

at this point with all clusters of the previous step and perform further clustering in order to merge all smaller clusters into one bigger cluster which in the end contains all queries for one entity, even if they are denoted by synonyms. Analogously to the ambiguity resolution step we adjusted a threshold for cluster stopping. This time we selected $t_s = 0.7$ to be our threshold. We choose a higher threshold than before to merge only queries with very similar contexts which we assume to be synonyms. In contrast to the previous step we decided to employ average-linkage clustering to produce balanced clusters.

As shown in Figure 3 the five input clusters A_1, A_2, B_1, B_2 and C_1 are merged into two clusters, cluster E_1 containing queries for the basketball player and cluster E_2 with queries for the researcher.

3.4 Features

The initial clustering step is based on the Levenshtein distance between the surface forms of two queries. This Levenshtein feature (*LV*) simply measures string similarity and does not consider the entity context. When taking into account the entity context in the ambiguity and synonymy resolution steps, we utilize a subset of entity linking features for calculating the similarity between query documents: the cosine similarity between the word vectors (*BOW*), the context entities (*EC*), and the links between the context entities in Wikipedia (*LC_IN*). In both clustering steps we employ the same features. To calculate the cluster similarity we compute a combined similarity between query documents by weighting and averaging the aforementioned NIL clustering features:

$$sim_{all}(d_1, d_2) = \frac{\sum_{i=1}^{n} w_i f_i(d_1, d_2)}{\sum_{i=1}^{n} w_i}$$

We implemented two context construction strategies which differ in the size of the document context they use. The first strategy calculates all NIL clustering features based on the whole document (*CONTEXT_ALL*). Here we assume the entire document contains relevant information supporting the disambiguation of the query name. However, we observed in several queries of the development dataset that only the surrounding part of the query name refers to the target entity and that the remaining document parts rather seem to be confusing than helpful. Thus, we implemented an additional strategy (*CONTEXT_PART*) that only takes into account the sentences where the query name occurs, as well as the first three sentences of the document, in order to describe the context of the target entity more precisely. We included the first three sentences into the construction process because of the assumption that in many texts (especially in news) the most important information (such as full names of entities) is concentrated at the beginning of the text.

4 Experiments

We submitted three runs for the monolingual entity linking task. All runs used the KB article text, but none required web access during runtime to calculate Internet-based features. The three runs we submitted varied in the datasets used for training the classifier models, and in the way the document context was constructed for NIL clustering.

4.1 Training Data

Since our approach to entity linking is based on supervised learning, we used the KBP 2009 and KBP 2010 datasets as training data. The KBP 2009 dataset consists of 3904 queries, 69% of which refer to ORG entities, and 16% PER and 15% GPE. The KBP 2010 dataset has a total of 3750 queries, with 33% each for GPE, ORG and PER. The KBP 2009 and 2010 datasets differ slightly with respect to the distribution of KB and NIL queries (KBP 2009 57% NIL, 43% KB, KBP 2010 44% NIL, 56% KB),

Name	Type	Feature
LV	Real	Levenshtein distance between two lower-cased query names with more than 3 characters.
BOW	Real	TF weighted cosine similarity score for texts of query documents
EC	Real	TF weighted cosine similarity score for context NEs of query documents
LC_IN	Real	TF weighted cosine similarity score for incoming links of the context NEs of query documents

Table 2: NIL clustering features

and our observations suggested that models trained on the KBP 2009 dataset did not perform well when tested on the KBP 2010 dataset. We therefore decided to train two separate sets of models for candidate ranking and NIL detection. The first set of models uses only the 3750 queries of the KBP 2010 dataset, whereas the other set is trained on a total of 7654 example queries, of which 3960 (52%) were ORG, 1878 (25%) PER, and 1816 (24%) GPE. In this latter training dataset, NIL and KB queries roughly equally distributed (51% NIL, 49% KB).

4.2 Model Training and Parameter Settings

We use a Support Vector Machine classification algorithm (Vapnik, 1995) to train models for both subtasks of entity linking and NIL detection, utilizing the LibSVM implementation (Chang and Lin, 2001). For training the candidate classifier we label as a positive example at most one candidate from the set of candidates for a given surface form s, and all others as negative. For training the NIL classifier, we create a single feature vector per query, which we label as positive if the query refers to a NIL entity. Both classifiers use a radial basis function kernel, with parameter settings of $C = 32$ and $\gamma = 8$. Since the candidate classifier has to learn from a highly imbalanced dataset (there are many more incorrect candidates than correct ones), we set the weight of training errors on positive examples to 6. We experimentally optimized these settings on the KBP 2010 training dataset.

For training the final models, we split each training dataset into 10 equally-sized folds, and stratify the folds to ensure a similar distribution of KB and NIL queries. We normalize all feature values to be in $[0; 1]$. For each fold, we trained classifier models on 90% of the queries, once more using 10-fold cross validation to find more reasonable accuracy estimates. The models with the lowest cross-validation error were then used for testing the remaining 10% of the queries. This procedure was repeated for each test set of queries. For our submitted runs we selected the candidate and NIL classifier with the highest entity linking accuracy from the 10 model sets.

RUN	Settings
DAI1	- EL models trained on KBP 2010 queries (train + eval) - CONTEXT_ALL for NIL clustering
DAI2	- EL models trained on KBP 2009 and 2010 queries (train + eval) - CONTEXT_ALL for NIL clustering
DAI3	- EL models trained on KBP 2009 and 2010 queries (train + eval) - CONTEXT_PART for NIL clustering

Table 3: Settings for the three runs of the DAI-Lab

We adjusted the weights for our NIL clustering features on the 2250 KBP 2010 evaluation queries. We performed a grid search and found $w = 1.0$ for the features *BOW* and *EC* and $w = 0.5$ for the feature *LC_IN* to be the most suitable weights. The clustering thresholds t_a and t_s for ambiguity and synonymy resolution were also adjusted experimentally on that dataset. Observations have shown that $t_a = 0.1$ and $t_s = 0.7$ delivered the best results. For the initialization step we set $lv = 1$ so that only lower-cased surface forms with a very low Levenshtein distance were considered equal and therefore were clustered.

4.3 Run Details

Our first run (DAI1) uses only the 3750 queries of the KBP 2010 dataset, whereas the two other runs (DAI2, DAI3) utilized classifier models trained on a combined KBP 2009 and KBP 2010 dataset. The DAI1 and DAI2 runs used the CONTEXT_ALL strategy for context construction for NIL clustering. The DAI3 run varied from these in that the NIL clustering algorithm used strategy CONTEXT_PART for context construction.

	MAA	B^3 Prec	B^3 Rec	B^3 F1
DAI1	0.708	0.666	0.676	**0.671**
DAI2	0.700	0.659	0.670	0.664
DAI3	0.700	0.665	0.673	0.669
Best	-	-	-	0.846
Median	-	-	-	0.716

Table 4: B^3 scores and micro-averaged accuracy for our submitted runs (no web) compared to best and median performance.

	ALL	KB	NIL
DAI1	0.708	0.515	0.900
DAI2 & DAI3	0.700	0.5	0.899

Table 5: Micro-averaged accuracy for KB and NIL queries. The very low accuracy for KB queries decreases overall accuracy considerably.

4.4 Results

Table 4 summarizes the results of all three runs. The performance of the different runs is very similar, with negligible differences. B^3 precision and recall values of each of the runs are roughly equal, suggesting that our system strikes a good balance between these two measures. Overall results are rather low though, and slightly below the median performance of the 21 systems which submitted results for this task.

An error analysis showed that our system predicted NIL queries far too often. Around 34 % of the KB queries were classified as NIL, while only 10% of the NIL queries were classified as KB. The NIL prediction error was especially high for persons where we classified 60% of the KB queries as NIL. Our system introduced almost 10% of NIL prediction errors on KB queries due to its candidate selection method which missed the correct candidate for 106 queries. While the candidate generation strategy achieved a recall of 96.8% on the KBP 2010 dataset, and of 92.5% on the combined KBP 2009 and KBP 2010 dataset (see Section 4.1), the candidate recall on the KBP 2011 dataset dropped to 90.6%. Altogether, the KB accuracy of 50% decreases the overall accuracy considerably to 70% (Table 5).

We run several experiments with the Gold Standard answers after the challenge. To this end we used exactly the same system which produced the submitted results. In order to evaluate our NIL clustering approach independently from the entity linking task, we considered only the NIL queries of the Gold Standard answers and run our clustering algorithm only on these queries. We first calculated a baseline based on surface form similarity (exact matches of lower-cased query names) which achieved a B^3 F1-measure of 0.967. Compared to this strong baseline our system produced approximately the same result of B^3 F1=0.969. After examining our system outputs we observed that the synonymy resolution step slightly decreases the B^3 precision by 0.4% but has no effect on the recall. We obtained the best results after the ambiguity resolution step with B^3 F1=0.971. Table 6 summarizes the post-challenge NIL clustering results. The results confirm the efficiency of our NIL clustering approach.

	B^3 Prec	B^3 Rec	B^3 F1
All-in-one Baseline	0.002	1.000	0.003
One-in-one Baseline	1.000	0.718	0.836
SF Baseline	0.942	0.993	0.967
SF Initialization	0.920	0.994	0.956
Ambiguity Res.	0.965	0.976	**0.971**
Synonymy Res.	0.961	0.976	0.969

Table 6: B^3 scores for our post-challenge NIL clustering experiments using only NIL queries of the Gold Standard answers.

5 Conclusions and Future Work

The DAI-Lab of TU Berlin participated in the monolingual entity linking task at TAC 2011. Our best run achieved a B^3 F1 measure of 0.671, which is slightly below the median performance of the 21 systems which submitted results for this task. We found that we predicted NIL far too often and that the low accuracy on KB queries decreased the overall performance significantly. Our NIL clustering approach evaluated exclusively on correct NIL queries proved to achieve high B^3-scores. In future work we plan to improve our candidate generation strategy by including acronym expansion and alias detection. We further intend to extend our feature set and to adjust our SVM parameters to train models that generalize better for new NED datasets. For NIL clustering we want to explore if the usage of different features for the ambiguity and synonymy resolution steps improves the NIL clustering quality.

References

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. In *Proc. of WWW '07*, pages 107–117.

Razvan Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL-06*, pages 9–16.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Silviu Cucerzan. 2007. Large-Scale named entity disambiguation based on Wikipedia data. In *Proc. of EMNLP-CoNLL*, pages 708–716.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Ann Arbor, Michigan.

Xianpei Han and Jun Zhao. 2009. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proc. of CIKM '09*, pages 215–224, Hong Kong, China.

John Lehmann, Sean Monahan, Luke Nezda, Arnold Jung, and Ying Shi. 2010. LCC Approaches to Knowledge Base Population at TAC 2010. In *Proc. of TAC 2010*.

Paul McNamee, Hoa Trang Dang, Heather Simpson, Patrick Schone, and Stephanie M. Strassel. 2010. An evaluation of technologies for knowledge base population. In *Proc. of LREC'10*.

Danuta Ploch. 2011. Exploring entity relations for named entity disambiguation. In *Proc. of ACL 2011 Student Session*, pages 18–23.

Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.