Collective Search for Concept Disambiguation

Anja PILZ *Gerhard PAASS*
Fraunhofer IAIS, Schloss Birlinghoven, 53757 Sankt Augustin, Germany
anja.pilz@iais.fraunhofer.de, gerhard.paass@iais.fraunhofer.de

ABSTRACT
Name ambiguity is a major problem in information retrieval: The name "Metropolis" may refer to a movie, a physicist, or Superman's hometown. Recent work resolves ambiguity in natural language text by linking name mentions against the corresponding Wikipedia concept (Wikification). Standard methods comparing a single mention with the corresponding Wikipedia concept can potentially be improved by simultaneously considering all mentions in the input document. We propose a novel multiple assignment process based on a collective search over an inverted index that exploits the coherence of Wikipedia concepts. Based on this coherence, we compute the best fitting candidate concept for each mention and combine it with context information in a second search step. Using additional attributes an SVM then re-ranks the result of this search and estimates if a concept is not covered in Wikipedia. We give a unified view over the different performance measures used in other state-of-the art approaches and evaluate our approach on five benchmark corpora. On these corpora, our method has the most stable performance yielding similar or better results compared to other approaches.

KEYWORDS: Concept and Entity Disambiguation, Wikification, Natural Language Processing, Search and Ranking.

Proceedings of COLING 2012: Technical Papers, pages 2243–2258,
COLING 2012, Mumbai, December 2012.

1 Introduction

A major aim of search engines is the retrieval of information about concepts which may be any existing object, e.g. person, thing, notion, etc., with a designation or name. In natural language text however, many concepts share the same name and one concept may be referenced by different names. Consequently, a search based on pure string matching often yields many irrelevant results, such as a web page on Superman's hometown when indeed the user seeked information on the physicist Metropolis. Concept disambiguation which assigns the correct sense to the mention of a concept in a given context, can reduce the number of irrelevant results or group results by sense. The disambiguation of concept mentions is required in applications such as semantic search, but also many other areas like knowledge base construction or data base curation.

Recent work, for example (Ratinov et al., 2011), resolves name ambiguity by linking the name mention against the corresponding Wikipedia article, thus often terming the problem *Wikification*. For that, a name mention together with the features of its neighboring context is compared to the corresponding features of the Wikipedia article. If the difference between these features is small, a Wikipedia concept is linked to the mention and thus the name's ambiguity considered as resolved. A large number of features have been evaluated for concept disambiguation. Starting with simple bag-of-word descriptions more advanced features were developed characterizing the sense of surrounding words, e.g. topic model indices (Pilz and Paass, 2011). But often, approaches remained *local* and did not exploit the *global* coherence of candidate concepts.

In this paper we follow the *global* approach by simultaneously considering all mentions of an input document and jointly exploiting relations between potential concepts. We present a novel measure for concept coherence. We encode this information in a search index allowing fast and comprehensive access to the relational information present in large knowledge bases such as Wikipedia. One deficit of most current concept disambiguation methods is that they do not thoroughly handle the case when a concept mention is not covered by Wikipedia (nil-concepts). We use an SVM classifier to fine-tune the assignment and to detect nil-concepts. We discuss the various evaluation measures presented in other papers and apply our algorithm to five benchmark corpora. While fast and memory efficient, our algorithm yields similar or better results than its competitors and has the most stable performance of the compared methods.

2 Related Work

Concept disambiguation is closely related to the task of word sense discrimination (Schuetze, 1998), but in addition links the concepts to entries in a reference knowledge base which is often Wikipedia. Standard or *local* approaches like Cucerzan (2007) build word and feature vectors over the words occurring in a context window around the concept mention m and cluster them using similarity measures such as cosine similarity. Bunescu and Pasca (2006) correlate context words with Wikipedia categories to formulate a word-taxonomy kernel. This is used in a Ranking SVM which generates a ranked list of plausible Wikipedia concepts for a given context of a name mention m. Pilz and Paass (2011) showed that topic model indices instead of bag-of-word approaches provide a more informative context representation with better generalization properties.

Recent work on concept disambiguation follows a more *global* approach, where all concept mentions in a document are disambiguated collectively using a coherence measure that is usually

derived from the graph built over an existing knowledge base. Kulkarni et al. (2009) formulate concept assignment as an optimization problem that assigns concepts to mentions such that the mention-concept compatibility and global concept-concept coherence is maximal. They solve the problem using local hill-climbing and linear program relaxations, yielding favorable results on the **MSNBC** corpus (Cucerzan, 2007) as well as their own dataset **IITB**. Han et al. (2011) propose a graph-based collective concept linking method which can model and exploit the global interdependence between different assignment decisions. Ratinov et al. (2011) present the disambiguation model GLOW, a global approach that employs the normalized Google distance (Milne and Witten, 2008) as well as pointwise mutual information to measure the relatedness between concepts. To refine the assignment decision they additionally exploit the conditional probability that a concept belongs to a mention based on Wikipedia link information. Hoffart et al. (2011b) introduced AIDA which employs YAGO2 (Hoffart et al., 2011a) as an entity catalog and a rich source of entity types and semantic relationships among entities. They build a graph containing mentions from the input text and candidate concepts from the reference set as nodes. The edges are weighted capturing context similarities as well as coherence between Wikipedia concepts. Using a greedy algorithm they identify a dense sub-graph that contains exactly one mention-concept edge for each mention, yielding the most likely disambiguation.

We propose an approach that is based on a search index. The usefulness of search indices for concept resolution was also observed by Song and Heflin (2011) who present an efficient and scalable system for concept resolution on structured data. Opposed to our objective which is concept resolution in unstructured data, exploitable attributes are very different and often carry an inherent distinctive function. In the sequel, we give the details of our approach and compare it to a representative selection of four recent works showing that it is the most stable method yielding similar or better results on different benchmark corpora. Although all prior work shows improved results on benchmark corpora, none of them handles nil-concepts thoroughly. For specific tasks this might be appropriate, but in a more general setting this means a drastic simplification as most entities (e.g. persons) are *not* covered by Wikipedia.

3 Disambiguation as a Search Problem

We study the task of Wikification, i.e. concept disambiguation using Wikipedia as a reference knowledge base. We use the English version of Wikipedia[1] and represent it in the Lucene[2] search index **Wiki** that allows efficient search over the concepts contained in Wikipedia.
We resolve the ambiguity of a *mention* m in a text document through its assignment to a *unique concept* $c(m)$ described in Wikipedia, i.e. $c(m) \in \mathbf{Wiki} = \{c_1, ..., c_{|\mathbf{Wiki}|}\}$. If the true concept for m is not covered by an article in Wikipedia, then $c(m) \in \mathbf{C}_0$, the set of nil-concepts that we do not distinguish. Basically, **Wiki** contains all Wikipedia concepts apart from meta pages. We also excluded disambiguation pages since we assume that an assignment to such a page does not solve the task of name disambiguation. Furthermore, the varying usage of Wikipedia mark up language led to un-processable documents that are also not contained in **Wiki**. Thus, in the following, we distinguish between linkable concepts contained in the index $c \in \mathbf{Wiki}$, nil-concepts c_0 originally not covered by Wikipedia and ignored or missing concepts $\tilde{c}_0 \notin \mathbf{Wiki}$.

We assume the input to be a natural language text document with a collection of mentions $\mathbf{M} = \{m_1, \ldots, m_k\}$ to disambiguate. In the case of the benchmark corpora, these mentions are given. In other real-world applications, they can be provided by an automatic annotator, such

[1] Downloaded on September 1th, 2011.
[2] An open source search engine for large scale text collections, `http://lucene.apache.org/`

as a noun phrase or named entity recognizer (NER). Note that we do not restrict the mentions to named entities (persons, locations, etc) but also treat general concepts such as *bank* or *tree*.

To improve the individual disambiguation performance for each m_i, we simultaneously consider all mentions $\mathbf{M}$ to determine the best fitting *candidate concepts* $bestFit(m_i)$. We propose a disambiguation process that uses the search index **Wiki** to generate candidate concepts, as well as a supervised SVM classifier to adjust the ranking of these candidates and to detect nil-concepts c_0. This process consists of the following steps that are described in more detail in the following sections:

Step 1 Run a **collective search** using an **ensemble query** with terms from all mentions $m_1, \ldots, m_k$ to create sets of **potential candidates** $\mathbf{C}_i \subset$ **Wiki** for each m_i (Alg. 1.1-1.10).
Step 2 Compute the **cross coherence** over all candidates in the sets $\mathbf{C}_1, \ldots, \mathbf{C}_k$ to find related concept sets (c.f. Eq. 4), Alg.1.11-1.13).
Step 3 Determine the $bestFit_i \in$ **Wiki** for each mention m_i, based on the maximum cross coherence of each candidate in $\mathbf{C}_i$ (c.f. Eq. 6, Alg. 1.15).
Step 4 For each m_i combine the attributes of m_i and $bestFit_i$ into one query and search **Wiki**, which yields a set of improved concepts $\mathbf{C}_i^* \subset$ **Wiki** (Alg. 2.2-2.12).
Step 5 Apply an **SVM** classifier to all $\mathbf{C}_1^*, \ldots, \mathbf{C}_k^*$ for **re-ranking** and **nil-concept detection**, resulting in the final predicted concept $\hat{c}_i \in \mathbf{C}_i^* \cup \mathbf{C}_0$ for each m_i (c.f. Sec. 4.2, Alg. 2.17).

3.1 Concept Attributes in the Wiki index

Using the information stored in the article itself as well as Wikipedia's hyperlink graph, we enhance the representing concept $c \in$ **Wiki** with the searchable fields outlined in this section.

Name fields Special attention is given to name fields, since for unambiguous mentions the name is often sufficient for linkage. Each concept has a unique `titleLong` field which contains the title of the associated Wikipedia article. From this, we generate additional fields. The `title` field stores the part of `titleLong` that is not used as a disambiguation term (usually a qualifying term in parentheses). Abbreviations are generated via a simple heuristic and stored in separate `abbreviation` fields. As an example, the index concept representing the Wikipedia article *Michael Jordan (footballer)* has the fields: (`titleLong`, "Michael Jordan (footballer)"), (`title`, "Michael Jordan"), (`abbreviation`, "MJ"), (`abbreviation`, "M. Jordan") etc.

Furthermore, we add the redirect information from the Wikipedia redirect dump to the corresponding index concepts. In general, redirects provide a large resource of synonyms. In some cases, however, they can also be misleading, since they do not necessarily compose equivalence relations. For instance, *Ulrich Merkel* is a redirect for German chancellor *Angela Merkel*, but actually is the latter's spouse. Still, we consider all redirects without pre-processing, since a more well defined redirect scheme would already require a disambiguation step. The index concept for *Angela Merkel* is hence enriched with the field (`redirect`,"Ulrich Merkel").

Inspired by Ratinov et al. (2011), we create `meantBy` fields that, similar to redirects, provide concept names that may not be found in the article text itself. In a pre-processing step, we iterate over all articles in Wikipedia and analyze the pairs (c, m) of link target concept c and associated anchor text m. For each pair (c, m) we record the frequency of occurrence $\#(c, m)$ and estimate the concept-mention probability $p(c|m)$ through

$$p(c|m) \approx \frac{\#(c, m)}{\sum_{c_i \in \mathbf{Wiki}} \#(c_i, m)}. \quad (1)$$

For instance, we obtain $p(Japan|\text{"Japan"}) \approx 0.97$. Note that these are not true probabilities, since due to parsing errors or too aggressive stemming, we may observe that $\sum_i p(c_i|m) \neq 1$.

Lucene ranks the search results for a query according to a product of the following factors: the term frequency of the term x in the document, its inverted document frequency *idf(x)*, a weight factor *boost(x)* and the document's length norm (Hatcher et al., 2010). For the final index creation, we use the above probabilities as *boosts* on the `meantBy` fields: the index concept for *Japan* has the field (`meantBy`,"Japan", 0.97), where the field's searchable content is the surface form "Japan" and the field's boost is the estimated probability value $p(c|m) = 0.97$. To keep memory consumption as low as possible, we create an auxiliary index to retrieve these values efficiently.

In the following we refer to the above fields as *name* fields. Name fields allow queries of the form (`title`, m), (`redirect`, m) or (`meantBy`, m). In our experiments, we will show results when additional context information is ignored and only name fields are used for disambiguation.

Context fields Assuming that each concept is thoroughly depicted in the article's main text, we use this context (except stop words) in a designated `context` field. This allows us to place queries of the form (`context`,"w"), where "w" may be the mention itself or any other key word extracted from the input document.

Type fields For all **Wiki** concepts that can be automatically aligned with YAGO (Suchanek et al., 2008), we add the type information extracted from YAGO, such as person, location, etc.. If the mention text has been tagged as a named entity by a NER, we can use this additional meta information to place a more distinctive query, for example a query (`type`,"person").

Both context and type fields can be queried separately, and we will show the influence of context and type usage in our experiments.

Link fields Relational information is an important factor for concept resolution and Wikipedia's link structure provides a straightforward resource to model relations among concepts. We store all outlinks $\{c \rightarrow c'\}$ of a Wikipedia concept in the fields (`linkText`,"m") of the respective index concept c, where "m" is the anchor text used for the outlink target concept c'. These fields are used to compute the relatedness among concepts (c.f. Eq. 3) but also queried in the collective search step of our disambiguation algorithm (Alg. 1.1-1.2).

3.2 Mention-specific Attributes

To create specific disambiguating attributes for each mention m_i, we first extract the mention's *name*, *type* and *context* attributes from the input document.

Name and type attributes Having collected all mentions from the input document, we keep the name (i.e. the surface form) and if present, the type information as attributes for each m_i. We then run a *mention expansion* that searches for mentions that are token-wise contained in previous mentions. If the type of two mentions is the same, the shorter mention is expanded to the longer one. For example, if $\mathbf{M}$ ={("Al Gore", per), ("Gore", per), ("Gore Bay", loc)}, the result of mention expansion is $\mathbf{M}$ ={("Al Gore", per), ("Al Gore", per), ("Gore Bay", loc)}. If the NER did not identify the type of "Gore", we still assume that it refers to the person "Al Gore", since the abbreviation of person names is much more common compared to the abbreviation

of location names. In our experiments, we found that the expansion of mention names has a positive impact on disambiguation performance.

Context attributes We use both local as well as document level context information. The local context is a $[2,2]$ noun-window around the mention without stop words. Additionally, we extract tf-idf ranked key words from the document text and keep the 20 words with highest tf-idf value as document key words. This set is then localized for each m_i: from the joint set of local context words and document key words, we keep only those words that appear at least once in the text of an index concept whose title matches m_i. In the same way, we compute key words from the headline of the input document, assuming that headline information is especially important.

Topic information Additionally to the pure word-based context information, we use an LDA topic model (Blei and Lafferty, 2009) to infer the most likely topic distribution of the input document. The LDA model was trained with $Z = 500$ topics on the **CoNLL** training corpus (c.f. Sec. 5) where words are the surface forms of the named entities appearing in the documents. We then apply this topic model to the input document giving local context words of mention m_i a five-fold weight. This yields a specific topic distribution $topic(m_i)$ for each mention m_i.

Name, type and context attributes of the input mentions can be matched to the according index fields using specific queries. Topic information is used for relatedness computation as well as a distinct feature for the Ranking SVM.

4 Disambiguation via Search and Ranking

Having defined the components of our search index and the input to our system, we explain the search process for Wikification in this section. The first part of our disambiguation procedure is to *jointly* treat all mentions $m_1, \ldots, m_k$ in the input document to generate a *bestFit* candidate for each m_i. The algorithm for this is depicted in Alg. 1

4.1 bestFit concepts from collective search using ensemble queries

Our assumption is that Wikipedia articles containing many of the input mentions are likely to be of a similar content as the input document. From the *outlink target concepts* these articles provide, we can automatically generate good disambiguation candidates concepts (**step 1**).

To retrieve these candidate concepts, we create an *ensemble query* that jointly treats the names of all mentions m_i and thus exploits the co-occurrence of mentions as link texts (see Alg. 1.1). This query then contains one query term (`linkText`, m_i) per mention m_i. Using this query, a search in **Wiki** then yields a ranked list of concepts $\mathbf{C}_{coll}$ that *collectively* contain the input mentions m_i as values in their `linkText` fields (Alg. 1.2). Lucene ranks each concept $c_{coll} \in \mathbf{C}_{coll}$ with a score s_L, based on the number of matches c_{coll} has on the fields (`linkText`, m_i). The higher the ranking of c_{coll}, the more mentions the concept c_{coll} contains as link text.
We keep the top 30 concepts in $\mathbf{C}_{coll}$ from which we extract the collection of outlink targets $\mathbf{C}_{\rightarrow}$. Next, we endow each outlink target concept $c \in \mathbf{C}_{\rightarrow}$ with a weight $w(c)$ that is the sum over the concepts' scores in which it appears as an outlink target, i.e. $c_{coll} \rightarrow c$ (Alg. 1.4):

$$\forall c \in \mathbf{C}_{\rightarrow}: \quad w(c) = \sum_{c_{coll} \in \mathbf{C}_{coll}} \delta_c s_L(c_{coll}), \quad \delta_c = \begin{cases} 1 & \text{iff } c_{coll} \rightarrow c, \\ 0 & \text{else.} \end{cases} \tag{2}$$

Since the collection $\mathbf{C}_{\rightarrow}$ may contain a huge number of concepts appearing only once as an outlink target, we keep only the top 100 candidate concepts in $\mathbf{C}_{\rightarrow}$, that have the highest weights $w(c)$.
Next, we need to relate the elements in the candidate concept set $\mathbf{C}_{\rightarrow}$ to the input mentions. More specifically, we analyze for each $c \in \mathbf{C}_{\rightarrow}$ if either the title or the redirect of c contains the text of mention m_i. If so, we add c to the candidate set $\mathbf{C}_i$ for mention m_i (Alg. 1.7 ff). Note that one c can then be contained in multiple candidate sets. The result of the collective search is the collection $\{\mathbf{C}_i\}_{i=1}^{k}$, where each $\mathbf{C}_i$ is a set of candidate concepts for mention m_i.

Our intuition is that concepts mentioned jointly in an input document should be related. To model the relatedness between Wikipedia concepts, we follow the approach of Milne and Witten (2008) who define the normalized Google distance (NGD) of two concepts c_i and c_j as

$$\mathrm{NGD}(c_i, c_j) = \frac{\log\Big(|\{c' \rightarrow c_i\} \cap \{c' \rightarrow c_j\}|\Big) - \log\Big(\max(|\{c' \rightarrow c\}|, |\{c' \rightarrow c_j\}|)\Big)}{\log(|\{c' \rightarrow \cdot\}|) - \log\Big(\min(|\{c' \rightarrow c_i\}|, |\{c' \rightarrow c_j\}|)\Big)}, \tag{3}$$

where $\{c' \rightarrow c_i\}$ is the collection of all concepts c' that link to c_i (i.e. the inlinks of c_i) and $|\{c' \rightarrow \cdot\}|$ is the total number of links in Wikipedia. In the case that the concepts c_i and c_j share no inlinks, i.e. $\{c' \rightarrow c_i\} \cap \{c' \rightarrow c_j\} = \emptyset$, we define $\mathrm{NGD}(c_i, c_j) = 0$.

Using the above NGD, we can measure the relatedness of two candidate concepts. To account for the collective fitness of a set of candidates, we introduce *cross coherence* which basically states how well a concept $c_{ij} \in \mathbf{C}_i$ *fits* to the other candidate concepts $\{\mathbf{C}_l\}_{l=1}^{k}$. More formally, we define the cross coherence of a candidate concept c_{ij} and a collection of concepts $\{\mathbf{C}_l\}_{l=1}^{k}$ as

$$\text{cross coherence}(c_{ij}, \{\mathbf{C}_l\}_{l=1}^{k}) = \frac{1}{k} \sum_{\substack{i'=1 \\ i' \neq i; \mathbf{C}_i \neq \mathbf{C}_{i'}}}^{k} \frac{1}{|\mathbf{C}_{i'}|} \sum_{\substack{c' \in \mathbf{C}_{i'} \\ c_{ij} \neq c'}} \Delta\mathrm{NGD}(c_{ij}, c'), \tag{4}$$

with k the number of mentions in the document, i the index of mention m_i and j the index over the candidate concepts for m_i. The second sum is the average NGD (Eq. 3) of c_{ij} to the concepts in another candidate set $\mathbf{C}_{i'}$ which is again averaged over all candidate sets by the first sum. Cross coherence can be interpreted as the average distance of a concept to a collection of concepts and has range $[0, 1]$, where 0 denotes a *completely unrelated* concept. We compute cross coherence in **step 2** (Alg. 1.11-1.13) to determine the relatedness of candidates extracted in the previous **step 1**.

The factor Δ in Eq. 4 serves as an additional relatedness weighting between two concepts. While both Milne and Witten (2008) and Ratinov et al. (2011) used the standard NGD with $\Delta = 1$, we analyze three additional weighting schemes. The scheme $\Delta_{\cos}\mathrm{NGD}$ weighs the NGD via the cosine distance $\cos(c_i, c_j)$ between the term vectors of two article texts. Additionally, we introduce $\Delta_{topics}\mathrm{NGD}$ that uses the thematical distance between two article link text collections. More specifically, we use a LDA topic model to infer the topic probability distribution over the words contained in a concept's outlink collection $\{c \rightarrow c'\}$ (for more details on the topic model, see 3.2). We define Δ_{topics} as the Hellinger distance between two concepts' outlink text topic probability distributions:

$$\Delta_{topics}(c_i, c_j) = 1 - \sum_{z=1}^{Z} \sqrt{topic_z(c_i) \cdot topic_z(c_j)}, \tag{5}$$

where $topic(c_i)$ and $topic(c_j)$ are the topic probability distribution vectors for the link texts of the concepts c_i and c_j and Z is the number of topics in the LDA model. The subtraction from 1 assures that $\Delta_{topics} = 0$ iff $topic(c_i) = topic(c_j)$ and is required to maintain the interpretation of cross coherence as a distance. The last relatedness measure we analyze is cosine distance without NGD.

In **step 3**, we compute the final result of the collective search procedure, i.e. the *bestFit* concepts. We define the *bestFit* candidate concept for each mention m_i by the product of the weight $w(c)$ (computed in **step 1**) and c's cross coherence value (computed in **step 2**):

$$bestFit_i = \arg\max_{c \in \mathbf{C}_i} (w(c) \cdot \text{cross coherence}(c)). \tag{6}$$

When the concept-mention association in **step 1(c)** yields no result, no *bestFit* candidate can be assigned. Note that if we used the triple of $w(c)$, cross coherence(c) and $p(c|m)$, high-prior candidates are likely to dominate, even if their coherence is low.

Algorithm 1: Collective search for *bestFit* candidate generation

Input: List of mentions $\mathbf{M} = \{m_1, \ldots, m_k\}$
Output: A $bestFit_i$ candidate for each $m_i \in \mathbf{M}$, i.e. $\{(m_1, bestFit_1), \ldots, (m_k, bestFit_k)\}$
1.1 $query = (\texttt{linkText}, name(m_1)) \wedge \ldots \wedge (\texttt{linkText}, name(m_k))$ // **step 1(a)**: create ensemble query using all m_i
1.2 $\mathbf{C}_{coll}$ = search **Wiki** using *query*
1.3 $\mathbf{C}_{\rightarrow} = \bigcup_{c_{coll} \in \mathbf{C}_{coll}} \{c_{coll} \rightarrow c'\}$ // collect outlink target concepts from collective search result $\mathbf{C}_{coll}$
1.4 **for** $c \in \mathbf{C}_{\rightarrow}$ **do** // **step 1(b)**: compute concept weights
1.5 compute concept weight according to Eq. 2
1.6 keep only top 100 link target concepts in $\mathbf{C}_{\rightarrow}$
1.7 **for** $m_i \in \mathbf{M}$ **do** // **step 1(c)**: relate concepts to mentions
1.8 initialize candidate set $\mathbf{C}_i = \emptyset$
1.9 **for** $c \in \mathbf{C}_{\rightarrow}$ **do**
1.10 add c to candidate set $\mathbf{C}_i$ if `title` or `redirect` of c contains mention text m_i
1.11 **for** $i = 1, \ldots, k$ **do** // **step 2**
1.12 **for** $c_{ij} \in \mathbf{C}_i$ **do**
1.13 compute cross coherence according to Eq. 4
1.14 **for** $m_i \in \mathbf{M}$ **do** // **step 3**
1.15 find *bestFit* concept according to Eq. 6
1.16 **return** $\{(m_1, bestFit_1), \ldots, (m_k, bestFit_k)\}$

4.2 Combining Search Results and Supervised Learning

The final disambiguation algorithm Alg. 2 has two steps (**step 4** and **5**). First, we run a search on **Wiki** to create ranked sets of candidate concepts $\mathbf{C}_1^*, \ldots, \mathbf{C}_k^*$ with one set $\mathbf{C}_i^* \subset$ **Wiki** per mention m_i. Second, a pre-trained SVM is applied to re-rank this output and detect nil-concepts. The result is the disambiguated list of input mentions, where each mention m_i is associated with a unique concept $\hat{c}_i \in$ **Wiki** $\cup \mathbf{C}_0$, i.e. $\{(m_1, \hat{c}_1), \ldots, (m_k, \hat{c}_k)\}$.

In the search part (**step 4**), we restrict the size of each $\mathbf{C}_i^*$ (i.e. the number of search results) to 5, which we experimentally found to be sufficient. Initially, we also require each concept $c_i^* \in \mathbf{C}_i^*$ to have at least 5 inlinks. This *inlink prior* aims at filtering out rarely referenced concepts. Then we run separate searches using only the `titleLong`, `title` and `redirect` fields of the index documents to find direct matches between mention m_i and concepts $c \in$ **Wiki** (Alg. 2.3). If such a match $\tilde{c}$ has been found, we give an additional query boost for the attributes of $\tilde{c}$, that is the title of $\tilde{c}$ is used as an additional query term with a five times higher weight than the other

query terms. For the *bestFit* concept we proceed analogously.
If either $\tilde{c}$ or $bestFit_i$ has a lower number of inlinks than initially assumed, the inlink prior is adapted automatically (Alg. 2.6). Alternatively, if the maximum returned score of the first search (Alg. 2.10) is less than a threshold $\tau = 1$, we re-run the search without the prior constraint (Alg. 2.12). After prioritisation on the results from direct and collective search, we add each mention's individual attributes to account for type and context information. In our experiments, we evaluate searches of different coverage, more specifically searches using

- name attributes, i.e. we add queries only on name fields (Alg. 2.7)
- name and type attributes, i.e. we extend the query using the mention's type (Alg. 2.8)
- name, type and context attributes, i.e. we additionally query context fields (Alg. 2.9).

Using this comprehensive query, the search result in **Wiki** is a either a set of ranked concepts $\mathbf{C}_i^*$ or an empty set, in which case the search did yield no result. We collect all concept sets $\mathbf{C}_i^*$ into an overall set $\{\mathbf{C}_i^*\}_{i=1}^k$ on which we apply a linear ranking SVM (**step 5**). Each concept c_i^* is represented by a vector of features that are computed both from the index ranking $s_L(c_i^*)$ as well as in relation to the input mention. We use the ranking in different feature representations: $s_{L,\log}(c_i^*) = \log s_L(c_i^*)$, $s_{L,norm}(c_i^*) = \frac{s_L(c_i^*)}{\sum_{c_i^{*'} \in \mathbf{C}_i^*} s_L(c_i^{*'})}$, $s_{L,rank}(c_i^*) = \frac{s_L(c_i^*)}{\arg\max\ s_L(c_i^* \in \mathbf{C}_i^*)}$. Additional features are the concept-mention probability $p(c_i^*|m_i)$, the cross coherence of c_i^* computed as in 4 but now in relation to the improved concept set $\{\mathbf{C}_i^*\}_{i=1}^k$, the Hellinger distance over the topic distributions $topic(m)$ and $topic(c_i^*)$. As proposed by Bunescu and Pasca (2006), we use a feature f_0 for nil-concepts c_0 that is required for the automatic detection of these nil-concepts.

We train the Ranking SVM on the **CoNLL train** corpus which is annotated with Wikipedia concepts as well as nil-concepts. Positive and negative examples are extracted in the same way as we generate disambiguation candidates. For instance, a positive example is the correct candidate c_i^* for a mention m_i and the negative examples are all other $c_i^* \in \mathbf{C}_i^*$ for that mention. Additionally, if not already present when the search did yield no result, we add a candidate c_0 for each mention whose only feature is f_0.

In the final **step 5** we use the trained SVM to re-rank the index output (Alg. 2.17). While the index search often provides a reliable candidate, implicit features such as coherence, concept-mention probability and topic similarity are only partially graspable by **Wiki** and may induce a SVM re-ranking.

5 Benchmark Corpora

Recent work published a variety of benchmark corpora for Wikification, most of them consisting of English newspaper articles from different time periods. Table 1 gives an overview of the corpora treated in this paper. The major difference between these corpora is the annotation scheme. Cucerzan, Ratinov et al., Milne and Witten and Kulkarni et al. treated mentions of all types on **MSNBC**, **ACE**, **AQUAINT**[3] and **IITB**[4] respectively. Hoffart et al. considered only named entity mentions in the **CoNLL** corpus[5]. Additional to differing mention types, there are also annotation differences that render comparison difficult. For instance, in **CoNLLb** the mention "Taiwan" is linked to *Republic of China*, while in **ACE** it is linked to *Taiwan*. We also observed

[3] **MSNBC**, **AQUAINT** and **ACE** are publicly available and described in detail in (Ratinov et al., 2011).
[4] **IITB** is publicly available and described in detail in (Kulkarni et al., 2009).
[5] **CoNLL** is publicly available and described in detail in (Hoffart et al., 2011b), we consider **CoNLL testb** called **CoNLLb** in the following.

Algorithm 2: Disambiguation algorithm

Input: List of mentions $\mathbf{M} = m_1, \ldots, m_k$, where each m_i has name, type, context & *bestFit* attributes.
Output: List of disambiguated mentions $\mathbf{M} = \{(m_1, \hat{c}_1), \ldots, (m_k, \hat{c}_k)\}$

2.1 **for** $m_i \in \mathbf{M}$ **do** // **step 4**
2.2 $p_{in} = 5$ // initialize inlink prior
2.3 $\tilde{c} = \text{directMatch}(m_i)$ // c.f. Sec. 4.2
2.4 **if** $\tilde{c} \neq \emptyset$ **then** add boosted query terms for attributes of $\tilde{c}$
2.5 **if** $bestFit_i \neq \emptyset$ **then** add boosted query terms for attributes of $bestFit_i$
2.6 $p_{in} = \min\left(p_{in}(\tilde{c}), p_{in}(bestFit_i), p_{in}\right)$ // reduce prior on inlinks
2.7 *query*.addNameQuery(name(m_i)) // add name attributes to the query terms
2.8 **if** *type* **then** *query*.addTypeQuery(type(m_i)) // add type attributes to the query terms
2.9 **if** *context* **then** *query*.addContextQuery(context(m_i)) // add context attributes to the query terms
2.10 $\mathbf{C}_i^* =$ search **Wiki** using *query* and inlink prior p_{in}
2.11 **if** $\max_{c_j^* \in \mathbf{C}_i^*} s_L(c_j^*) \leq \tau$ **then**
2.12 $\mathbf{C}_i^* =$ search **Wiki** using *query* without inlink prior
2.13 **if** $\mathbf{C}_i^* \neq \emptyset$ **then** $\{\mathbf{C}_i^*\} \cup \mathbf{C}_i^*$ **else** $\{\mathbf{C}_i^*\} \cup c_0$ // add $\mathbf{C}_i^*$ to concept set $\{\mathbf{C}_i^*\}$ or add c_0 if the search yields no results
2.14 **for** $i = 1, \ldots, k$ **do**
2.15 **for** $c_{ij}^* \in \mathbf{C}_i^*$ **do**
2.16 compute cross coherence$(c_{ij}^*, \{\mathbf{C}_l^*\}_{l=1}^k)$ according to Eq. 4 and set other features (c.f. Sec. 4.2)
2.17 $\hat{c}_i = \arg\max_{c_{ij}^* \in \mathbf{C}_i^*} \text{SVM rank}(c_{ij}^*)$ // **step 5**: rank candidates by trained SVM for final concept prediction
2.18 **return** $\mathbf{M} = \{(m_1, \hat{c}_1), \ldots, (m_k, \hat{c}_k)\}$

corpus	#documents	#Wikipedia concepts (unique)	$\#c_0$	$\#c \in$ **Wiki**	$\#\tilde{c}_0 \notin$ **Wiki**	$\varnothing\lvert\mathbf{M}\rvert$ per doc.
MSNBC	20	658 (279)	97	640	18	37.75
ACE	36	257 (185)	49	254	3	8.5
AQUAINT	50	727 (572)	0	702	25	14.54
CoNLLb	228	4363 (1527)	0	4317	46	19.13
IITB	104	11185 (3755)	0	9439	1746	107.54

Table 1: Benchmark corpora with number of documents, the number of (unique) Wikipedia concepts and nil-concepts c_0, the number of linkable concepts $c \in$ **Wiki**, the number of Wikipedia concepts $\tilde{c}_0$ missing in **Wiki** and the average number of mentions per document.

some inconsistencies in the **CoNLL** training corpus that are presumably due to inter-annotator disagreement (20%) or candidate selection: while the phrase "European Union" is linked to the appropriate Wikipedia concept, it's acronym "EU" is linked to c_0. While Hoffart et al. neglected nil-concepts for evaluation on **CoNLLb**, these inconsistencies might be harmful for the SVM training of our approach. Moreover, **CoNLLb** contains many news articles about sport events. These are often not truly natural language texts, but more table-like. These variations make it challenging to apply the same system to different corpora.

For all corpora we proceed as follows: given the input mention, we first check if the ground truth concept is linkable, i.e. contained in our index. If this is not the case, but the mention is linked to some $c \neq c_0$, we change the ground truth to $\tilde{c}_0$ which is always considered during evaluation. Since we also resolve redirects, the number of distinct concepts in Tab. 1 may differ from the one published in the respective paper. Note that the overall number of mentions remains unchanged. The procedure is the same for concepts that do no longer exist in Wikipedia.
For a consistent set of named entity tags, we run the Apache OpenNLP NER[6] on all corpora.

[6] `http://opennlp.apache.org/`

6 Performance Measures for Wikification

In the following, we discuss different Wikification evaluation techniques. While in many areas performance measures are defined by the task at hand and used thoroughly by most authors, this is not the case in the field of concept disambiguation or Wikification. Consequently, published results are often hard to comparable.

Following Milne and Witten (2008), Ratinov et al. used **Bag-of-Titles** (**BOT**) evaluation which compares the predicted set of titles (i.e. concepts) with the ground truth set of concepts, ignoring duplicates in either set, and further utilizes standard Precision, Recall, and F1. For discussion, we take the example from Ratinov et al. (2011). Let the ground truth be $truth =$ {("China", *People's Rep. of China*), ("Taiwan", *Taiwan*), ("Jiangsu", *Jiangsu*}, with $truth_{BOT} =$ {*People's Rep. of China*, *Taiwan*, *Jiangsu*}. Assume the system predicts {("China", *People's Rep. of China*), ("China", *History of China*), ("Taiwan", c_0), ("Jiangsu", *Jiangsu*)}, with associated BOT $pred_{BOT} =$ {*People's Rep. of China*, *History of China*, *Jiangsu*}. According to Ratinov et al., both Precision and Recall for $pred_{BOT}$ are 0.66. Consequently, the nil prediction c_0 for *Taiwan* is not counted as a false positive, since we already observe *History of China* as a false positive, with two true positives from *People's Rep. of China* and *Jiangsu* resulting in $P = 0.66$.

The first remarkable point is the ignorance of duplicate concepts which obscures both erroneous as well as correct predictions: if a concept appears 5 times in the ground truth annotation, and the disambiguation model fails to resolve it correctly, the number of false negatives is only 1 in BOT, whereas it would be 5 if all instances were considered. Analogously this holds for the number of true positives. Second, nil predictions are not counted as false positives, which renders Precision less comparable. In our implementation of BOT, we assume that the sequential input order is taken into account.

The performance measure used by Hoffart et al. (2011b) is **Mean Average Precision** (**MAP**) which is defined as $MAP = \frac{1}{m}\sum_{i=1}^{m} p@\frac{i}{m}$, where $p@\frac{i}{m}$ is the Precision at a specific Recall level. Here, the model output is ranked according to the model's confidence s, i.e. mention-concept pairs with high model confidence are ranked at leading positions, pairs with low confidence at late positions. Consider the following prediction $\{s(m_3, c_3) = 0.9, s(m_2, c_2) = 0.8, s(m_1, c_1) = 0.2\}$, that is sorted by some confidence s instead of order of appearance. If c_1 is an incorrect prediction, the associated Precision values are $\{p@1 = \frac{1}{1}, p@2 = \frac{2}{2}, p@3 = \frac{2}{3}\}$. According to the above definition, the MAP of this example is $\frac{1+1+2/3}{3} = \frac{8}{9}$. If in contrast, we followed the sequential input order, the MAP would be $\frac{0+1/2+2/3}{3} = \frac{7}{9}$. Note that the interpretation of Recall differs from that in BOT since it is related to the position in the output list and not the number of false negatives. In terms of BOT, the performance result for this example is $P = R = \frac{2}{3}$.

Assuming that incorrect predictions have in general a low confidence, MAP shuffles erroneous predictions to the end of the ranked output list. Then the sum is dominated by correct predictions (high confidence) at the top of the ranking, which are propagated through the whole list. This is of great importance, if the number of mentions in a document is especially large. In our implementation of MAP, the model confidence is represented by the SVM's prediction, i.e. the instance's hyperplane offset.

The most crucial difference between current systems is the treatment of covered and uncovered concepts. Hoffart et al. decided to ignore nil-concepts during evaluation and hence roughly 20% of the mentions. To compare our method with AIDA, we follow this restriction when applying our method on **CoNLLb** and ignore nil-concepts for this corpus as well. Using the

search coverage	no *bestFit*	*bestFit* via NGD	*bestFit* via Δ_{topics}NGD	*bestFit* via $\Delta_{\cos}$NGD	*bestFit* via $\cos(c_i, c_j)$
mention (exp.)	87.69/94.55	86.83/94.70	88.12/95.58	88.96/95.94	86.73/94.44
+type	86.10/94.32	88.79/95.60	88.22/95.96	89.53/95.98	88.46/95.73
+context	86.43/94.30	89.50/96.10	89.30/96.60	89.95/96.54	89.20/96.45
+topics	87.59/95.16	89.47/96.46	89.50/96.48	**89.95**/96.81	89.60/96.67
⌀ cross coherence		0.381	0.179	0.104	0.215

Table 2: $F1_{BOT}$/MAP of our system on **MSNBC** for different configurations (all values in %).

search coverage	no *bestFit*	*bestFit* via NGD	*bestFit* via Δ_{topics}NGD	*bestFit* via $\Delta_{\cos}$NGD	*bestFit* via $\cos(c_i, c_j)$
mention (exp.)	84.46/92.74	86.18/93.23	87.91/94.02	87.02/93.01	86.70/92.95
+type	83.30/91.62	87.23/93.43	87.75/93.51	87.18/93.21	87.23/93.10
+context	86.49/93.28	86.76/92.98	88.40/93.80	88.85/94.12	87.75/93.54
+topics	86.50/91.16	86.97/91.25	88.44/94.67	**89.01**/94.33	88.24/93.68
⌀ cross coherence		0.354	0.139	0.096	0.211

Table 3: $F1_{BOT}$/MAP of our system on **ACE** for different configurations (all values in %).

AIDA online version that treats nil-concepts, we can evaluate this system on the other corpora. Kulkarni et al. also used a different evaluation scheme (KUL_{F1}) that is comparable to BOT but takes incorrect nil predictions into account. For more details, we refer to the respective paper.

7 Evaluation

As most alternative approaches rely on different versions of large-scale knowledge bases, it is practically not feasible to re-implement every competitor system. GLOW is publicly available, but we decided against using it, since we could not reproduce the results published in (Ratinov et al., 2011) and assumed that there was a crucial difference we could not solve[7]. Hence we compare our system to the figures reported by Ratinov et al. (2011) both for GLOW as well as the M&W system (Milne and Witten, 2008). For comparison with AIDA, we used the online interface $AIDA_{web}$ which was kindly provided to us by the authors[8] and run on all corpora. To give a unified view, we report the results for our system in all performance measures outlined in the previous section.

We ran initial experiments on all corpora to evaluate the effect of the mention expansion described in Sec. 3.2 and found that it increased $F1_{BOT}$ on all corpora by about 2%, when only the mention's name was considered in the search. We report the effect of different *search coverages* and show that in many cases results can be improved when we extend searches relying only the expanded mention name by additional type and context information. For all coverages, we show the effect of *bestFit* configurations: weighting NGD with Δ_{topics}, $\Delta_{\cos}$ and replacing NGD by the cosine distance over article texts. The influence of the topic feature computed from the Hellinger distance (Eq. 5) of $(topics(m_i), topics(c))$ is reported as well since it is computationally the most expensive SVM feature.

Tables 2 to 6 show the results obtained for the different configurations of our system. The corresponding performance figures of of GLOW, M&W and $AIDA_{web}$ are given in the text. For **MSNBC** (Tab. 2), the best configuration of our system (complete coverage, topics, *bestFit* candidate via $\Delta_{\cos}NGD$) achieves a $F1_{BOT}$ of 89.95%, which is 15% higher than that of GLOW (74.88%) and 20% higher than for M&W (68.49%). Also, the MAP of our system is with 96.81% more than 25% higher than that of $AIDA_{web}$ (69.52%). We found that the same configuration

[7]Thanks to Lev-Arie Ratinov for his useful comments on this.

[8]We use the most current version of July 30th, 2012.

search coverage	no *bestFit*	*bestFit* via NGD	*bestFit* via Δ_{topics}NGD	*bestFit* via $\Delta_{\cos}$NGD	*bestFit* via $\cos(c_i, c_j)$
mention (exp.)	84.77/94.50	84.71/94.83	85.07/94.47	85.45/94.65	84.53/94.65
+type	84.41/94.69	84.93/94.87	85.61/95.02	85.43/94.73	84.41/94.57
+context	84.81/92.92	84.50/93.83	84.19/94.10	84.59/93.70	82.95/93.36
+topics	**86.81**/91.97	84.46/91.97	84.33/93.87	84.94/93.53	83.20/93.55
⌀ cross coherence		0.31	0.119	0.07	0.161

Table 4: $F1_{BOT}$/MAP of our system on **AQUAINT** for different configurations (all values in %).

search coverage	no *bestFit*	*bestFit* via NGD	*bestFit* via Δ_{topics}NGD	*bestFit* via $\Delta_{\cos}$NGD	*bestFit* via $\cos(c_i, c_j)$
mention (exp.)	84.89	85.03	85.71	85.75	85.12
+type	85.36	86.72	88.13	87.26	87.44
+context	86.04	88.23	89.25	88.70	88.80
+topics	87.56	88.65	**89.32**	89.13	89.12
⌀ cross coherence		0.402	0.208	0.134	0.262

Table 5: MAP of our system on **CoNLLb** for different configurations (all values in %).

also yields the best result on **ACE** (Tab. 3). On this corpus, our system achieves a $F1_{BOT}$ of 89.01%, which outperforms GLOW (77.25%) and M&W (72.67%) by more than 12%. Also, the MAP of our system is with 94.33% about 9% higher than that of $AIDA_{web}$ (86.14%).
For **AQUAINT** (Tab. 4), the best configuration of our system is complete search coverage and using the topic feature in SVM ranking. Here, *bestFit* candidate generation did not increase performance. We argue that this is due to the rather low average cross coherence over the ground truth concepts. Without the usage of collective information, our system achieves a $F1_{BOT}$ of 86.81%, which outperforms GLOW (83.94%) and M&W (83.61%) by 3%. Note that even the slightly worse results using collective search are higher. Also, the MAP of our system is with 91.97% about 30% higher than that of $AIDA_{web}$ (58.61%). For all of the above corpora, we found that not using the SVM for candidate re-ranking and nil-concept detection reduces the $F1_{BOT}$ of our system between 5 and 10%, which shows the usefulness of a supervised classifier.
For **CoNLLb** (Tab. 5), the best configuration of our system is the complete search coverage, the topic feature in SVM ranking, and *bestFit* candidate generation via Δ_{topics}NGD. This corpus has the highest avg. cross coherence over the ground truth concepts. Our system achieves a MAP of 89.32% (with corresponding $F1_{BOT}$= 82.16%), which is only slightly better than the figures published for AIDA (89.05%) but about 4% higher than that of $AIDA_{web}$ (85.66%). Without SVM application, the MAP of our system would be reduced to 86.70%. This indicates the necessity of features that are not graspable by **Wiki** but available in SVM candidate re-ranking.
We found that on **IITB** (Tab. 6), the best results can be achieved when we use only name and type based queries in combination with *bestFit* candidate selection via Δ_{topics}NGD. Although we found that this corpus has the lowest avg. cross coherence over the ground truth concepts, the collective search increases performance. We argue that this result is due to the very high number of mentions per document, which has a diminishing effect on the avg. cross coherence. Our system achieves a KUL_{F1} of 75.26%, which is 5% better than the result published by Kulkarni et al. (2009) (69.69%). Note that the performance of $AIDA_{web}$ on **IITB** is only MAP=43.62%, whereas the corresponding MAP of our system is 90%. We are aware that the performance reported by Han et al. (2011) is with KUL_{F1}=78.95% about 4% higher than that of our system. Still, even though our system was not tuned on specific data sets, we achieve a high performance on all of the 5 different benchmark corpora. We argue that this makes our system the most stable compared to other approaches both in terms of generalizability and applicability.

search coverage	no *bestFit*	*bestFit* via NGD	*bestFit* via Δ_{topics}NGD	*bestFit* via $\Delta_{\cos}$NGD	*bestFit* via $\cos(c_i, c_j)$
mention (exp.)	73.81/89.92	74.74/89.91	**75.26/89.95**	74.68/89.67	73.89/89.32
+type	73.96/89.93	74.90/89.94	**75.10/90.00**	74.85/89.68	74.08/89.40
+context	72.57/88.01	69.07/85.69	69.81/85.59	68.54/86.23	69.29/86.14
+topics	71.10/87.26	68.74/85.41	69.41/86.31	68.35/86.10	69.13/85.83
⌀ cross coherence		0.224	0.087	0.041	0.112

Table 6: KUL_{F1}/MAP on **IITB** for different method configurations (all values in %).

To summarize, we observe for all corpora a positive correlation between the avg. cross coherence of ground truth concepts and the effect of the collective search. The influence of confidence sorting in MAP becomes obvious on **MSNBC** and **ACE**: while F1_{BOT} differs only by 1%, the associated MAP value can differ by 4%. This can be the case, when the average number of mentions per document differs (8.5 on **ACE** and 37.75 on **MSNBC**).

The concept-mention probability $p(c|m)$ is a very strong feature as often the name is sufficient for disambiguation. This is most obvious on **IITB**, where the incorporation of context features even decreased performance. We also found that this prior-like attribute can mislead the SVM re-ranking on **CoNLLb**. This corpus contains many sport statistics that, for instance, mention countries participating in a match. As an example, even though the *bestFit Japan National Football Team* is correct due to high cross coherence, the SVM re-ranked the output to *Japan*, since the concept-mention probability $p(\textit{Japan}|\text{"Japan"}) = 0.97$ is much higher than $p(\textit{Japan National Football Team}|\text{"Japan"}) = 0.0063$. While Hoffart et al. (2011b) thus did not always use this feature, we could not find an appropriate threshold for our system.

Especially for **CoNLLb** we found that our system suffered from annotation scheme differences. While our system links mentions like "British" to concepts such as *English language* or *British people*, the ground truth concept in **CoNLLb** is always *United Kingdom*. An investigation showed that these annotations are often correct but in general depend on the gusto of the annotator. Also, we observed for **IITB** that many ground truth concepts are disambiguation pages. These are not contained in our index and thus treated as $\tilde{c}_0$. For example, we observed a document on sports mentioning the word "fitness" which was linked to the disambiguation page *Fitness* by the **IITB** annotators. While our system predicted the suitable concept *Physical Fitness*, this was still treated as an error since we relinked the disambiguation page *Fitness* to $\tilde{c}_0$.

Conclusion

In this paper we described a novel algorithm for concept disambiguation through concept assignment to Wikipedia articles. We exploit the coherence of Wikipedia concepts and take into account a variety of features to perform the assignment. The algorithm also estimates if a concept is not covered by Wikipedia. It turned out that the collective search is more efficient, if the average coherence between the concepts is higher.

We analyzed different evaluation criteria and discussed their relative strengths and weaknesses. We evaluated various configurations of our approach on five benchmark corpora and compared the results to four competitor systems. For some benchmark data sets our system is dramatically better than the other approaches, while for other corpora the differences are not so pronounced. We observed that these benchmark corpora are not error free, which can limit their usability. Except for one case our system always has better performance figures than the competitor systems and therefore can be considered as a stable alternative ready for practical application. In future work we will consider the assignment of concepts not described in Wikipedia.

References

Blei, D. and Lafferty, J. (2009). Topic models. In Srivastava, A. and Saham, M., editors, *Text Mining: Theory and Applications*. Taylor and Francis.

Bunescu, R. C. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*, pages 9–16.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *Proc. 2007 Joint Conference on EMNLP and CoNLL*, pages 708–716.

Han, X., Sun, L., and Zhao, J. (2011). Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR '11)*, pages 765–774. ACM.

Hatcher, E., Gospodnetic, O., and McCandless, M. (2010). *Lucene in Action*. Manning, 2nd revised edition edition.

Hoffart, J., Suchanek, F. M., Berberich, K., Kelham, E. L., de Melo, G., and Weikum, G. (2011a). Yago2: Exploring and querying world knowledge in time, space, context, and many languages. In *Demo paper in the proceedings of the 20th International World Wide Web Conference (WWW 2011)*.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011b). Robust disambiguation of named entities in text. In *Conference on Empirical Methods in Natural Language Processing*, pages 782–792.

Kulkarni, S., Singh, A., Ramakrishnan, G., and Chakrabarti, S. (2009). Collective annotation of wikipedia entities in web text. In *15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Milne, D. N. and Witten, I. H. (2008). Learning to link with wikipedia. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM'2008)*, pages 509–518.

Pilz, A. and Paass, G. (2011). From names to entities using thematic context distance. In *Proceedings of 20th ACM Conference on Information and Knowledge Management (CIKM)*, Glasgow, UK.

Ratinov, L.-A., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In *ACL*, pages 1375–1384.

Schuetze, H. (1998). Automatic word sense discrimination. *Computational Linguistics - Special issue on word sense disambiguation*, 24:97–123.

Song, D. and Heflin, J. (2011). Automatically generating data linkages using a domain-independent candidate selection approach. In *Proceedings of International Semantic Web Conference (ISWC)*, pages 649–664.

Suchanek, F., Kasneci, G., and Weikum, G. (2008). Yago - a large ontology from wikipedia and wordnet. *Elsevier Journal of Web Semantics*, 6(3):203–217.