Robust Disambiguation of Named Entities in Text

Johannes Hoffart[1], **Mohamed Amir Yosef**[1], **Ilaria Bordino**[2], **Hagen Fürstenau**[3], **Manfred Pinkal**[3], **Marc Spaniol**[1], **Bilyana Taneva**[1], **Stefan Thater**[3], **Gerhard Weikum**[1]
1 Max Planck Institute for Informatics, Saarbrücken, Germany
2 Yahoo! Research Lab, Barcelona, Spain
3 Saarland University, Saarbrücken, Germany
{jhoffart,mamir,mspaniol,btaneva,weikum}@mpi-inf.mpg.de
bordino@yahoo-inc.com {hagenf,pinkal,stth}@coli.uni-sb.de

Abstract

Disambiguating named entities in natural-language text maps mentions of ambiguous names onto canonical entities like people or places, registered in a knowledge base such as DBpedia or YAGO. This paper presents a robust method for collective disambiguation, by harnessing context from knowledge bases and using a new form of coherence graph. It unifies prior approaches into a comprehensive framework that combines three measures: the prior probability of an entity being mentioned, the similarity between the contexts of a mention and a candidate entity, as well as the coherence among candidate entities for all mentions together. The method builds a weighted graph of mentions and candidate entities, and computes a dense subgraph that approximates the best joint mention-entity mapping. Experiments show that the new method significantly outperforms prior methods in terms of accuracy, with robust behavior across a variety of inputs.

1 Introduction

1.1 Motivation

Web pages, news articles, blog postings, and other Internet data contain mentions of named entities such as people, places, organizations, etc. Names are often ambiguous: the same name can have many different meanings. For example, given a text like "They performed Kashmir, written by Page and Plant. Page played unusual chords on his Gibson.", how can we tell that "Kashmir" denotes a song by Led Zeppelin and not the Himalaya region (and that Page refers to guitarist Jimmy Page and not to Google founder Larry Page, and that Gibson is a guitar model rather than the actor Mel Gibson)?

Establishing these mappings between the mentions and the actual entities is the problem of *named-entity disambiguation (NED)*.

If the possible meanings of a name are known upfront - e.g., by using comprehensive gazetteers such as GeoNames (`www.geonames.org`) or knowledge bases such as DBpedia (Auer07), Freebase (`www.freebase.com`), or YAGO (Suchanek07), which have harvested Wikipedia redirects and disambiguation pages - then the simplest heuristics for name resolution is to choose the most prominent entity for a given name. This could be the entity with the longest Wikipedia article or the largest number of incoming links in Wikipedia; or the place with the most inhabitants (for cities) or largest area, etc. Alternatively, one could choose the entity that uses the mention most frequently as a hyperlink anchor text. For the example sentence given above, all these techniques would incorrectly map the mention *"Kashmir"* to the Himalaya region. We refer to this suite of methods as a *popularity-based (mention-entity) prior*.

Key to improving the above approaches is to consider the *context* of the mention to be mapped, and compare it - by some *similarity measure* - to contextual information about the potential target entities. For the example sentence, the mention "Kashmir" has context words like "performed" and "chords" so that we can compare a bag-of-words model against characteristic words in the Wikipedia articles of the different candidate entities (by measures such as cosine similarity, weighted Jaccard distance, KL divergence, etc.). The candidate entity with the highest similarity is chosen. Alternatively, labeled training data can be harnessed to learn a multi-way classifier, and additional features like entire phrases, part-of-speech tags, dependency-parsing paths, or nearby

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 782–792,
Edinburgh, Scotland, UK, July 27–31, 2011.

hyperlinks can be leveraged as well. These methods work well for sufficiently long and relatively clean input texts such as predicting the link target of a Wikipedia anchor text (Milne08). However, for short or more demanding inputs like news, blogs, or arbitrary Web pages, relying solely on context similarity cannot achieve near-human quality. Similarity measures based on syntactically-informed distributional models require minimal context only. They have been developed for common nouns and verbs (Thater10), but not applied to named entities.

The key to further improvements is to jointly consider multiple mentions in an input and aim for a *collective assignment* onto entities (Kulkarni09). This approach should consider the *coherence* of the resulting entities, in the sense of semantic relatedness, and it should combine such measures with the context similarity scores of each mention-entity pair. In our example, one should treat "Page", "Plant" and "Gibson" also as named-entity mentions and aim to disambiguate them together with "Kashmir".

Collective disambiguation works very well when a text contains mentions of a sufficiently large number of entities within a thematically homogeneous context. If the text is very short or is about multiple, unrelated or weakly related topics, collective mapping tends to produce errors by directing some mentions towards entities that fit into a single coherent topic but do not capture the given text. For example, a text about a football game between "Manchester" and "Barcelona" that takes place in "Madrid" may end up mapping either all three of these mentions onto football clubs (i.e., Manchester United, FC Barcelona, Real Madrid) or all three of them onto cities. The conclusion here is that none of the prior methods for named-entity disambiguation is robust enough to cope with such difficult inputs.

1.2 Contribution

Our approach leverages recently developed knowledge bases like YAGO as an entity catalog and a rich source of entity types and semantic relationships among entities. These are factored into new measures for the similarity and coherence parts of collectively disambiguating all mentions in an input text. For similarity, we also explore an approach that leverages co-occurrence information obtained from large, syntactically parsed corpora (Thater10).

We cast the joint mapping into the following graph problem: mentions from the input text and candidate entities define the node set, and we consider weighted edges between mentions and entities, capturing context similarities, and weighted edges among entities, capturing coherence. The goal on this combined graph is to identify a dense subgraph that contains exactly one mention-entity edge for each mention, yielding the most likely disambiguation. Such graph problems are NP-hard, as they generalize the well-studied Steiner-tree problem. We develop a greedy algorithm that provides high-quality approximations, and is customized to the properties of our mention-entity graph model.

In addition to improving the above assets for the overall disambiguation task, our approach gains in robustness by using components selectively in a self-adapting manner. To this end, we have devised the following multi-stage procedure.

- For each mention, we compute popularity priors and context similarities for all entity candidates as input for our tests.
- We use a threshold test on the prior to decide whether popularity should be used (for mentions with a very high prior) or disregarded (for mentions with several reasonable candidates).
- When both the entity priors and the context similarities are reasonably similar in distribution for all the entity candidates, we keep the best candidate and remove all others, fixing this mention *before* running the coherence graph algorithm.

We then run the coherence graph algorithm on all the mentions and their remaining entity candidates. This way, we restrict the coherence graph algorithm to the critical mentions, in situations where the goal of coherence may be misleading or would entail high risk of degradation.

The paper makes the following novel contributions: 1) a framework for combining popularity priors, similarity measures, and coherence into a robust disambiguation method; 2) new measures for defining mention-entity similarity; 3) a new algorithm for computing dense subgraphs in a mention-entity graph, which produces high-quality mention-entity mappings; 4) an empirical evaluation on a demanding corpus (based on additional annotations for the dataset of the CoNLL 2003 NER task), with signifi-

cant improvements over state-of-the-art opponents.

2 State of the Art

Recognizing named entities (NER tagging) in natural-language text has been extensively addressed in NLP research. The output is labeled noun phrases. However, these are not yet canonical entities, explicitly and uniquely denoted in a knowledge repository. Approaches that use Wikipedia for explicit disambiguation date back to (Bunescu06) and have been further pursued by (Cucerzan07; Han09; Milne08; Nguyen08; Mihalcea07). (Bunescu06) defined a similarity measure that compared the context of a mention to the Wikipedia categories of an entity candidate. (Cucerzan07; Milne08; Nguyen08) extended this framework by using richer features for the similarity comparison. (Milne08) additionally introduced a supervised classifier for mapping mentions to entities, with learned feature weights rather than using the similarity function directly. (Milne08) introduced a notion of semantic relatedness between a mention's candidate entities and the unambiguous mentions in the textual context. The relatedness values are derived from the overlap of incoming links in Wikipedia articles. (Han09) considered another feature: the relatedness of common noun phrases in a mention's context, matched against Wikipedia article names. While these features point towards semantic coherence, the approaches are still limited to mapping each mention separately. Nonetheless, this line of feature-rich similarity-driven methods achieved very good results in experiments, especially for the task of predicting Wikipedia link targets for a given href anchor text. On broader input classes such as news articles (called "wikification in the wild" in (Milne08)), the precision was reported to be about 75 percent.

The first work with an explicit collective-learning model for joint mapping of all mentions has been (Kulkarni09). This method starts with a supervised learner for a similarity prior, and models the pairwise coherence of entity candidates for two different mentions as a probabilistic factor graph with all pairs as factors. The MAP (maximum a posteriori) estimator for the joint probability distribution of all mappings is shown to be an NP-hard optimization problem, so that (Kulkarni09) resorts to approximations and heuristics like relaxing an integer linear program (ILP) into an LP with subsequent rounding or hill-climbing techniques. The experiments in (Kulkarni09) show that this method is superior to the best prior approaches, most notably (Milne08). However, even approximate solving of the optimization model has high computational costs.

Coreference resolution is the task of mapping mentions like pronouns or short phrases to a preceding, more explicit, mention. Recently, interest has arisen in cross-document coreference resolution (Mayfield09), which comes closer to NED, but does not aim at mapping names onto entities in a knowledge base. Word sense disambiguation (McCarthy09; Navigli09) is the more general task of mapping content words to a predefined inventory of word senses. While the NED problem is similar, it faces the challenges that the ambiguity of entity names tends to be much higher (e.g., mentions of common lastnames or firstname-only).

Projects on automatically building knowledge bases (Doan08) from natural-language text include KnowItAll (Banko07), YAGO and its tool SOFIE (Suchanek09; Nakashole11), StatSnowball (Zhu09), ReadTheWeb (Carlson10), and the factor-graph work by (Wick09). Only SOFIE maps names onto canonical entities; the other projects produce output with ambiguous names. SOFIE folds the NED into its MaxSat-based reasoning for fact extraction. This approach is computationally expensive and not intended for online disambiguation of entire texts.

3 Framework

Mentions and Ambiguity: We consider an input text (Web page, news article, blog posting, etc.) with mentions (i.e., surface forms) of named entities (people, music bands, songs, universities, etc.) and aim to map them to their proper entries in a knowledge base, thus giving a disambiguated meaning to entity mentions in the text. We first identify noun phrases that potentially denote named entities. We use the Stanford NER Tagger (Finkel05) to discover these and segment the text accordingly.

Entity Candidates: For possible entities (with unique canonical names) that a mention could denote, we harness existing knowledge bases like DBpedia or YAGO. For each entity they provide a set of short names (e.g., "Apple" for `Apple Inc.` and para-

phrases (e.g., "Big Apple" for `New York City`). In YAGO, these are available by the `means` relation, which in turn is harvested from Wikipedia disambiguation pages, redirects, and links.

Popularity Prior for Entities: Prominence or popularity of entities can be seen as a probabilistic prior for mapping a name to an entity. The most common way of estimating this are the Wikipedia-based frequencies of particular names in link anchor texts referring to specific entities, or number of inlinks.

Context Similarity of Mentions and Entities: The key for mapping mentions onto entities are the contexts on both sides of the mapping. We consider two different approaches. First, for each mention, we construct a context from all words in the entire input text. This way, we can represent a mention as a set of (weighted) words or phrases that it co-occurs with. Second, we alternatively consider similarity scores based on syntactically-parsed contexts, based on (Thater10). On the entity side of the mapping, we associate each entity with characteristic keyphrases or salient words, precomputed from Wikipedia articles and similar sources. For example, `Larry Page` would have keyphrases like "Stanford", "search engine", etc., whereas `Jimmy Page` may have keyphrases "Gibson guitar", "hard rock", etc. Now we can define and compute *similarity* measures between a mention and an entity candidate, e.g., the weighted word overlap, the KL divergence, n-gram-based measures, etc. In addition, we may use syntactic contextualization techniques, based on dependency trees, that suggest phrases that are typically used with the same verb that appears with the mention in the input text (Thater10).

Coherence among Entities: On the entity side, each entity has a context in the underlying knowledge base(s): other entities that are connected via semantic relationships (e.g., `memberOf`) or have the same semantic type (e.g., `rock musician`). An asset that knowledge bases like DBpedia and YAGO provide us with is the same-as cross-referencing to Wikipedia. This way, we can quantify the coherence between two entities by the number of incoming links that their Wikipedia articles share. When we consider candidate entities for different mentions, we can now define and compute a notion of *coherence* among the corresponding entities, e.g., by the overlap among their related entities or some form of type distance. Coherence is a key asset because most texts deal with a single or a few semantically related topics such as rock music or Internet technology or global warming, but not everything together.

Overall Objective Function: To aim for the best disambiguation mappings, our framework combines prior, similarity, and coherence measures into a combined objective function: for each mention m_i, $i = 1..k$, select entity candidates e_{j_i}, one per mention, such that

$$\alpha \cdot \sum_{i=1..k} \text{prior}(m_i, e_{j_i}) +$$
$$\beta \cdot \sum_{i=1..k} \text{sim}(\text{cxt}(m_i), \text{cxt}(e_{j_i})) +$$
$$\gamma \cdot \text{coh}(e_{j_1} \in \text{cnd}(m_1) \ldots e_{j_k} \in \text{cnd}(m_k)) = \max!$$

where $\alpha + \beta + \gamma = 1$, $\text{cnd}(m_i)$ is the set of possible meanings of m_i, cxt( ) denotes the context of mentions and entities, respectively, and coh( ) is the coherence function for a set of entities.

Section 4 gives details on each of these three components. For robustness, our solution selectively enables or disables the three components, based on tests on the mentions of the input text; see Section 5.

4 Features and Measures

4.1 Popularity Prior

As mentioned above, our framework supports multiple forms of popularity-based priors, but we found a model based on Wikipedia link anchors to be most effective: For each surface form that constitutes an anchor text, we count how often it refers to a particular entity. For each name, these counts provide us with an estimate for a probability distribution over candidate entities. For example, "Kashmir" refers to `Kashmir` (the region) in 90.91% of all occurrences and in 5.45% to `Kashmir (Song)`.

4.2 Mention-Entity Similarity

Keyphrase-based Similarity: On the **mention side**, we use all tokens in the document (except stopwords and the mention itself) as context. We experimented with a distance discount to discount the weight of tokens that are further away, but this did not improve the results for our test data.

On the **entity side**, the knowledge base knows authoritative sources for each entity, for example, the

corresponding Wikipedia article or an organizational or individual homepage. These are the inputs for an offline data-mining step to determine characteristic *keyphrases* for each entity and their statistical weights. We describe this only for Wikipedia as input corpus, the approach extends to other inputs. As keyphrase candidates for an entity we consider its corresponding Wikipedia article's link anchors texts, including category names, citation titles, and external references. We extended this further by considering also the titles of articles linking to the entity's article. All these phrases form the keyphrase set of an entity: $KP(e)$.

For each word w that occurs in a keyphrase, we compute a *specificity weight* with regard to the given entity: the **MI** (mutual information) between the entity e and the keyword w, calculating the joint probabilities for MI as follows:

$$p(e,w) = \frac{\left|w \in \left(KP(e) \cup \bigcup_{e' \in IN_e} KP(e')\right)\right|}{N}$$

reflecting if w is contained in the keyphrase set of e or any of the keyphrase sets of an entity linking to e, $IN(e)$, with N denoting the total number of entities. The joint probabilities for the cases $p(e,\bar{w})$, $p(\bar{e},w)$, $p(\bar{e},\bar{w})$ are calculated accordingly.

Keyphrases may occur only partially in an input text. For example, the phrase "Grammy Award winner" associated with entity `Jimmy Page` may occur only in the form "Grammy winner" near some mention "Page". Therefore, our algorithm for the similarity of mention m with regard to entity e computes partial matches of e's keyphrases in the text. This is done by matching individual words and rewarding their proximity in an appropriate score. To this end we compute, for each keyphrase, the shortest window of words that contains a maximal number of words of the keyphrase. We refer to this window as the phrase's *cover* (cf. (Taneva11)). For example, matching the text "winner of many prizes including the Grammy" results in a cover length of 7 for the keyphrase "Grammy award winner". By this rationale, the score of partially matching phrase q in a text is set to:

$$score(q) = z \left(\frac{\sum_{w \in cover} weight(w)}{\sum_{w \in q} weight(w)} \right)^2$$

where $z = \frac{\#\ matching\ words}{length\ of\ cover(q)}$ and $weight(w)$ is either the **MI** weight (defined above) or the collection-wide **IDF** weight of the keyphrase word w. Note that the second factor is squared, so that there is a superlinear reduction of the score for each word that is missing in the cover.

For the similarity of a mention m to candidate entity e, this score is aggregated over all keyphrases of e and all their partial matches in the text, leading to the *similarity score*

$$simscore(m,e) = \sum_{q \in KP(e)} score(q)$$

Syntax-based Similarity: In addition to surface features of words and phrases, we leverage information about the immediate syntactic context in which an entity mention occurs. For example, in the sentence "Page played unusual chords", we can extract the fact that the mention "Page" is the subject of the verb "play". Using a large text corpus for training, we collect statistics about what kinds of entities tend to occur as subjects of "play", and then rank the candidate entities according to their compatibility with the verb.

Specifically, we employ the framework of (Thater10), which allows us to derive vector representations of words in syntactic contexts (such as being the subject of a particular verb). We do not directly apply this model to derive contextualized representations of entity mentions, as information about specific proper names is very sparse in corpora like GigaWord or Wikipedia. Instead, we consider a set of *substitutes* for each possible entity e, which we take as its context cxt(e). For this, we use the WordNet synsets associated with the entity's YAGO types and all their hypernyms. For each substitute, we compute a standard distributional vector and a contextualized vector according to (Thater10). Syntax-based similarity between cxt(e) and the context cxt(m) of the mention is then defined as the sum of the scalar-product similarity between these two vectors for each substitute. This results in high similarity if the syntactic contextualization only leads to small changes of the vectors, reflecting the compatibility of the entity's substitutes.

In our example, we compute a vector for "guitarist" as subject of "play", and another one for "entrepreneur" in the same context. The former is more

compatible with the given context than the latter, leading to higher similarity for the entity `Jimmy Page`.

4.3 Entity-Entity Coherence

As all entities of interest are registered in a knowledge base (like YAGO), we can utilize the *semantic type system*, which is usually a DAG of classes. The simples measure is the distance between two entities in terms of `type` and `subclassOf` edges.

The knowledge bases also provide same-as cross-referencing to Wikipedia, amd we quantify the coherence between two entities by the number of *incoming links* that their Wikipedia articles share. This approach has been refined by Milne and Witten (Milne08), taking into account the total number N of entities in the (Wikipedia) collection:

$$mw\_coh(e_1, e_2) = 1 - \frac{\log(\max(|IN_{e_1}|, |IN_{e_2}|)) - \log(|IN_{e_1} \cap IN_{e_2}|)}{\log(|N|) - \log(\min(|IN_{e_1}|, |IN_{e_2}|))}$$

if > 0 and else set to 0.

5 Graph Model and Algorithms

5.1 Mention-Entity Graph

From the popularity, similarity, and coherence measures discussed in Section 4, we construct a weighted, undirected graph with mentions and candidate entities as nodes. As shown in the example of Figure 1, the graph has two kinds of edges:

- A mention-entity edge is weighted with a similarity measure or a combination of popularity and similarity measure. Our experiments will use a linear combination with coefficients learned from withheld training data.
- An entity-entity edge is weighted based on Wikipedia-link overlap, or type distance, or some combination along these lines.

Our experiments will focus on anchor-based popularity, keyphrase-based and/or syntactic similarity, and link-based coherence (mw_coh). The mention-entity graph is dense on the entities side and often has hundreds or thousands of nodes, as the YAGO knowledge base offers many candidate entities for common mentions (e.g., country names that could also denote sports teams, common lastnames, firstnames, etc.).

5.2 Graph Algorithm

Given a mention-entity graph, our goal is to compute a *dense subgraph* that would ideally contain all mention nodes and exactly one mention-entity edge for each mention, thus disambiguating all mentions. We face two main challenges here. The first is how to specify a notion of density that is best suited for capturing the coherence of the resulting entity nodes. The seemingly most natural approach would be to measure the density of a subgraph in terms of its total edge weight. Unfortunately, this will not work robustly for the disambiguation problem. The solution could be dominated by a few entity nodes with very high weights of incident edges, so the approach could work for prominent targets, but it would not achieve high accuracy also for the long tail of less prominent and more sparsely connected entities. We need to capture the weak links in the collective entity set of the desired subgraph. For this purpose, we define the *weighted degree* of a node in the graph to be the total weight of its incident edges. We then define the density of a subgraph to be equal to the minimum weighted degree among its nodes. Our goal is to compute a subgraph with maximum density, while observing constraints on the subgraph structure.

The second critical challenge that we need to face is the computational complexity. Dense-subgraph problems are almost inevitably NP-hard as they generalize the Steiner-tree problem. Hence, exact algorithms on large input graphs are infeasible.

To address this problem, we adopt and extend an approximation algorithm of (Sozio10) for the problem of finding strongly interconnected, size-limited groups in social networks. The algorithm starts from the full mention-entity graph and iteratively removes the entity node with the smallest weighted degree. Among the subgraphs obtained in the various steps, the one maximizing the minimum weighted degree will be returned as output. To guarantee that we arrive at a coherent mention-entity mapping for all mentions, we enforce each mention node to remain connected to at least one entity. However, this constraint may lead to very suboptimal results.

For this reason, we apply a pre-processing phase to prune the entities that are only remotely related to the mention nodes. For each entity node, we compute the distance from the set of all mention nodes in terms

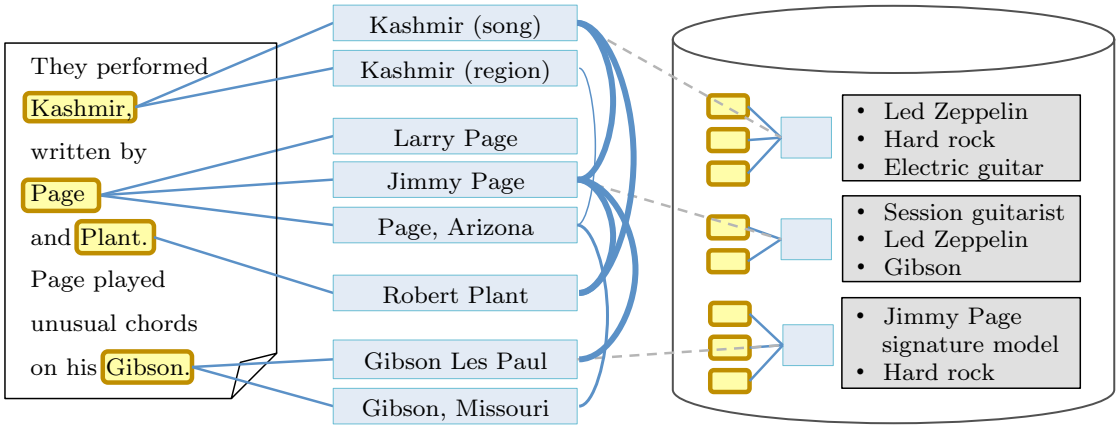

Figure 1: Mention-Entity Graph Example

of the sum of the corresponding squared shortest-path distances. We then restrict the input graph to the entity nodes that are closest to the mentions. An experimentally determined good choice for the size of this set is five times the number of the mention nodes. Then the iterative greedy method is run on this smaller subgraph. Algorithm 1 summarizes this procedure, where an `entity` is `taboo` if it is the last candidate for a mention it is connected to.

Algorithm 1: Graph Disambiguation Algorithm

Input: weighted graph of mentions and entities
Output: result graph with one edge per mention
begin
  pre–processing phase;
  foreach `entity` **do**
    calculate distance to all mentions;
  keep the closest ($5\times$ `mentions_count`) `entities`, drop the others;
  main loop;
  while *graph has* `non-taboo entity` **do**
    determine `non-taboo entity` node with lowest weighted degree, remove it and all its incident edges;
    if *minimum weighted degree increased* **then**
      set `solution` to current graph;
  post–processing phase;
  process `solution` by local search or full enumeration for best configuration;

The output of the main loop would often be close to the desired result, but may still have more than one mention-entity edge for one or more mentions. At this point, however, the subgraph is small enough to consider an exhaustive enumeration and assessment of all possible solutions. This is one of the options that we have implemented as post-processing step. Alternatively, we can perform a faster local-search algorithm. Candidate entities are randomly selected with probabilities proportional to their weighted degrees. This step is repeated for a prespecified number of iterations, and the best configuration with the highest total edge-weight is used as final solution.

5.3 Robustness Tests

The graph algorithm generally performs well. However, it may be misled in specific situations, namely, if the input text is very short, or if it is thematically heterogeneous. To overcome these problems, we introduce two *robustness tests* for individual mentions and, depending on the tests' outcomes, use only a subset of our framework's features and techniques.

Prior test: Our first test ensures that the popularity prior does not unduly dominate the outcome if the true entities are dominated by false alternatives. We check, for each mention, whether the popularity prior for the most likely candidate entity is above some threshold ρ, e. g. above 90% probability. If this is not the case, then the prior is completely disregarded for computing the mention-entity edge weights. Otherwise, the prior is combined with the context-based similarity computation to determine edge weights.

We never rely solely on the prior.

Coherence test: As a test for whether the coherence part of our framework makes sense or not, we compare the popularity prior and the similarity-only measure, on a per-mention basis. For each mention, we compute the $L1$ distance between the popularity-based vector of candidate probabilities and the similarity-only-based vector of candidate probabilities:

$$\sum_{i=1..k} |\text{prior}(m, e_i) - \text{simscore}(m, e_i)|$$

This difference is always between 0 and 2. If it exceeds a specified threshold λ (e.g., 1), the disagreement between popularity and similarity-only indicates that there is a situation that coherence may be able to fix. If, on the other hand, there is hardly any disagreement, using coherence as an additional aspect would be risky for thematically heterogeneous texts and should better be disabled. In that case, we choose an entity for the mention at hand, using the combination of prior and similarity. Only the winning entity is included in the mention-entity graph, all other candidates are omitted for the graph algorithm. The robustness tests and the resulting adaptation of our method are fully automated.

6 Experiments

6.1 Setup

System: All described methods are implemented in a prototype system called AIDA (Accurate Online Disambiguation of Named Entities). We use the Stanford NER tagger (Finkel05) to identify mentions in input texts, the YAGO2 knowledge base (Hoffart11) as a repository of entities, and the English Wikipedia edition (as of 2010-08-17) as a source of mining keyphrases and various forms of weights. The graph algorithm makes use of Webgraph (Boldi04).

Datasets: There is no established benchmark for NED. The best prior work (Kulkarni09)) compiled its own hand-annotated dataset, sampled from online news. Unfortunately, this data set is fairly small (102 short news articles, about 3,500 proper noun mentions). Moreover, its entity annotations refer to an old version of Wikipedia. To avoid unfair comparisons, we created our own dataset based on CoNLL 2003 data, extensively used in prior work on NER tagging (Sang03).

articles	1,393
mentions (total)	34,956
mentions with no entity	7,136
words per article (avg.)	216
mentions per article (avg.)	25
distinct mentions per article (avg.)	17
mentions with candidate in KB (avg.)	21
entities per mention (avg)	73
initial annotator disagreement (%)	21.1

Table 1: *CoNLL* Dataset Properties

This consists of proper noun annotations for 1393 Reuters newswire articles. We hand-annotated all these proper nouns with corresponding entities in YAGO2. Each mention was disambiguated by two students and resolved by us in case of conflict. This data set is referred to as *CoNLL* in the following and fully available at `http://www.mpi-inf.mpg.de/yago-naga/aida/`. Table 1 summarizes properties of the dataset.

Methods under comparison: Our framework includes many variants of prior methods from the literature. We report experimental results for some of them. AIDA's parameters were tuned by line-search on 216 withheld development documents. We found the following to work best:

- threshold for prior test: $\rho = 0.9$
- weights for popularity, similarity, coherence: $\alpha = 0.43$, $\beta = 0.47$, $\gamma = 0.10$
- initial number of entites in graph: $5 \cdot \#mentions$
- threshold for coherence test: $\lambda = 0.9$

We checked the sensitivity of the hyper-parameter settings and found the influence of variations to be small, e. g. when varying λ within the range [0.5,1.3], the changes in precision@1.0 are within 1%.

The baseline for our experiments is the collective-inference method of (Kulkarni09), which outperforms simpler methods (such as (Milne08)). We refer to this method as *Kul_CI*. Since program code for this method is not available, we re-implemented it using the LP solver CPLEX for the optimization problem with subsequent rounding, as described in (Kulkarni09). In addition, we compare against (our re-implementation of) the method of (Cucerzan07),

	Our Methods							Competitors				
	sim-k	prior sim-k	prior sim-s	sim-k sim-s	r-prior sim-k	r-prior sim-k coh	r-prior sim-k r-coh	prior	Cuc	Kul_s	Kul_sp	Kul_CI
Macro P@1.0	76.53	75.75	71.43	76.40	80.71	80.73	**81.91**	71.24	43.74	58.06	76.74	76.74
Micro P@1.0	76.09	70.72	66.09	76.13	79.57	81.77	**81.82**	65.84	51.03	63.42	72.31	72.87
MAP	66.98	83.99	85.97	67.00	85.91	**89.05**	87.31	86.63	40.06	63.90	86.50	85.44

Table 2: Experimental results on *CoNLL* (all values in %)

referred to as *Cuc*. For all methods, weights for combining components were obtained by training a SVM classifier on 946 withheld *CoNLL* training documents.

Performance measures: The key measures in our evaluation are *precision* and *recall*. We consider the precision-recall curve, as there is an inherent trade-off between the two measures. Precision is the fraction of mention-entity assignments that match the ground-truth assignment. Recall is the fraction of the ground-truth assignments that our method(s) could compute. Both measures can aggregate over of all mentions (across all texts) or over all input texts (each with several mentions). The former is called micro-averaging, the latter macro-averaging.

As we use a knowledge base with millions of entities, we decided to neglect the situation that a mention may refer to an unknown entity not registered in the knowledge base. We consider only mention-entity pairs where the ground-truth gives a known entity, and thus ignore roughly 20% of the mentions without known entity in the ground-truth. This simplifies the calculation of aggregated precision-recall measures like (interpolated) MAP (mean average precision):

$$\text{MAP} = \frac{1}{m} \sum_{i=1..m} \text{precision@}\frac{i}{m}$$

where precision@$\frac{i}{m}$ is the precision at a specific recall level. This measure is equivalent to the area under the precision-recall curve.

For constructing the precision-recall curve, we sort the mention-entity pairs in descending order of confidence, so that x% recall refers to the x% with the highest confidence. We use each method's mention-entity similarity for the confidence values.

6.2 Results

The results of AIDA vs. the collective-inference method of (Kulkarni09) and the entity disambiguation method of (Cucerzan07) on 229 test documents are shown in Table 2[1]. The table includes variants of our framework, with different choices for the similarity and coherence computations. The shorthand notation for the combinations in the table is as follows: *prior*: popularity prior; *r-prior*: popularity prior with robustness test; *sim-k*: keyphrase based similarity measure; *sim-s*: syntax-based similarity; *coh*: graph coherence; *r-coh*: graph coherence with robustness test.

The shorthand names for competitors are: *Cuc*: (Cucerzan07) similarity measure; *Kul_s*: (Kulkarni09) similarity measure only; *Kul_sp*: Kul_s combined with plus popularity prior; *Kul_CI*: Kul_sp combined with coherence. All coherence methods use the Milne-Witten inlink overlap measure mw_coh.

The most important measure is macro/micro precison@1.0, which corresponds to the overall correctness of the methods for all mentions that are assigned to an entity in the ground-truth data. Our *sim-k* precision is already very good. Combining it with the syntax-based similarity improves micro-averaged precision@1.0, but the macro-averaged results are a bit worse. Thus, the more advanced configurations of AIDA did not use syntax-based similarity. Unconditionally combining *prior* and *sim-k* degrades the quality, but including the prior robustness test (*r-prior sim-k*) improves the results significantly. The precision for our best method, the prior- and coherence-tested Keyphrase-based mention-entity similarity (*r-prior sim-k r-coh*), significantly outperforms all competitors (with a p-value of a paired t-test < 0.01). Our macro-averaged precision@1.0 is 81.91%, whereas *Kul_CI* only achieves 76.74%. Even *r-prior sim-k*, without any coherence, significantly outperforms

[1] 2 of the 231documents in the original test set could not be processed by *Kul_CI* due to memory limitations. All results are given for the subset, for the sake of comparability. Results for the complete set are available on our website.

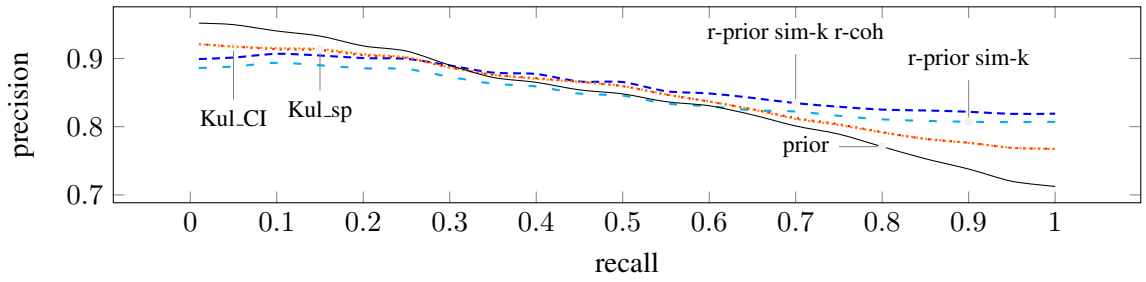

Figure 2: Experimental results on *CoNLL*: precision-recall curves

Kul_CI (with coherence) with a p-value of < 0.01. In micro-average precision@1.0, the differences are even higher, showing that we perform better throughout all documents.

The macro-averaged precision-recall curves in Figure 2 show that the best AIDA method performs particularly well in the tail of high recall values. The MAP underlines the robustness of our best methods.

The high MAP for the *prior* method is because we rank by mention-entity edge weight; for *prior* this is simply the prior probability. As the prior is most probably correct for mentions with a very high prior for their most popular entity (by definition), the initial ranking of the prior is very good, but drops more sharply. We believe that the main difficulty in named entity disambiguation lies exactly in the "long tail" of not-so-prominent entities.

We also tried the (Milne08) web service on a subset of our test collection, but this was obviously geared for Wikipedia linkage and performed poorly.

6.3 Discussion

Our keyphrase-based similarity measure performs better than the *Kul_s* measure, which is a combination of 4 different entity contexts (abstract tokens, full text tokens, inlink anchor tokens, inlink anchor tokens + surrounding tokens), 3 similarity measures (Jaccard, dot product, and tf.idf cosine similarity), and the popularity prior. Adding the prior to our similarity measure by linear combination degrades the performance. We found that our measure already captures a notion of popularity because popular entities have more keyphrases and can thus accumulate a higher total score. The popularity should only be used when one entitiy has a very high probability, and introducing the robustness test for the prior achieved this, improving on both our similarity and *Kul_sp*.

Unconditionally adding the notion of coherence among entities improves the micro-average precision, but not the macro-average. Investigating potential problems, we found that the coherence can be led astray when parts of the document form a coherent cluster of entities, and other entities are then forced to be coherent to this cluster. To overcome this issue, we introduced the coherence robustness test, and the results with *r-coh* show that it makes sense to fix an entity for a mention when the prior and similarity are in reasonable agreement. Adding this coherence test leads to a signigicant (p-value < 0.05) improvement over the non-coherence based measures in both micro- and macro-average precision. Our experiments showed that when adding this coherence test, around $\frac{2}{3}$ of the mentions are solved using local similarity only and are assigned an entity before running the graph algorithm. In summary, we observed that the AIDA configuration with *r-prior*, keyphrase-based *sim-k*, and *r-coh* significantly outperformed all competitors.

7 Conclusions and Future Work

The AIDA system provides an integrated NED method using popularity, similarity, and graph-based coherence, and includes robustness tests for self-adaptive behavior. AIDA performed significantly better than state-of-the-art baselines. The system is fully implemented and accessible online (`http://www.mpi-inf.mpg.de/yago-naga/aida/`). Our future work will consider additional semantic properties between entities (types, memberOf/partOf, etc.) for further enhancing the coherence algorithm.

Acknowledgements

This work has been partially supported by the German Science Foundation (DFG) through the Cluster of Excellence on "Multimodal Computing and Interaction" and the European Union through the 7th Framework IST Integrated Project "LivingKnowledge" (no. 231126). We also thank Mauro Sozio for the discussion on the graph algorithm.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, Zachary G. Ives: DBpedia: A Nucleus for a Web of Open Data. ISWC 2007

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, Oren Etzioni: Open Information Extraction from the Web. IJCAI 2007

Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: Compression techniques. WWW 2004, software at `http://webgraph.dsi.unimi.it/`

Razvan C. Bunescu, Marius Pasca: Using Encyclopedic Knowledge for Named entity Disambiguation. EACL 2006

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. AAAI 2010

Silviu Cucerzan: Large-Scale Named Entity Disambiguation Based on Wikipedia Data. EMNLP-CoNLL 2007

AnHai Doan, Luis Gravano, Raghu Ramakrishnan, Shivakumar Vaithyanathan. (Eds.). Special issue on information extraction. SIGMOD Record, 37(4), 2008.

Jenny Rose Finkel, Trond Grenager, Christopher Manning: Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. ACL 2005, software at `http://nlp.stanford.edu/software/CRF-NER.shtml`

Xianpei Han, Jun Zhao: Named entity disambiguation by leveraging wikipedia semantic knowledge. CIKM 2009.

Johannes Hoffart, Fabian Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, Gerhard Weikum: YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. Demo Paper, WWW 2011, data at `http://www.mpi-inf.mpg.de/yago-naga/yago/`

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, Soumen Chakrabarti: Collective annotation of Wikipedia entities in web text. KDD 2009

James Mayfield et al.: Corss-Document Coreference Resolution: A Key Technology for Learning by Reading. AAAI Spring Symposium on Learning by Reading and Learning to Read, 2009.

Diane McCarthy. Word Sense Disambiguation: An Overview. Language and Linguistics Compass 3(2): 537-558, Wiley, 2009

Rada Mihalcea, Andras Csomai: Wikify!: Linking Documents to Encyclopedic Knowledge. CIKM 2007

David N. Milne, Ian H. Witten: Learning to Link with Wikipedia. CIKM 2008

Ndapandula Nakashole, Martin Theobald, Gerhard Weikum: Scalable Knowledge Harvesting with High Precision and High Recall. WSDM 2011

Roberto Navigli: Word sense disambiguation: A survey. ACM Comput. Surv., 41(2), 2009

Hien T. Nguyen, Tru H. Cao: Named Entity Disambiguation on an Ontology Enriched by Wikipedia. RIVF 2008

Erik F. Tjong Kim Sang, Fien De Meulder: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. CoNLL 2003

Mauro Sozio, Aristides Gionis: The Community-search Problem and How to Plan a Successful Cocktail Party. KDD 2010

Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum: YAGO: a Core of Semantic Knowledge. WWW 2007

Fabian Suchanek, Mauro Sozio, Gerhard Weikum: SOFIE: a Self-Organizing Framework for Information Extraction. WWW 2009

Bilyana Taneva, Mouna Kacimi, and Gerhard Weikum: Finding Images of Rare and Ambiguous Entities. Technical Report MPI-I-2011-5-002, Max Planck Institute for Informatics, 2011.

Stefan Thater, Hagen Fürstenau, Manfred Pinkal. Contextualizing Semantic Representations using Syntactically Enriched Vector Models. ACL 2010

Michael L. Wick, Aron Culotta, Khashayar Rohanimanesh, Andrew McCallum: An Entity Based Model for Coreference Resolution. SDM 2009: 365-376

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, Ji-Rong Wen: StatSnowball: a Statistical Approach to Extracting Entity Relationships. WWW 2009