Summary of Changes

CHANGES

Below lists several significant extensions over our previous work.

- First, we revised and simplified our optimization goal, resulting in a more robust greedy iterative algorithm with fewer tuning parameters.
- Second, we show how to tune our method with an off-the-shelf standard "learning-to-rank" approach. We apply this idea using an existing manually-annotated dataset (the CoNLL dataset widely used in the literature) leading to an algorithm that consistently outperforms all previous methods, across benchmarks and by a wide margin.
- Third, we report on a deeper experimental evaluation than previous works in the area: (1) our analysis shows that previous benchmarks are "easy", in the sense that a simple baseline can correctly disambiguate most mentions; (2) we introduce two benchmarks from real Web corpora (Wikipedia and Clueweb 2012) with documents of increasing difficulty, which are also *balanced* (i.e., they have the same number of documents in each difficulty class); finally, (3) we aggregate the per-document accuracy in a more meaningful way.

Undefined 0 (2016) 1–0
IOS Press

Robust Named Entity Disambiguation with Random Walks

Zhaochen Guo and Denilson Barbosa *
Department of Computing Science, University of Alberta, Edmonton, AB, Canada.
E-mail: {zhaochen, denilson}@ualberta.ca

Abstract. Named Entity Disambiguation is the task of assigning entities from a Knowledge Base to *mentions* of such entities in a textual document. This article presents two novel approaches guided by a natural notion of *semantic similarity* for the collective disambiguation of all entities mentioned in a document at the same time. We adopt a unified semantic representation for entities and documents—the probability distribution obtained from a random walk on a subgraph of the knowledge base—as well as lexical and statistical features commonly used for this task. The first approach is an iterative and greedy approximation, while the second is based on learning-to-rank. Our experimental evaluation uses well-known benchmarks as well as new ones introduced here. We justify the need for these new benchmarks, and show that both our methods outperform the previous state-of-the-art by a wide margin.

Keywords: Named entities, entity linking, entity disambiguation, relatedness measure, random walk

1. Introduction

Named entities are the persons, organizations, locations, etc. that are explicitly mentioned in text using proper nouns. *Named Entity Recognition* (NER), the task of finding named entities in text and assigning them a semantic type corresponding to the kind of entity, is a key step towards algorithmic understanding of natural language text, especially in the context of the Web and social media where facts or properties about named entities are described in many documents. The NER task is often performed in conjunction with other text processing to *resolve* pronouns and/or abbreviations in the text to the actual named entities they refer to, a task called co-reference resolution [17].

Named Entity Disambiguation (NED), also known as *Entity Linking*, is the task of linking the named entities mentioned in the text (explicitly or implicitly via a pronoun or abbreviation) to pre-existing objects in a Knowledge Base (KB) of interest, thus grounding them in a surrogate to a real world entity. NED is key for Information Extraction (IE) and thus has many applications, such as: (1) expanding or correcting KBs with facts of entities extracted from text [16]; (2) semantic search [27], the emerging paradigm of Web search that combines Information Retrieval approaches over document corpora with KB-style query answering and reasoning; and (3) in Natural Language Question Answering [33].

Our work mainly focused on the NED task. This article describes highly effective algorithms for solving this problem, assuming the input is a KB and a document where all mentions to named entities (explicit or implicit) have been identified.

Challenges The inherent ambiguity of natural language makes NED a hard problem, even to humans. Most real world entities can be referred to in many different ways (e.g., people have nicknames), while the same textual mention may refer to multiple real-world entities (e.g., different people have the same name). The following examples illustrate the issues:

Example 1
Saban, previously a head coach of NFL's Miami, is now coaching Crimson Tide. His achievements include

*Corresponding author. E-mail: denilson@ualberta.ca

leading LSU to the BCS National Championship once and Alabama three times.

Example 2
After his departure from Buffalo, Saban returned to coach college football teams including Miami, Army and UCF.

In Example 1, the mentions "Crimson Tide" and "Alabama" refer to the same football team, the *Alabama Crimson Tide* of the University of Alabama, while the mention "Saban" refers to two different people (both football coaches): *Nick Saban* in Example 1 and *Lou Saban* in Example 2. Similarly, "Miami" in Example 1 refers to the NFL football team *Miami Dolphins*, while in Example 2 it refers to the college football team *Miami Hurricanes*.

1.1. Canonical Solution

The NED task can be cast as an all-against-all matching problem: given m mentions in a document and n entities in a KB, perform the $m \times n$ comparisons and pick the ones with the highest similarity (one for each mention). This is prohibitively expensive and unnecessary, however. Most methods, including ours, solve this issue in two stages: the first stage reduces the search space by selecting a suitable set of candidate entities for each mention, and the second performs the actual mention disambiguation. Selecting candidate entities is done, primarily, by consulting alias dictionaries (see, e.g., [30]). Candidate selection is meant to be fast, and thus leads to false positives which must be filtered out in the actual disambiguation phase. In both examples above, both coaches *Lou Saban* and *Nick Saban* would be picked as candidates for the mention "Saban", as would other kinds of entities, such as the organizations *Saban Capital Group* and *Saban Entertainment*.

As for the disambiguation phase, the methods in the literature can be divided into two groups: local and global disambiguation methods, as discussed next.

Local Disambiguation The first group of NED systems focused mainly on lexical features, and statistical features, such as contextual words surrounding the mentions in the document [2,3] or statistical probability from knowledge bases. Moreover, mentions are disambiguated independently, typically by ranking the entities according to the similarity between the context vector of the mention and the text describing the entities in the KB (e.g., *keyphrases*). These approaches work best when the context is rich enough to uniquely identify the mentions or the linked entities are popular enough, which is not always the case. For instance, both "Saban" and "Miami" in the examples above are hard to disambiguate locally because *Lou Saban* coached the *Miami Hurricanes* while *Nick Saban* coached the *Miami Dolphins* and such a dependency cannot be enforced via local methods based on context similarity. In fact, these kinds of dependencies are the main motivation for the global disambiguation methods.

Global Disambiguation Most current approaches are based on the premise that the disambiguation of one mention should affect the disambiguation of the remaining mentions (e.g., disambiguating *Saban* to *Nick Saban* should increase the confidence for *Miami* to be linked to the *Miami Dolphins*). In general, the idea is to start with a *disambiguation graph* containing all mentions in the document and all *candidate entities* from the KB. In many global NED systems, the disambiguation graph also contains the immediate neighbors of the candidates (Fig. 1). The actual disambiguation is done *collectively* on all mentions at the same time [7,14,19,28], by finding an embedded forest in the disambiguation graph in which each mention remains linked to just one of the candidates. The edges are weighted according to multiple criteria, including local (context) similarity, priors, and some notion of *semantic* similarity, possibly computed from the graph itself. Finding such an assignment is NP-Hard [19] under reasonable assumptions, and all methods turn to approximate algorithms or heuristics. One common way to cope with the complexity is to use a greedy search, starting with mentions that are easy to disambiguate (e.g., have just one candidate entity) [25].

The choice of semantic similarity determines the accuracy and cost of each method. One successful strategy [13] computes the set-similarity involving (multiword) *keyphrases* about the mentions and the entities, collected from the KB. This approach works best when the named entities in the document are mentioned in similar ways to those in the corpus from which the knowledge base is built (typically, Wikipedia). Another approach [24] computes the set similarity between the neighbor entities directly connected in the KB. Doing so, however, ignores entities that are indirectly connected yet semantically related, making limited use of the KB graph.

In previous work [10], we introduced a method that used an information-theoretic notion of similarity

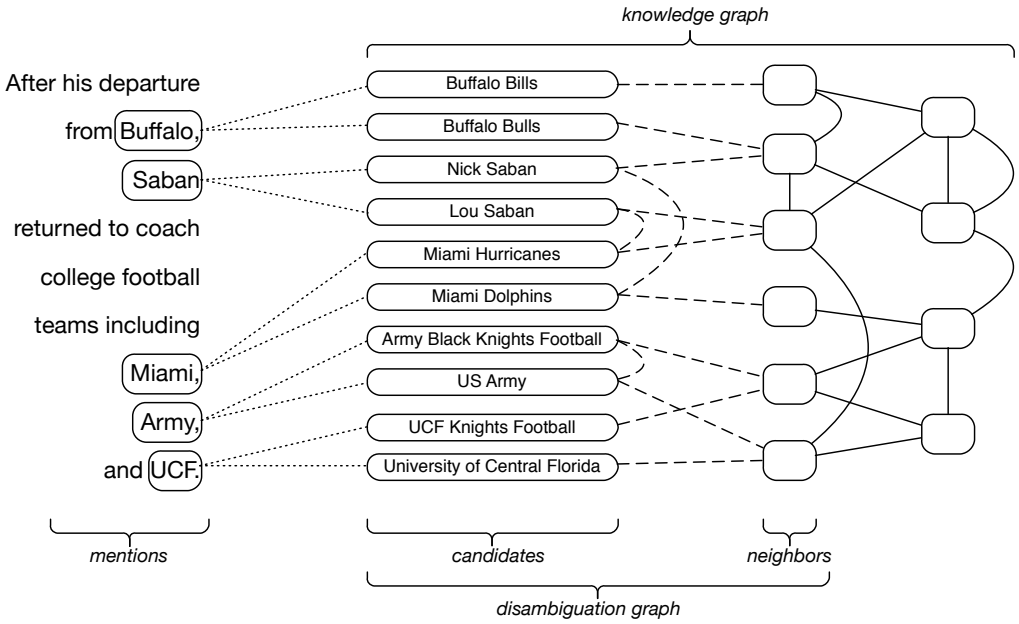


Fig. 1. Example named entity disambiguation scenario.

based on stationary probability distributions resulting from random walks [31] on the disambiguation graph, which led to consistently superior accuracy. This paper presents substantial extensions over that method.

1.2. Our approach

Our WNED (Walking Named Entity Disambiguation) method is a greedy, global NED algorithm based on a sound information-theoretic notion of semantic relatedness derived from random walks on carefully built disambiguation graphs [10]. We build specific disambiguation graphs for each document, thus adhering to the notion of global coherence assumption—that coherent entities form a dense subgraph. By virtue of using random walks, our notion of similarity leverages indirect connections between nodes in the disambiguation graph, and is thus less susceptible to false positives incurred by disproportionately high priors of head entities. As confirmed by our experiments, our approach outperforms the previous state-of-the-art, and excels in disambiguating mentions of less popular entities.

Contributions This article presents several significant extensions over our previous work.

- First, we revised and simplified our optimization goal, resulting in a more robust greedy iterative algorithm with fewer tuning parameters.
- Second, we show how to tune our method with an off-the-shelf standard "learning-to-rank" approach. We apply this idea using an existing manually-annotated dataset (the CoNLL dataset widely used in the literature) leading to an algorithm that consistently outperforms all previous methods, across benchmarks and by a wide margin.
- Third, we report on a deeper experimental evaluation than previous works in the area: (1) our analysis shows that previous benchmarks are "easy", in the sense that a simple baseline can correctly disambiguate most mentions; (2) we introduce two benchmarks from real Web corpora (Wikipedia and Clueweb 2012) with documents of increasing difficulty, which are also *balanced* (i.e., they have the same number of documents in each difficulty class); finally, (3) we aggregate the per-document accuracy in a more meaningful way.

It is also worth noting that our statically hand-tuned algorithm also outperformed all previous methods and is quite competitive with the learning approach. These observations corroborate the superiority and robustness of using random walks for the NED task. In summary, the algorithm and the evaluation methodology described in this article significantly push the state-of-the-art in this task.

2. NED with Random Walks

This section first describes Named Entity Disambiguation as an optimization problem and then gives an overview of our solution based on using Random Walks to estimate semantic similarities (Section 3) and a greedy, iterative approximation algorithm (Section 4).

2.1. The NED Problem

Let d be a document with all mentions to named entities marked up through an NER process, and $KB = (E, L)$ be a knowledge base represented as a graph whose nodes in E correspond to real world entities and links in L capture relationships among them. The task of NED is to assign unique entity identifiers from E to the mentions in d, whenever appropriate. NIL is used for mentions outside of the KB.

More precisely:

Definition 1 (Named Entity Disambiguation) *Given a set of mentions $M = \{m_1, \ldots, m_m\}$ in a document d, and a knowledge base $KB = (E, L)$, the NED problem is to find an assignment $A : M \rightarrow E \cup \{\mathsf{NIL}\}$.*

A good assignment A balances two forces: the *local* similarity between a mention m_i and its linked entity $e_j = A(m_i)$, and the *global* coherence among all entities in the assignment.

As usual, we define $local(m_i, e_j)$ as:

$$local(m_i, e_j) = \alpha\, prior(m_i, e_j) + (1-\alpha)\, context(m_i, e_j) \tag{1}$$

where $prior(m_i, e_j)$ is a corpus prior probability that e_j is the right entity for m_i, usually derived from alias dictionaries built from the KB, and $context(m_i, e_j)$ is the similarity between *local* features extracted from text (e.g., keywords) surrounding m_i in the document and descriptions associated to e_j in the KB.

Conversely, $global(A)$ captures the global coherence assumption that all entities in the assignment form a semantically coherent unit, as follows:

$$global(A) = \sum_{e \in A[M]} semantic(e, A) \tag{2}$$

in which $semantic(e, A)$ measures the semantic similarity between an entity e and all others in the assignment A. Maximizing the sum in Eq. 2 is consistent with the *document coherence assumption*, in which one expects the input document to belong to a single *topic* (e.g., sports) under which all entities in the assignment A are related. In fact, this maximization is what allows the *collective* disambiguation of mentions (e.g., disambiguating the mention Saban to Nick Saban in Ex. 2 increases the odds of linking Miami to the Miami Dolphins).

Our notion of semantic coherence, rooted in Information Theory, corresponds to the mutual information between probability distributions obtained from the stationary distributions from two separate *random walk* processes on the entity graph: one always restarting from the entity, and the other restarting from all entities used in the assignment. This is accomplished by appropriately defining random restarting probability to each entity in the preference vectors used in the walks accordingly.

Under the reasonable assumption that the local similarity is normalized, we can formulate NED as a min-max optimization where the goal is to *maximize* the global coherence while minimizing the *loss* in local pairwise similarity by the assignment, which can be estimated as $|M| - \sum_{m_i \in M} local(m_i, A(m_i))$. An equivalent and simpler formulation of the problem is to find an assignment A^* that maximizes:

$$A^* = \arg\max_A \left(global(A) \cdot \sum_{m_i, e_j \in A} local(m_i, e_j) \right) \tag{3}$$

Note that both global coherence and local similarity are normalized among all candidates of each mention.

2.2. Iterative Walking NED

As previously observed (see, e.g., [14]), the NED problem is intimately connected with a number of NP-hard optimizations on graphs, including the maximum m-clique problem [9], from which a polynomial time reduction is not hard to construct. Thus we resort to an iterative heuristic solution, which works as follows.

We start with a disambiguation graph G (recall Fig. 1) containing all candidate entities and their immediate neighbors. We order the mentions by their number of candidates (i.e., their degree of ambiguity), and iterate over them in increasing order, incrementally building the *approximate* assignment one mention

at a time. In iteration i, we assign to mention m_i the entity e maximizing

$$A(m_i) = \underset{e \in cand(m_i)}{\arg\max} \ (semantic(e, \hat{A}) \cdot local(m_i, e)) \quad (4)$$

Here, $semantic(e, \hat{A})$ is the mutual information between the probability distributions generated from random walks restarting from e and from *all entities* in the partial assignment up to step $i-1$. More precisely, let n be the size of G. We define the *semantic signature* of candidate entity $e \in G$ the n-dimensional vector corresponding to the stationary probability distribution of reaching each node in G from e, obtained through a random walk that always restarts from e. The *semantic signature* of the partial assignment $\hat{A}$ is also an n-dimensional vector with the stationary probability obtained from a random walk restarting from the entities corresponding to all mentions disambiguated up until that point.

To disambiguate n mentions, we re-compute $\hat{A}$ and its corresponding signature $n-1$ times. In the first step, we initialize $\hat{A}$ with (the entities of) all unambiguous mentions (inspired by [25]). If all mentions are ambiguous to start with, we initialize $\hat{A}$ with all candidate entities weighted by their prior probability.

Linking to NIL A mention is linked to NIL in one of two cases: (1) when the mention has no good candidate entities; and (2) when the similarity score of the entity maximizing Eq. 4 and the document is below a threshold. Both thresholds are, of course, application-specific. In future work, we will study their impact on typical EL tasks.

2.3. Supervised Walking NED

NED is the kind of problem suitable to supervised machine learning methods, as the assignment of entities to mentions is very hard to codify algorithmically. We developed a supervised algorithm that casts NED as learning to rank the candidate entities for a mention. The features are the building blocks for computing the local and semantic similarities used by our iterative algorithm, including those derived from the random walks. Our algorithm uses Boosted Regression Trees, and returns the entity with the highest score among those predicted as a correct match. Of course, one drawback of this approach is the limited availability of training data; however, for the domain of news articles, we were able to develop a robust method trained on the CoNLL dataset which proved to be highly effective across the board in our experimentations on public benchmarks and the novel benchmarks we develop in this work.

3. Semantic Similarity

This section explains how we compute the semantic similarity used in Eq. 4 to incrementally solve the NED problem. In essence, we compute the mutual information between two probability distributions, derived from random walks, which we call *semantic signatures*. In Eq. 4 we need signatures from single entities and from partial assignments consisting of many (previously disambiguated) entities. Both kinds of signatures are computed using random walks; the only difference between them is the initialization of the preference vector used in the walks.

3.1. Disambiguation Graphs

Our KBs, built from Web-scale knowledge resources such as Wikipedia, are graphs whose nodes correspond to entities and whose edges connect pairs of entities that are semantically related. More precisely, we build a graph $KB = (E, L)$ where E consists of all individual articles in Wikipedia and a link $e1 \rightarrow e2$ exists in L if either: (1) the article of e_1 has an explicit hyperlink to e_2; or (2) there is an article with hyperlinks to e_1 and e_2 within a window [4] of 500 words. In order to compute the *local* similarity (recall Eq. 4) we keep the entire article associated with each entity.

Our KB is both large and dense, rendering the cost of performing random walks prohibitive. Moreover, starting from the principle that each document belongs to a single yet unknown *topic*, and given that it mentions only a small number of entities, it is reasonable to assume that the space of entities relevant to disambiguate any given document is a small subgraph of the KB. It is on this *disambiguation graph* G consisting of all candidate entities and their immediate neighbors (recall Fig. 1) that we perform the random walks.

Selecting Candidate Entities Given a mention $m \in M$, we query an *alias dictionary* to find the KB entities that are known to be referred to as m. This dictionary is built from various resources in Wikipedia, including the anchor text in the Wikilinks [7] and redirect and

disambiguation pages.To keep the computational cost low, we proceed as follows. First, we rank candidates separately by: (1) $prior(m, e)$, the probability that the mention m refers to entity e; and (2) $context(m, e)$, the context similarity between the text describing e in the KB and the section of the document containing m. Next, we take the top-k candidates from each list (we used $k = 10$ in the experiments reported here) and discard candidate entities that are not in the union of the two lists.

As mentioned, our disambiguation graph contains not only candidate entities but also their immediate neighbors. However, because our KB is so dense, keeping all neighbors would result in a prohibitively large disambiguation graph. To further reduce the computation costs, we prune all non-candidate entities that are either (1) connected to a single candidate entity or (2) have degree below 200. (This threshold was chosen experimentally.) Candidate entities are never pruned regardless of their degree. Fig. 3 in the Experimental Section 5 shows statistics about the disambiguation graphs built in our xperimental evaluation. As one can see, even after this aggressive pruning, our disambiguation graphs have several thousand nodes for each document.

3.2. Obtaining Semantic Signatures

Let $G = (V, E)$ be a graph such that $|V| = n$. A random walk with restart is a stochastic process that repeatedly traverses the graph randomly. The starting point of these traversals is determined by an n-dimensional *preference vector*. If repeated a sufficient number of times, this process results in an n-dimensional vector corresponding to the probability distribution of reaching each vertex in the graph. We call such vectors *semantic signatures*, which we use to solve the NED problem.

Random Walks Let T be the transition matrix of G, with T_{ij} being the probability of reaching vertex e_j from vertex e_i, computed as follows:

$$T_{ij} = \frac{w_{ij}}{\sum_{e_k \in Out(e_i)} w_{ik}}$$

in which $Out(e_i)$ is the set of entities directly reachable from e_i, and w_{ij} is the weight of the edge between e_i and e_j, defined as the co-occurrence count of the corresponding entities in the KB (more details see below).

Algorithm 1 docVecInit

Input: $M = \{m_1, m_2, \ldots, m_n\}, KB = (E, L), A : M \rightarrow E$

Output: Document disambiguation vector $\mathbf{d}$

```
let n be the size of the disambiguation graph
d = 0_(n)
if A ≠ ∅ then
    for m, e ∈ A do
        d_e = 1
    end for
else
    for m ∈ M do
        for e ∈ cand(m) do
            d_e = prior(e, m) · tf idf(m)
        end for
    end for
end if
normalize d
return d
```

Let r^t be the probability distribution at iteration t, and r_i^t be the value for vertex e_i, then r_i^{t+1} is computed as follows:

$$r_i^{t+1} = \sum_{e_j \in In(e_i)} r_j^t \cdot T_{ji} \tag{5}$$

in which $In(e_i)$ is the set of entities linking to e_i.

As customary, we incorporate a random restart probability in the n-dimensional *preference vector*. Formally, the random walk process can be described as:

$$r^{t+1} = \beta \times r^t \times T + (1 - \beta) \times \mathbf{v} \tag{6}$$

where $\mathbf{v}$ is the preference vector, and $\sum v_i = 1$. We also follow the standard convention and set $\beta = 0.85$.

Semantic Signature of an Entity The semantic signature of an entity e is obtained by setting the preference vector $\mathbf{v}$ with $\mathbf{v}_e = 1$, and $\mathbf{v}_{e'} = 0$, where $e' \neq e$. The resulting distribution can be interpreted as the semantic relatedness of every node in the disambiguation graph to e.

Semantic Signature of an Assignment The semantic signature of a (partial) assignment is the result of a random walk using a preference vector $\mathbf{d}$ meant to contain the entities representing the *topic* of the document. In the iterative NED algorithm, this vector is computed from the partial assignment $\hat{A}$ and updated

each time a new mention is disambiguated (and thus added to $\hat{A}$).

This formulation does not work when no mentions have been disambiguated yet (i.e., $\hat{A} = \emptyset$), which may occur at the first step of the algorithm. In this case, we use *all* candidate entities of all mentions to represent the document, weighting them proportionally to their prominence. The procedure is given in Alg. 1; here, $prior(m, e)$ is a corpus prior derived from Wikipedia capturing how often the mention m is used to refer to entity e, while the $tf\,idf$ score of each mention is precomputed using the entire Wikipedia corpus, considering all entity aliases.

3.3. Computing the Semantic Similarity

There are several ways one can estimate the similarity of two semantic signatures (i.e., probability distributions) P and Q. One standard way of doing so is to use The Kullback-Leibler (KL) divergence:

$$D_{KL}(P \parallel Q) = \sum_i P_i \log \frac{P_i}{Q_i} \tag{7}$$

In this work, we use Zero-KL Divergence [15], a better approximation of the KL divergence that handles the case when Q_i is zero.

$$ZKL_\gamma(P, Q) = \sum_i P_i \begin{cases} \log \frac{P_i}{Q_i} & Q_i \neq 0 \\ \gamma & Q_i = 0 \end{cases} \tag{8}$$

in which γ is a real number coefficient. (Following the recommendation in [15], we set $\gamma = 20$.) The semantic similarity used in Equation 4:

$$semantic(e, \hat{A}) = \frac{1}{ZKL_\gamma(signature(e), signature(\mathbf{d}))} \tag{9}$$

Above, $\mathbf{d}$ is a vector computed from the partial assignment $\hat{A}$, as explained in Alg. 1.

4. Disambiguation Algorithms

This section gives the details of our two random-walk-based NED algorithms.

Algorithm 2 Iterative Mention Disambiguation

Input: $M = \{m_1, m_2, \ldots, m_n\}$, $KB = (E, L)$
Output: Assignment $A : M \to E \cup \{\mathsf{NIL}\}$

1: $\hat{A} = \emptyset$
2: **for** $m_i \in M$ such that $|cand(m_i)| = 1$ **do**
3: $\quad \hat{A}(m_i) = cand(m_i)$
4: **end for**

5: **for** $m_i \in M$ sorted by increasing $|ambiguity(m_i)|$ **do**
6: $\quad \mathbf{d}$ = docVecInit($M, KB, \hat{A}$)
7: $\quad max = 0$
8: $\quad$ **for** $e_j \in cand(m_i)$ **do**
9: $\quad\quad P = signature(e_j)$; $Q = signature(\mathbf{d})$
10: $\quad\quad semantic(e_j, \hat{A}) = \frac{1}{ZKL_\gamma(P, Q)}$
11: $\quad\quad score(e_j) = local(m_i, e_j) \times semantic(e_j, \hat{A})$
12: $\quad\quad$ **if** $score(e_j) > max$ **then**
13: $\quad\quad\quad e^* = e_j$; $max = score(e_j)$
14: $\quad\quad$ **end if**
15: $\quad$ **end for**

16: $\quad \hat{A}(m_i) = e^*$
17: **end for**

18: **for** $m, e \in \hat{A}$ **do**
19: $\quad$ **if** $score(e) < \theta$ **then**
20: $\quad\quad \hat{A}(m) = \mathsf{NIL}$
21: $\quad$ **end if**
22: **end for**
23: **return** $\hat{A}$

4.1. Iterative Disambiguation

Alg. 2 shows our iterative Walking NED (WNED) algorithm. Intuitively, it builds a partial assignment $\hat{A}$ by disambiguating one mention in each iteration.

It starts (lines 1–4) by assigning all *unambiguous* mentions, if any, to their respective entities. The main loop of the algorithm (lines 5–17) goes through the mentions sorted by increasing *ambiguity*, defined as the number of entities that a mention can refer to[1], computing the preference vector for the signature of the partial assignment (line 6) and greedily assigning

[1] Note this number is usually much higher than the number of *candidates* our algorithm considers, due to the pruning described in Sec. 3.1.

to the current mention the candidate entity with highest combined score (lines 7–16).

In line 11, $local(m, e)$ is computed as in Eq. 1, with $\alpha = 0.8$ (tuned experimentally).

A final step of the algorithm is to assign NIL to those mentions whose even the best candidate entity has a low score. The cut-off threshold θ is application-defined.

4.2. Disambiguation via Learning to Rank

The NED problem can be cast as an entity ranking problem for which we rank the candidates based on a score function (e.g. E.q 4), selecting the one with the highest rank. As is the case in most scenarios, the ranking is based on multiple criteria (e.g., prior probability, context similarity, semantic similarity, etc.) that may apply differently in different situations, making it hard or impossible to craft a concise algorithm that performs well in all cases. If training data is available, one can use sophisticated learning models to arrive at a more accurate ranking.

This Learning to Rank approach originates from Information Retrieval, where the methods can be divided into three groups [20]. The *pointwise* approaches consider query-document pairs as independent instances, and employ classification or regression to predict the scores of the documents (given the query) and rank them accordingly. As such, *pointwise* methods are not trained on actual rankings. On the other hand, *listwise* approaches are trained on the full ranking of all documents returned for a query. Since rankings can be hard to obtain, the *pairwise* approaches take ordered pairs of documents in which one document ranks *higher* than the other.

Our Learning-to-Rank Walking NED solution (L2R.WNED) is a *pairwise* method based on the LambdaMART method that employs the MART (Multiple Additive Regression Trees) algorithm to learn Boosted Regression Trees. It takes advantage of LambdaRank gradients to bypass the difficulty of handling non-smooth cost function introduced by most IR measures, and combines the cost function and IR metrics together to utilize the global ranking structure of documents. LambdaMART has been proven successful for solving real world ranking problems [2].

[2] An ensemble of LambdaMART rankers won Track 1 of the 2010 Yahoo! Learning to Rank Challenge

Features Although there are many useful features for ranking [36], our main goal is to establish the robustness and utility of the semantic similarity for the NED task, rather than performing exhaustive feature engineering at the risk of over-fitting. Thus we use four features, all of which are familiar in this research area: prior probability, context similarity, semantic relatedness, and *name similarity* which is measured by the N-Gram distance [18] between a mention and the canonical name of the entity.

More precisely, given a mention-entity pair $m - e$, we extract the following features: (1) prior probability $prior(m, e)$, (2) context similarity $context(m, e)$, (3) name similarity $nameSim(m, e)$, and (4) semantic relatedness $semantic(e, \mathbf{d})$ in which $\mathbf{d}$ is obtained by Alg. 1 using the the initial $\hat{A}$, computed as in lines 1–4 in Alg. 2.

Training data Using a *pairwise* ranking method, for each mention we need *ordered* pairs of entities e_1, e_2 such that e_1 is ranked *higher* than e_2. Such pairs are easy to obtain when gold standard benchmarking data is available. Given a mention m and its correct entity e in the gold standard, we obtain several other candidate entities for m and use them as the lower ranked entities.

In our experiments, we verify the system performance with two training approaches. First, we used the standard 10-fold cross-validation in which we use $1/5$ of the dataset for training and the rest for testing. Also, we experimented with using one dataset for training, while testing on other datasets. Both approaches worked well. In particular, we found that training on the fairly large CoNLL dataset led to a robust method that worked well on other benchmarks.

5. Experimental Validation

We now report on a comprehensive experimental evaluation of both WNED and the L2R.WNED methods, comparing them to the state-of-the-art. We refer to previous work [10] for the tuning of the parameters in our approach. All datasets used in this work, including the new benchmarks introduced below, as well as the accuracy results obtained with each method on each document can be downloaded from `http://bit.ly/1IcfdqS`.

We compare WNED and L2R.WNED to the state-of-the-art systems (Detailed descriptions of these systems are in the Related Work at Section 6):

Table 1
Accuracy results of all methods on the 4 public benchmarks.

Method	MSNBC			AQUAINT			ACE2004			AIDA-CoNLL		
	Accuracy	F1@MI	F1@MA	Accuracy	F1@MI	F1@MA	Accuracy	F1@MI	F1@MA	Accuracy	F1@MI	F1@MA
PRIOR	0.86	0.86	0.87	0.84	0.87	0.87	0.85	0.85	0.87	0.75	0.75	0.76
CONTEXT	0.77	0.78	0.72	0.66	0.68	0.68	0.61	0.62	0.57	0.40	0.40	0.35
Cucerzan	0.88	0.88	0.88	0.77	0.79	0.78	0.79	0.79	0.78	0.73	0.74	0.72
M&W	0.68	0.78	0.80	0.80	0.85	0.85	0.75	0.81	0.84	0.60	0.68	0.68
Han11	0.88	0.88	0.88	0.77	0.79	0.79	0.72	0.73	0.67	0.62	0.62	0.58
AIDA	0.77	0.79	0.76	0.53	0.56	0.56	0.77	0.80	0.84	0.78	0.79	0.79
GLOW	0.66	0.75	0.77	0.76	0.83	0.83	0.75	0.82	0.83	0.68	0.76	0.71
RI	0.89	0.90	0.90	0.85	0.88	0.88	0.82	0.87	0.87	0.79	0.81	0.80
WNED	0.89	0.90	0.90	0.88	0.90	0.90	0.83	0.86	0.89	0.84	0.84	0.83
L2R-CoNLL	0.91	0.92	0.92	0.85	0.87	0.87	0.85	0.88	0.90	0.89	0.89	0.89
L2R-SELF	0.91	0.92	0.91	0.88	0.90	0.90	0.85	0.88	0.89			

Table 2
Breakdown of the public benchmarks by the accuracy of the PRIOR method; #docs and #mentions are, respectively, the number of documents and the average number of mentions per document in each bracket; the number in parenthesis is the fraction of the entire benchmark covered by each bracket.

Accuracy	MSNBC		AQUAINT		ACE2004		AIDA-CoNLL	
	#docs	#mentions	#docs	#mentions	#docs	#mentions	#docs	#mentions
0.0 – 0.1	0 (0%)	0	0 (0%)	0	0 (0%)	0	5 (0.4%)	5.0
0.1 – 0.2	0 (0%)	0	0 (0%)	0	0 (0%)	0	35 (2.5%)	40.4
0.2 – 0.3	0 (0%)	0	0 (0%)	0	0 (0%)	0	29 (2.1%)	20.2
0.3 – 0.4	0 (0%)	0	0 (0%)	0	0 (0%)	0	62 (4.5%)	17.4
0.4 – 0.5	2 (10%)	51.5	0 (0%)	0	0 (0%)	0	61 (4.4%)	30.0
0.5 – 0.6	3 (15%)	45.7	0 (0%)	0	0 (0%)	0	100 (7.2%)	22.5
0.6 – 0.7	3 (15%)	37.0	1 (2%)	8.0	5 (14.3%)	10.8	164 (11.8%)	21.7
0.7 – 0.8	4 (20%)	29.8	12 (24%)	15.3	5 (14.3%)	10.8	210 (15.1%)	26.8
0.8 – 0.9	3 (15%)	53.0	16 (32%)	14.4	12 (34.3%)	8.5	267 (19.2%)	28.3
0.9 – 1.0	3 (15%)	25.0	11 (22%)	15.0	2 (5.7%)	12.0	164 (11.8%)	43.5
1.0	2 (10%)	17.5	10 (20%)	13.9	11 (31.4%)	6.4	291 (21.0%)	13.2

– Cucerzan [7]—the first global NED approach,
– M&W [25]—a leading machine learning NED solution,
– Han11 [12]—a global method that also uses random walks (on a disambiguation graph built differently than ours),
– AIDA [14]—a global method that formulates NED as a subgraph optimization problem,
– GLOW [28]—a system combining local and global, and
– RI [6]—the start-of-the-art NED system using relational inference for mention disambiguation.

We also evaluate two useful baselines: CONTEXT which chooses the candidate entity with highest textual similarity to the mention, $context(m, e)$, and PRIOR which picks the entity with highest prior probability for each mention, $prior(m, e)$. These baselines are informative as virtually all methods rely on these measures in one way or another, including ours (recall Eq. 4). Somewhat surprisingly, as shown next, not every method improves on both of them.

Measures We use the standard *accuracy, precision, recall*, and *F1*:

$$accuracy = \frac{|truth \cap result|}{|truth \cup result|}$$

$$precision = \frac{|truth \cap result|}{|result|}$$

$$recall = \frac{|truth \cap result|}{|truth|}$$

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

Where $truth$ is a ground truth assignment and $result$ is the assignment produced by the NED system.

5.1. Results on Public Benchmarks

There are four widely used public benchmarks for the NED task: (1) MSNBC [7], with 20 news articles from 10 different topics (two articles per topic) and 656 linkable mentions in total; (2) AQUAINT, compiled by Milne and Witten [25], with 50 documents and 727 linkable mentions from a news corpus from the Xinhua News Service, the New York Times, and the Associated Press; (3) ACE2004 [28], a subset of the documents used in the ACE2004 Coreference documents with 35 articles and 257 linkable mentions, annotated through *crowdsourcing*; and (4) AIDA-CoNLL [14], a hand-annotated dataset based on the CoNLL 2003 data, with 1388[3] Reuters news articles and 27817 linkable mentions.

Tab. 1 shows the results of the two baselines and all NED systems on the four public benchmarks. As customary, we report F1 aggregated across mentions (micro-averaged, indicated as **F1@MI**) and across documents (macro-averaged, **F1@MA**). For the learning to rank approaches, L2R-CoNLL refers to the method where the learning is done on the AIDA-CoNLL dataset, regardless of the test corpus, and L2R-SELF is the method where the model is trained on a fraction of the respective benchmark.

Discussion A few observations are worth making here. Among previous work, RI has the best performance across benchmarks. The disambiguation via textual similarity alone, as done by the CONTEXT baseline, leads to poor accuracy in general, especially on the more challenging AIDA-CoNLL benchmark. The PRIOR baseline, on the other hand, performs well across the board, outperforming several systems. This points to limitations in the benchmarks themselves: since they build on high quality news articles, where entities are likely to be mentioned at least once by their full name (which is easy to disambiguate with a prior alone).

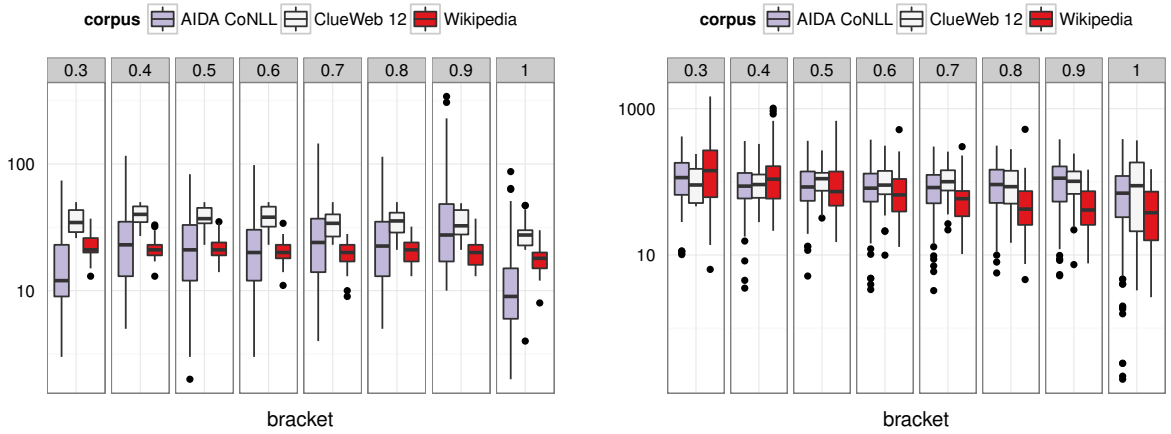
The reader will notice that virtually every method in the literature is evaluated against a baseline like PRIOR, and if one looks back to earlier works, the reported accuracy of such baseline is not nearly as high as what we report. This can be explained by the continuous cleaning process on Wikipedia—from which the statistics are derived. As we use a more recent and cleaner corpus, where the support for good and appropriate entity aliases is markedly higher than for spurious or inappropriate mentions.

With respect to WNED and L2R.WNED, both outperform all competitors on all benchmarks, with L2R.WNED performing best overall. Another observation is that training our L2R.WNED with AIDA-CoNLL data is quite effective on *all* other benchmarks, and sometimes superior to training our method with data from the specific benchmark. While not surprising (as all benchmarks come from the same domain—news), these results mean that L2R.WNED trained on AIDA-CoNLL can be seen as an effective and off-the-shelf NER system. Another general observation is that there is quite a lot of variability in the relative ordering of the previous methods across benchmarks, except for RI and our methods. This somewhat surprising lack of robustness in some systems may have been caused by over-tuning for the development benchmark, resulting in poor generalization when tested on different benchmarks.

5.2. The Need for New Benchmarks

Although the four benchmarks discussed above are useful reference points, since they are well-known and have been used for the evaluation of most NED systems, they leave a lot to be desired for a deeper and more systematic accuracy evaluation. As noted in the previous section, they are clearly biased towards popular entities, and thus, not representative of all scenarios where NED is necessary. To further illustrate the point, Tab. 2 breaks down the number of documents in each benchmark at different levels of accuracy achieved by PRIOR (i.e., the brackets are determined by the overall accuracy of all mentions in the document). As can be seen, the vast majority of docu-

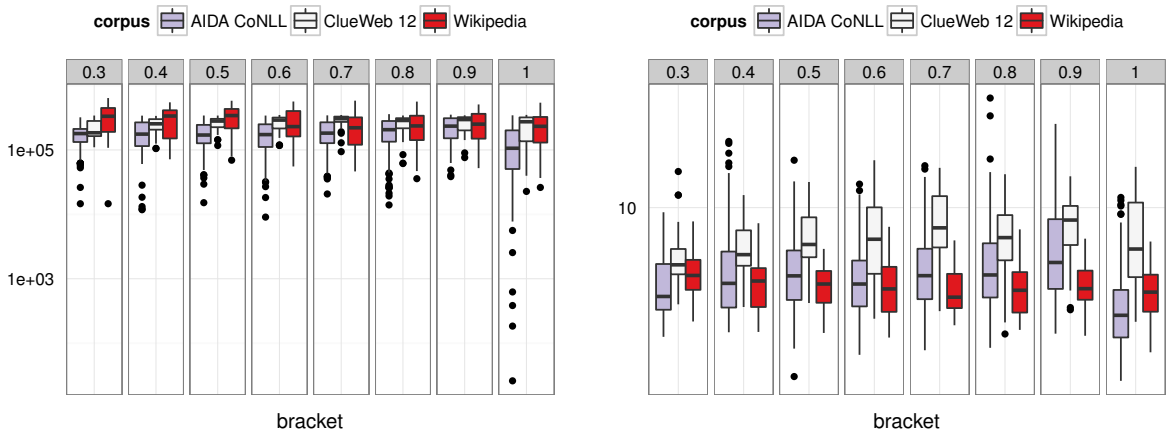
[3]The original dataset includes 5 other documents where all mentions are linked to NIL, and are therefore removed from our analysis.



(a) Mentions per document.

(b) Candidate entities per mention.

Fig. 2. Corpus statistics.



(a) Number of nodes.

(b) Average node degree.

Fig. 3. Disambiguation graph statistics.

ments in all previous benchmarks are not particularly challenging: In fact, PRIOR produces perfect results for as many as 20% of all documents of AQUAINT and AIDA-CoNLL and 31% of all documents in the case of the ACE2004 benchmark. It follows that these benchmarks are dated and unlikely to lead to further significant improvements in the area.

A desirable feature of any thorough evaluation that is not necessarily fulfilled by any of the previous benchmarks is that of *representativeness*. Namely, it would be ideal to have a mix of mentions or documents with *different levels of difficulty* in equal proportions (say on a 10-point scale from "easy" to "hard"). Without such equity, the effectiveness metrics reported in the literature (which aggregate at the mention or document level) may not be good predictors of actual performance in real applications. For instance, if a large fraction of the mentions in the benchmarks are "too easy" compared to real documents, the metrics will overestimate the true accuracy.

Of course, in order to fine tune the difficulty of the mentions and the documents in a benchmark one needs a reliable indicator of "difficulty" that can be applied to a large number of documents. Manual annotations are

clearly undesirable here, and so is *crowdsourcing*: the number of annotations needed might prove prohibitive and even if resources are not a concern this leads to a *single* benchmark (i.e., if more documents are needed, more annotations would be required).

5.3. New Benchmarks

To obtain new and balanced benchmarks, we consider the PRIOR baseline as a proxy for the true difficulty of a mention, and we obtain documents by sampling from large publicly annotated corpora such as ClueWeb and Wikipedia. In this way, we can easily collect large corpora of previously annotated documents and retain as many as needed while tuning disambiguation difficulty to the desired proportion.

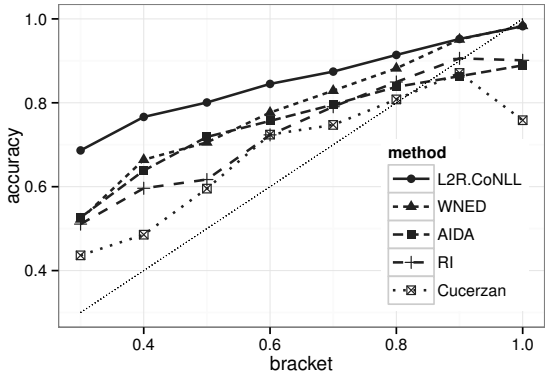
More precisely, we applied PRIOR to large samples of Wikipedia and the FACC1 annotated ClueWeb 2012 dataset. We grouped documents by the resulting average accuracy (of all mentions in the document), keeping 40 documents per bracket. Also, we further restricted the benchmarks to documents in which PRIOR achieved 0.3 or higher accuracy as we observed that below that threshold, the quality of the annotations in the ClueWeb dataset were very low. Finally, we controlled the number of mentions per document: for the Wikipedia corpus we have the mean at 20.8 ($\sigma = 4.9$) and for the ClueWeb 2012 we have the mean at 35.5 ($\sigma = 8.5$).

Fig. 2 shows statistics about our benchmarks: namely, the average number of mentions per document (Fig. 2a) and the average number of candidates per mention (Fig. 2b). For the sake of comparison, we also report the same statistics from the documents in the AIDA-CoNLL dataset in the respective accuracy brackets. Fig. 3 shows statistics about the disambiguation graphs built by our method (which, as discussed in Section 4, depend both on the number of candidates per mention and on how densely connected they are in the entity graph). Fig. 3a shows the average graph sizes (in terms of number of nodes) and Fig. 3b shows the average node degree.

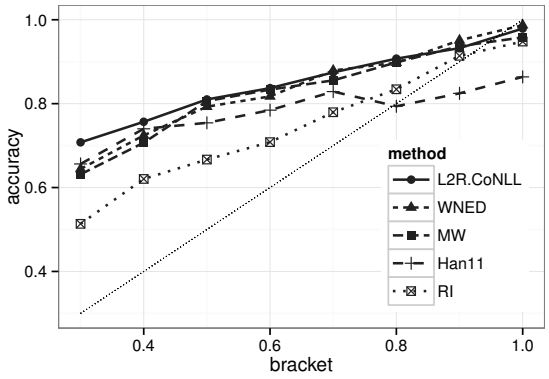
As one can see, the variability in our datasets is considerably smaller compared to AIDA-CoNLL, particularly when it comes to clear outliers (indicated as individual dots in the charts).

5.4. Results on the New Benchmarks

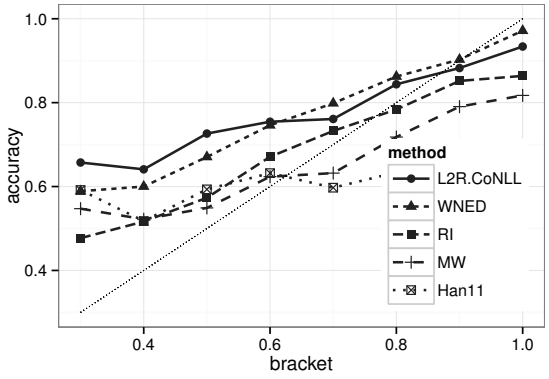
Fig. 4 shows the accuracy on the new benchmarks. We plot the accuracy of the best performing methods



(a) AIDA-CoNLL†.



(b) Wikipedia.



(c) ClueWeb 12.

Fig. 4. Average accuracy of the top-5 methods on the Wikipedia, Clueweb 12, and AIDA-CoNLL datasets grouped by the accuracy of the PriorProb baseline.

Table 3
Average per-bracket accuracy on large-scale benchmarks. Brackets for AIDA-CoNLL are as in Tab. 2; only those brackets with PRIOR accuracy 0.3 or higher were used.

Method	AIDA-CoNLL	Wikipedia	ClueWeb 12
PRIOR	0.57	0.56	0.57
CONTEXT	0.39	0.59	0.42
Cucerzan	0.68	0.66	0.60
M&W	0.58	0.83	0.65
Han11	0.57	0.78	0.61
AIDA	0.75	0.63	0.59
GLOW	0.61	0.69	0.57
RI	0.74	0.75	0.68
WNED	0.79	0.84	0.77
L2R.WNED	0.85	0.85	0.78

for each of the difficulty brackets (defined by the accuracy of the PRIOR baseline). For clarity, we plot the accuracy of the best 5 approaches. For comparison, we also show the accuracy of each method on the AIDA-CoNLL benchmark. For the Wikipedia and ClueWeb benchmarks, each bracket corresponds to exactly 40 documents, whereas for the AIDA-CoNLL dataset the brackets are as in Tab. 2. For convenience, a diagonal dotted line whose area under the curve (AUC) is 0.5 (loosely corresponding to the PRIOR baseline) is also shown. Methods consistently above that line are expected to outperform the PRIOR baseline in practice. Tab. 3 shows the average accuracy of every method across brackets, corresponding to the AUC in Fig. 4.

A few observations are worth mentioning here. First, the two new benchmarks complement the AIDA-CoNLL benchmark: overall, the Wikipedia benchmark is easier than AIDA-CoNLL, while the ClueWeb 12 is harder. Second, as before, the RI method performed very well, although not as dominantly as in the four public benchmarks. It also seems that the previous supervised methods tend to over-perform on their own development datasets (Wikipedia for M&W and CoNLL for AIDA).

Our L2R.WNED and WNED systems outperform all other competitors across all benchmarks, performing much better on the more "difficult" cases (i.e., in lower brackets). In concrete terms, WNED and L2R.WNED exhibit, on average, 21% and 26% relative gain in accuracy over the previous methods (excluding the baselines) on the three benchmarks combined, which is significant. Given that our development and tuning was done with a subset of the AQUAINT,

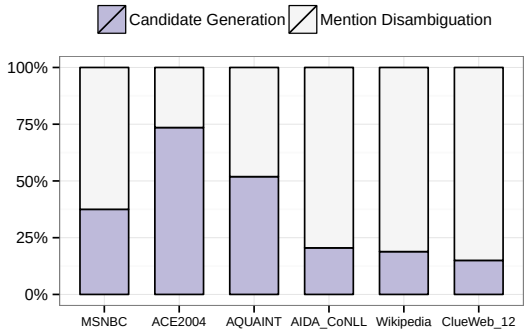


Fig. 5. Breakdown of errors by WNED across benchmarks; for AIDA-CoNLL, Wikipedia and ClueWeb 12, the errors are estimated from a sample.

MSNBC and ACE2004, the strong results of WNED and L2R.WNED demonstrate the robustness and generality of our approach.

5.5. Qualitative Error Analysis

We now look at the kinds of errors made by our method. To do so, we manually inspected every error for the smaller MSNBC, AQUAINT, and ACE2004 datasets, and analyzed 20 errors randomly picked in each bracket for the larger ones.

The first observation is that in the older benchmarks, a larger fraction of the errors in our method happen in the candidate selection phase, as illustrated in Fig. 5. On average, 54% of the errors in the smaller benchmarks are due to candidate selection (compared to 18% in the other ones). This reinforces the hypothesis that the entities mentioned in these older benchmarks are easier to disambiguate[4].

Below we discuss prototypical errors in each of the phases.

Errors during Candidate Selection

Incorrect Co-reference Resolution We employ a co-reference resolution algorithm in our text processing pipeline to increase recall. Due to the heuristic nature of the algorithm, it is possible that distinct named enti-

[4]Note: given that the smaller benchmarks are older, it is unlikely they mention more *out-of-KB* entities than the other ones, especially AIDA-CoNLL. Thus, because we use the same standard NLP pipeline for processing the inputs across all benchmarks, the discrepancy in Fig. 5 can only mean that our method is successful on most of the (*in-KB*) entities in the older benchmarks, making them "easier" than the other ones.

ties are incorrectly deemed to be the same. For example, in the sentence

> "Time Warner stagnated for five years after it was created in 1990 by the merger of Time and Warner."

the entity "Time" at the end of the sentence is incorrectly resolved to "Time Warner", leading to an error. About 1% of the errors (1.5% in the harder benchmarks) are due to incorrect resolution of named entities.

Incomplete Alias Dictionary Currently, we disambiguate only those mentions corresponding to an alias from Wikipedia, leading to problems in sentences like

> "Thirteen miners were trapped inside the Sago Mine near Buckhannon, W. Va."

In this case we miss the abbreviation "W. Va." for West Virginia. This kind of error was noticeably more common in the easier benchmarks (accounting for 30% of the errors in the ACE2004 dataset). In the AIDA-CoNLL benchmark only 2% of the errors are due to this problem.

Aggressive Pruning Another source of error by our method is pruning the correct entity from the disambiguation graph (recall Sec. 3.1). For example, in sentence

> "A state coordinator for the Florida Green Party said she had been ..."

the correct entity (the *Green Party of Florida*) is pruned due its low prior but could probably be correctly resolved given the mention to *Florida* in the same sentence. Instead, WNED links the mention to the US Green Party. Of course, pruning is done to reduce the cost of the random walks, and future algorithmic improvements can alter this trade-off.

Errors during Mention Disambiguation

These are errors where the correct entities according to the ground truth were selected as the candidates but not chosen during the mention disambiguation phase by our algorithm.

Lack of Application Domain We observed that most of the errors associated with locations happen because the documents in most benchmarks are news articles that start with the location of the news source braking the news (e.g., New York Times documents always start with a mention to New York). More often than not, such locations are totally unrelated to the topic of the documents and other mentions in the document, breaking the global coherence assumption. These errors, which can be easily fixed via pre-processing, accounts for 5% of the mistakes of our algorithm in the MSNBC and AIDA benchmarks and 2% across all benchmarks.

Need for Deeper Text Analysis There are of course very hard disambiguation cases where a deeper understanding of the text would be needed for a successful algorithmic approach. One example is the sentence:

> "Maj. Gen. William Caldwell, a U.S. military spokesman, told reporters that ..."

In this case there are two candidates with the same name and high military rank, thus being semantically related to the document and confusing the algorithm. In this case, extraneous facts about the candidates, unrelated to the text itself, could be used for disambiguating the mention. For instance the candidate incorrectly chosen by our algorithm died in the 1820s while the correct candidate was still alive at the time the benchmark article was written. Given the document states the facts as current news, the incorrect candidate could have been pruned out.

Questionable Errors

We argue that in many cases, our algorithm (as well as other systems) chose an entity that are considered erroneous by the ground truth but that would be acceptable to a human judge. For example, in the sentence:

> "Coach Saban said the things Crimson Tide fans most wanted to hear."

our system links "Crimson Tide" in the sentence to the *Alabama Crimson Tide football*, which is the *men's* varsity football team of the university while the ground truth refers to *Alabama Crimson Tide* which corresponds to both the men's and women's teams. We found that about 17% of the errors are in this category, with a higher prevalence in the harder benchmarks (21%). Tab. 4 lists many other similar errors, where a case can be made that the ground-truth itself is probably too strict.

Impact of the Greedy Approach

Given the iterative WNED is a greedy algorithm, it is interesting to see how an erroneous disambiguation decision influences future ones, especially in the very first round. In all benchmarks, we found one error in the first round among all the errors in MSNBC,

Table 4
Questionable disambiguation errors

Mention	WNED Suggestion	Ground Truth
Iraqi	Iraqi people	Iraq
Hungarian	Hungary	Hungarian people
executives	Corporate title	Chief executive officer
Russian	Russian language	Russians
Iranian	Iranian people	Iran
Greek	Greek language	Ancient Greece
civil war	American Civil War	Civil war
broadcaster	Presenter	Broadcasting

AQUAINT and ACE2004 datasets[5], and less than eight errors from the random samples in the other 3 benchmarks. In all cases, the first error did not prevent the algorithm from correctly disambiguating other mentions.

As for the initialization step, we found that most documents in our benchmarks do have unambiguous mentions available, and most of them are correctly linked to the true entity[6]. In MSNBC, we have 2 errors from the unambiguous mentions, *New york stock exchange* and *NYSE*, both are linked to *New york mercantile exchange*. This error barely affects the semantic signature since they are still stock related entities. There are 5 such errors in AQUAINT, and 1 error in ACE2004, all of which have little affect on the linking results of other mentions in the same document.

Finally, we found that most other errors happened after 5 iterations, when the document disambiguation vector already captures the topic fairly well. These errors are for mentions that are not semantically related to other mentions in the document, or simply due to the disproportionately high priors favoring (incorrectly) head entities.

6. Related Work

Earlier work on Entity Linking disambiguated each mention in isolation using a compatibility function to approximate the likelihood of an entity being the referent entity for a mention, and treated entity linking as a ranking problem which chose the candidate with the highest compatibility. In these approaches, mentions and entities are represented as feature vectors and vector similarity measures are used to estimate their compatibility. The most common local features include lexical features such as bag-of-words or named entities from surrounding context and statistical features such as the prior probability of entities given a mention from a knowledge base. Unsupervised approaches [2,3] commonly use the cosine similarity of feature vectors to measure the compatibility, while supervised approaches [22,25,35,37,34,8] exploit various classifiers trained on labeled datasets (often derived from Wikipedia) to predict the compatibility. By restricting themselves to local features, these methods suffer from the data sparsity problem. Moreover, individually linking mentions does not take into account the inherent semantic coherence among them.

More recent EL systems follow the global coherence hypothesis and take into account semantic relations between mentions and entities, employing various measures of semantic relatedness. Most of these approaches assume that mentions in a document are semantically coherent around the topic of the document, and thus cast the entity linking as an optimization problem aiming at finding the assignment with maximum semantic coherence. Cucerzan [7] measures the global coherence using Wikipedia categories. Milne and Witten (M&W) [25] use directly connected entities to represent each entity and measure relatedness using normalized Google distance. Kulkarni et al. [19] also use the M&W semantic relatedness measure and formalize the problem as an integer linear programming problem to find the assignment collectively. Ratinov et al. [28] add the PMI (Pointwise Mutual Information) of entities into their SVM classifier for entity linking. To use more fine-grained semantics such as relations between mentions, Cheng and Roth [6] formalize the EL as an integer linear programming problem with relations as constraints, and find the assignment to meet the relational constraints. Cai et al. [4] measure the semantic relatedness using the co-occurrence of entities from a link-enriched Wikipedia corpus. AIDA [14] formalizes the EL problem as a dense subgraph problem, which aims to find a dense subgraph from a mention-entity graph such that the subgraph contains all mentions and one mention-entity edge for each mention according to their definition of graph density. Han and Sun [11] combine the local compatibility and global coherence using a generative entity-topic model to infer the underlying referent entities. Also through a generative graphical model, Li et al. [21] propose to mine additional information for en-

[5] A mention to the *USS Cole* which should have been linked to *USS Cole (DDG-67)*, was linked to *USS Cole bombing*.

[6] Recall (Sec. 4) we initialize the document disambiguation vector with unambiguous mentions when available.

tities from external corpus, which can help improve the effectiveness of entity linking, especially for entities with rare information available in the knowledge base.

The approach closest to ours is that of Han et. al [12], which uses a random walk with restart to obtain a vector of relevance for all candidates of mentions, and considers the relevance value in the vector to be the relatedness between a mention and its candidate. Like other semantic relatedness measures, their measure can only compute the relatedness between two entities. Instead, we use a unified semantic representation for both documents and entities. As a result, we can measure the coherence between entities and between entities and documents in a unified way. Also our representation can capture the semantics of unpopular entities, which makes our EL approach more robust for datasets with less popular entities. The idea of using random walk with restart has been applied on graphs constructed from the WordNet [23], with the stationary distribution to represent the semantics of words. It has been shown to be effective in the word similarity measurement [1,15], and word sense disambiguation [26]. However, we are not aware of any previous work using the stationary distribution from random walk with restart to represent entities and documents in entity linking.

When it comes to learning to rank, Zheng et.al [36] applied learning to rank approaches on the entity linking task and demonstrated its superior effectiveness over most state-of-the-art algorithms. Their results showed that the *listwise* method ListNet [5] performed better than the *pairwise* approach Ranking Perceptron [29]. However, we found that the pairwise approach LambdaMART [32] achieved the best performance on our datasets among most learning to rank algorithms.

7. Conclusion

We described a method for named entity disambiguation that combines lexical and statistical features with *semantic* signatures derived from random walks over suitably designed disambiguation graphs. Our semantic representation uses more relevant entities from the knowledge base, thus reducing the effect of feature sparsity, and results in substantial accuracy gains. We described a hand-tuned greedy algorithm as well as one based on learning-to-rank. Both outperform the previous state-of-the-art by a wide margin. Moreover, we showed that our L2R.WNED algorithm trained on the standard AIDA-CoNLL corpus is quite robust across benchmarks.

Moreover, we demonstrated several shortcomings of the existing NED benchmarks and described an effective way for deriving *better* benchmarks and described two new such benchmarks based on web-scale annotated corpora (ClueWeb12 and Wikipedia). Our benchmark generation method can be tuned to produce "harder" or "easier" cases as desired. Overall, the benchmarks we describe complement the largest currently available public benchmark. Our experimental evaluation compared our methods against six leading competitors and two very strong baselines, revealing the superiority and robustness of our entity linking system in a variety of settings. Our method was particularly robust when disambiguating unpopular entities, making it a good candidate to address the "long tail" in Information Extraction.

Future work A number of opportunities for future work exist. Sec. 5.5 lists several ideas for algorithmic improvements that can lead to better NED systems in the future. Also, while the new benchmarks described here can be used for both accuracy and scalability tests (as one can easily obtain large quantities of documents from ClueWeb12 and Wikipedia), further work is needed in helping the design and verification of ground-truths.

Our algorithms require multiple random walk computations, making it time consuming if implemented naively (as in our current implementation). On a standard entry-level server, the average time to disambiguate a document in our benchmarks (usually with less than 100 mentions) is in the order of a few minutes. Therefore, designing proper system infrastructure with the appropriate indexes and/or parallel computing infrastructure to optimize these computations would be interesting. Moreover, other state-of-the-art systems perform other expensive operations as well, such as accessing the Web or querying large relational databases. Therefore, designing objective and fair benchmarks for comparing these different approaches in terms of both accuracy and performance would be of great value to the community.

References

[1] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa, *A study on similarity and relatedness using distributional and wordnet-based approaches*, in *HLT-NAACL*, 2009, pp. 19–27.

[2] A. Bagga and B. Baldwin, *Entity-based cross-document coreferencing using the vector space model*, in *COLING-ACL*, 1998, pp. 79–85.

[3] R. C. Bunescu and M. Paşca, *Using encyclopedic knowledge for named entity disambiguation*, in *EACL*, 2006.

[4] Z. Cai, K. Zhao, K. Q. Zhu, and H. Wang, *Wikification via link co-occurrence*, in *CIKM*, 2013, pp. 1087–1096.

[5] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, *Learning to rank: from pairwise approach to listwise approach*, in *ICML*, 2007, pp. 129–136.

[6] X. Cheng and D. Roth, *Relational inference for wikification*, in *EMNLP*, 2013, pp. 1787–1796.

[7] S. Cucerzan, *Large-scale named entity disambiguation based on wikipedia data*, in *EMNLP-CoNLL*, 2007, pp. 708–716.

[8] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin, *Entity disambiguation for knowledge base population*, in *COLING*, 2010, pp. 277–285.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[10] Z. Guo and D. Barbosa, *Robust entity linking via random walks*, in *CIKM*, 2014, pp. 499–508.

[11] X. Han and L. Sun, *An entity-topic model for entity linking*, in *EMNLP-CoNLL*, 2012, pp. 105–115.

[12] X. Han, L. Sun, and J. Zhao, *Collective entity linking in web text: a graph-based method*, in *SIGIR*, 2011, pp. 765–774.

[13] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum, *Kore: keyphrase overlap relatedness for entity disambiguation*, in *CIKM*, 2012, pp. 545–554.

[14] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum, *Robust disambiguation of named entities in text*, in *EMNLP*, 2011, pp. 782–792.

[15] T. Hughes and D. Ramage, *Lexical semantic relatedness with random graph walks*, in *EMNLP-CoNLL*, 2007, pp. 581–589.

[16] H. Ji and R. Grishman, *Knowledge base population: Successful approaches and challenges*, in *ACL*, 2011, pp. 1148–1158.

[17] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. Prentice Hall, 2008.

[18] G. Kondrak, *N-gram similarity and distance*, in *String Processing and Information Retrieval*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, vol. 3772, pp. 115–126.

[19] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti, *Collective annotation of wikipedia entities in web text*, in *KDD*, 2009, pp. 457–466.

[20] H. Li, *Learning to Rank for Information Retrieval and Natural Language Processing*, ser. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2011.

[21] Y. Li, C. Wang, F. Han, J. Han, D. Roth, and X. Yan, *Mining evidences for named entity disambiguation*, in *KDD*, 2013, pp. 1070–1078.

[22] R. Mihalcea and A. Csomai, *Wikify!: linking documents to encyclopedic knowledge*, in *CIKM*, 2007, pp. 233–242.

[23] G. A. Miller, *Wordnet: A lexical database for english*, *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[24] D. Milne and I. H. Witten, *An effective, low-cost measure of semantic relatedness obtained from wikipedia links*, in *WIKIAI*, 2008.

[25] D. Milne, I. H. Witten, *Learning to link with wikipedia*, in *CIKM*, 2008, pp. 509–518.

[26] M. T. Pilehvar, D. Jurgens, and R. Navigli, *Align, disambiguate and walk: A unified approach for measuring semantic similarity*, in *ACL*, 2013, pp. 1341–1351.

[27] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, and A. Kirilov, *Kim-a semantic platform for information extraction and retrieval*, *Natural language engineering*, vol. 10, no. 3-4, pp. 375–392, 2004.

[28] L.-A. Ratinov, D. Roth, D. Downey, and M. Anderson, *Local and global algorithms for disambiguation to wikipedia*, in *ACL*, 2011, pp. 1375–1384.

[29] L. Shen and A. K. Joshi, *Ranking and reranking with perceptron*, *Machine Learning*, vol. 60, no. 1-3, pp. 73–96, 2005.

[30] S. Singh, A. Subramanya, F. Pereira, and A. McCallum, *Wikilinks: A large-scale cross-document coreference corpus labeled via links to wikipedia*, University of Massachusetts, Tech. Rep. UM-CS-2012-015, 2012.

[31] H. Tong, C. Faloutsos, and J.-Y. Pan, *Fast random walk with restart and its applications*, in *ICDM*, 2006, pp. 613–622.

[32] Q. Wu, C. J. C. Burges, K. M. Svore, and J. Gao, *Adapting boosting for information retrieval measures*, *Inf. Retr.*, vol. 13, no. 3, pp. 254–270, 2010.

[33] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum, *Natural language questions for the web of data*, in *EMNLP*, 2012, pp. 379–390.

[34] W. Zhang, Y. C. Sim, J. Su, and C. L. Tan, *Entity linking with effective acronym expansion, instance selection, and topic modeling*, in *IJCAI*, 2011, pp. 1909–1914.

[35] W. Zhang, J. Su, C. L. Tan, and W. Wang, *Entity linking leveraging automatically generated annotation*, in *COLING*, 2010, pp. 1290–1298.

[36] Z. Zheng, F. Li, M. Huang, and X. Zhu, *Learning to link entities with knowledge base*, in *HLT-NAACL*, 2010, pp. 483–491.

[37] Y. Zhou, L. Nie, O. Rouhani-Kalleh, F. Vasile, and S. Gaffney, *Resolving surface forms to wikipedia topics*, in *COLING*, 2010, pp. 1335–1343.