AIDA-light: High-Throughput Named-Entity Disambiguation

Dat Ba Nguyen
Max Planck Institute for Informatics
datnb@mpi-inf.mpg.de

Johannes Hoffart
Max Planck Institute for Informatics
jhoffart@mpi-inf.mpg.de

Martin Theobald
University of Antwerp
martin.theobald@uantwerpen.be

Gerhard Weikum
Max Planck Institute for Informatics
weikum@mpi-inf.mpg.de

ABSTRACT

To advance the Web of Linked Data, mapping ambiguous names in structured and unstructured contents onto knowledge bases would be a vital asset. State-of-the-art methods for Named Entity Disambiguation (NED) face major tradeoffs regarding efficiency/scalability vs. accuracy. Fast methods use relatively simple context features and avoid computationally expensive algorithms for joint inference. While doing very well on prominent entities in clear input texts, these methods achieve only moderate accuracy when fed with difficult inputs. On the other hand, methods that rely on rich context features and joint inference for mapping names onto entities pay the price of being much slower.

This paper presents AIDA-*light* which achieves high accuracy on difficult inputs while also being fast and scalable. AIDA-*light* uses a novel kind of two-stage mapping algorithm. It first identifies a set of "easy" mentions with low ambiguity and links them to entities in a very efficient manner. This stage also determines the thematic domain of the input text as an important and novel kind of feature. The second stage harnesses the high-confidence linkage for the "easy" mentions to establish more reliable contexts for the disambiguation of the remaining mentions. Our experiments with four different datasets demonstrates that the accuracy of AIDA-*light* is competitive to the very best NED systems, while its run-time is comparable to or better than the performance of the fastest systems.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

Keywords

Entity Linking, Named Entity Disambiguation, Joint Inference, Scalability

LDOW2014, April 8, 2014, Seoul, Korea.

1. INTRODUCTION

1.1 Motivation and Approach

The Web of Linked Data [11] provides a wealth of data and knowledge sources that are richly interlinked at the entity level. The data itself is uniformly represented as RDF triples. However, RDF data can also be embedded in Web pages with surrounding text, and the Web also contains a huge amount of semi-structured contents like user-created HTML tables within Web pages. To further grow the Web of Linked Data and advance its value for semantic applications, we propose to consider also unstructured and semi-structured contents, detect names of embedded entities, and link these to knowledge bases like DBpedia, Freebase, or YAGO.

In computational linguistics, the problem of mapping ambiguous *entity mentions* (or just *mentions* for short) occurring in natural-language texts onto a set of known target entities is referred to as *Named Entity Disambiguation* (NED). State-of-the-art NED methods [3] face major tradeoffs regarding output accuracy vs. run-time efficiency and scalability. Fast methods, like TagMe2 [7], Illinois Wikifier [20] or DBpedia Spotlight [17, 5] use relatively simple contextual features, and avoid combinatorially expensive algorithms that use collective inference for jointly mapping all mentions in a text. These methods perform very well on straightforward input texts about prominent entities; however, on very difficult inputs with highly ambiguous names and mentions of long-tail entities, the accuracy of these methods is only moderate. On the other hand, more sophisticated methods that rely on rich contextual features such as key phrases and on joint-inference algorithms, such as AIDA [14, 12], tend to be much slower and thus face scalability limitations.

Based on our prior experience with AIDA, this paper reconsiders the design space of possible context features and mapping functions, in order to develop an NED system that achieves both high throughput and high output accuracy. We present the AIDA-*light* system that performs very well even on difficult input texts, while also being as efficient as the fastest methods. AIDA-*light* employs simpler features than AIDA, thus reducing computational cost, and also adds a new kind of feature: *thematic domains* like pop music, football, skiing, etc. A major novelty of AIDA-*light*

is its two-stage mapping algorithm: First, our method identifies a set of "easy" mentions which exhibit low ambiguity and can be mapped onto entities with high confidence. The second stage harnesses this intermediate result by inferring the primary domain of the input text and uses this as an additional feature to establish more reliable contexts for the remaining, more difficult, mentions.

Running Example. Consider the following natural-language input sentence as a running example:

"Under Fergie, United won the Premier League title 13 times."

It is obvious that the above mentions of "*Fergie*" and "*United*", with hundreds of candidate entities, are more ambiguous than the mention of "*Premier League*", which exhibits just a few candidate entities when using a DBpedia/Wikipedia-based set of target entities. So the risk of disambiguating the mention "*Premier League*" incorrectly is fairly low. Moreover, once we fix this mention to the entity `Premier_League`, the English professional football league, we can infer that the above text snippet likely (and primarily) relates to the general domain of "**Football**". Within the context of this domain, it thus becomes easier to map "*United*" to the entity `Manchester_United_F.C.` (rather than to `United_Airlines`) and "*Fergie*" to the entity `Alex_Ferguson` (rather than to `Fergie_(singer)` or `Sarah,_Duchess_of_York`).

1.2 State of the Art

Bunescu and Pasca [2] were the first to investigate the usage of a Wikipedia-based set of target entities by defining a similarity measure that compares the textual context of a mention to the Wikipedia categories associated with each entity candidate. This initial idea was later extended by using richer features for the similarity comparison [4, 10, 18]. In particular, instead of using the similarity function directly, Milne [18] introduced a supervised learning step to better estimate the weights of the underlying feature functions from labeled training data. Milne [18] also added a notion of semantic relatedness between candidate entities among unambiguous mentions. However, these approaches are still limited to mapping each mention individually and iteratively. That is, their disambiguation objective does not consider any mutual (i.e., joint) dependencies among the possible target entities.

Collective-learning models perform a joint mapping of all mentions [16, 14, 12] onto their matching target entities "in one shot". These approaches typically yield the highest NED accuracy for difficult input texts. Because of the high combinatorial cost for the joint mapping (which typically leads to NP-hard problems), finding an exact solution to this objective is prohibitive. To relax the problem—and due to sparseness of the available training data—the above methods build a graph with edges or "factors" for mention-entity pairs and entity-entity pairs. Even when considering only pairs of target entities, finding the most likely mapping over the joint probability distribution of all mappings in a probabilistic interpretation (a form of Maximum-A-Posteriori (MAP) inference [20]) remains an NP-hard optimization problem [4]. Therefore, various approximations and variations have been developed [14, 20, 12].

In addition, some of these high-accuracy NED methods also face high computational cost due to their rich contextual features, most notably, key phrases of entities, entity-entity relatedness measures, and corresponing statistics [14, 20]. This prevents such techniques from being applied to Web-scale corpora.

Fast NED methods, on the other hand, use simpler features, for example, only characteristic words and precomputed statistics. By the more compact feature space, they also avoid interacting with databases and instead use customized in-memory data structures. The probably fastest publicly available NED systems are TagMe [7] and Spotlight [17, 5]. TagMe uses a light-weight form of coherence by averaging relatedness scores over all candidate entities for a mention that co-occurs with a given mention. These scores are precomputed for NED throughput. Spotlight uses word-level statistics to estimate the probability of an entity given the context of a mention. This is combined, in a probabilistic manner, with prior distributions on entity names and entity prominence. Compared to full-fledged collective-inference methods, this approach is much more light-weight by avoiding key phrases and relatedness measures over entity pairs.

1.3 Contributions

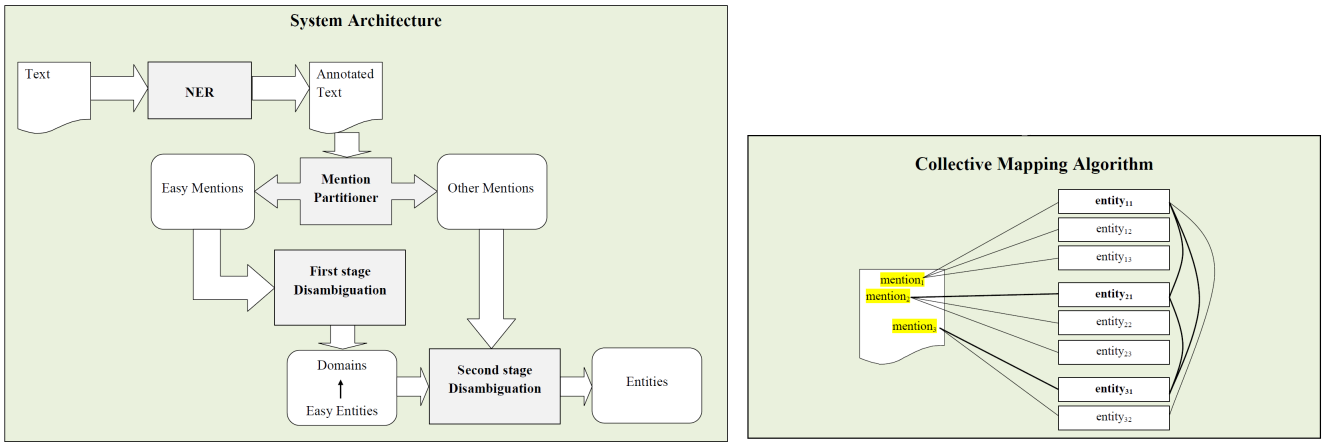
We summarize the novel aspects of this work as follows.

- AIDA-*light* makes a judicious choice of *contextual features* to compute pairwise *mention-entity* and *entity-entity similarities*. These feature functions allow us to store the underlying context statistics with low memory footprint, which contributes to faster processing.
- AIDA-*light* is the first approach that harnesses a thematic *domain hierarchy* to capture measures for *domain-entity* and *entity-entity coherence*.
- AIDA-*light* employs a novel *two-stage algorithm* in which we determine the "easy and low-cost" mappings first. By doing so, we reduce both the complexity and difficulty of the second-stage disambiguations.
- AIDA-*light* is a complete system for NED, which is orders of magnitude faster than AIDA while achieving comparable output quality. Our experimental comparisons against AIDA, Spotlight, and Illinois Wikifier show that AIDA-*light* is consistently close to the best competitor (or is even the best) in terms of both run-time and accuracy.

2. SYSTEM ARCHITECTURE

As shown in Figure 1, AIDA-*light* focuses on the disambiguation of named-entity mentions in natural-language input texts such as news articles, Web pages and Wikipedia articles. We assume that the mentions in the input text have been recognized and annotated by an *NER tool*, such as the Stanford tagger [8] or a similar annotation tool.

In its first processing stage, AIDA-*light* divides the sequence of mentions as they have been marked in the input text into two parts using its *mention partitioner* component. This form of partitioning is based on the number of possible candidate entities each of the mentions can be mapped to in the background KB. At this stage, we thus focus on detecting *easy mentions*, i.e., those mentions with very few candidate entities, which exhibit a low level of ambiguity and thus can be mapped to the correct target entity with high confidence. Once these easy mentions are fixed to their respective sets of target entities, the more general *domain* of the input text is discovered via the entity-domain dictionary. For example, if there are multiple occurrences of football clubs in the text snippet, then the text likely relates to the domain "**Football**". The chosen target entities



(a) The system architecture.

(b) The collective mapping algorithm [14].

Figure 1: The overview of AIDA-*light* .

as well as their domains are added to the mention context of the remaining, still ambiguous mentions. Finally, the detailed similarities (e.g., mention-entity context similarities) are recomputed for the second stage of disambiguation. At this stage, our *collective mapping algorithm* performs the disambiguation of the remaining mentions using the set of feature functions described in Section 3.

Figure 2 illustrates how AIDA-*light* processes our earlier example sentence "*Under Fergie, United won the Premier League title 13 times.*" After the preprocessing steps (including tokenization and NER) are performed, a sliding window is contiguously moved over the resulting token and mention sequences, thus keeping the current mention that is about to be disambiguated at the center of the window. Hence, for each further mention that we disambiguate, this window is moved forward. Both surrounding tokens and mentions within this window are taken into the context of this central mention. We remark that in this example, some intermediate results are hidden. For example, the tokenizer (e.g. Stanford tokenizer) splits "Premier League" into two tokens "Premier" and "League"; however, these two tokens create a mention based on the results of NER, and thus AIDA-*light* only considers this phrase as a single unit. As shown in the figure, the ambiguities of the three mentions in our example sentence vary substantially. The mention "*Premier League*" is mapped to only very few candidate entities (even just to one in this example). Thus, we obtain a high-confidence disambiguation of this mention to the entity `Premier_League`. Thanks to this entity and the *domain hierarchy*, which maps an entity to its respective domains (such as football, politics, etc.), AIDA-*light* is able to predict that the text relates to the domain "**Football**". As a result, it is now much easier to choose also the correct entities for the other mentions, for example, to map the mention of "*United*" to the entity `Manchester_United_F.C.` rather than to `United_Airlines`.

3. FEATURE MODEL

As mentioned before, AIDA-*light* employs a combination of various feature functions to perform NED via a two-stage mapping algorithm. Specifically, we investigate the usage of seven different feature functions that indicate either

1) the likelihood of a *mention* M within its given textual context to represent a *named entity* E out of a set of known entities $\mathbb{E}$ in our background KB, or
2) the coherence (i.e., the semantic similarity) of a pair of *entity candidates* E_1,$E_2 \in \mathbb{E}$ with respect to this KB.

These feature functions, which are defined in detail in the following subsections, partly consist of simplified context features used in AIDA (e.g., using sets of plain key tokens instead of sets of key phrases as in [14]). In addition, we employ two novel features, which are based on a predefined domain hierarchy and the Wikipedia category system with which the candidate entities are associated.

3.1 Preprocessing Steps

Token Sequences. We start by tokenizing a given input text into a *token sequence* $\mathbb{T} = \langle T_1, \ldots, T_t \rangle$ of white-space and punctuation separated terms. All tokens used as input for our context features are stemmed using the Porter stemmer. This token sequence serves as input to the various context features employed by the disambiguation algorithm.

Mention Sequences. We assume that a subset $\mathbb{T}' \subseteq \mathbb{T}$ of these tokens has been annotated as a sequence of mentions $\mathbb{M} = \langle M_1, \ldots, M_m \rangle$ by an auxiliary NER tool (using, e.g., the Stanford [8] or Illinois [19] taggers). Formally, we designate an *annotator function* $NER : \mathbb{T} \rightarrow \mathbb{M}$ for this purpose, which allows us to obtain the mention $M_j := NER(T_i)$ that is associated with a given token T_i (or *null* if T_i is not associated with any mention).

Mention Dictionary. In addition to the NER function, we also assume the availability of a *dictionary function* $Dict : \mathbb{M} \rightarrow 2^{\mathbb{E}}$ that identifies the set of candidate entities $\mathbb{E}_j \subseteq \mathbb{E}$ for a given mention $\mathbb{E}_j := Dict(M_j)$ to be provided as input to our NED algorithm. Just like AIDA [14, 12], AIDA-*light* employs the *means* relation of YAGO2[1] to identify this set of candidate entities for a (possibly ambiguous) mention M_j. The entries in the dictionary were extracted from link anchors, disambiguation pages and redirection links in Wikipedia.

Dictionary Augmentation via LSH. In order to improve recall over a variety of input texts, including Web

[1] YAGO2.4.0 - `http://www.mpi-inf.mpg.de/yago-naga/yago/`

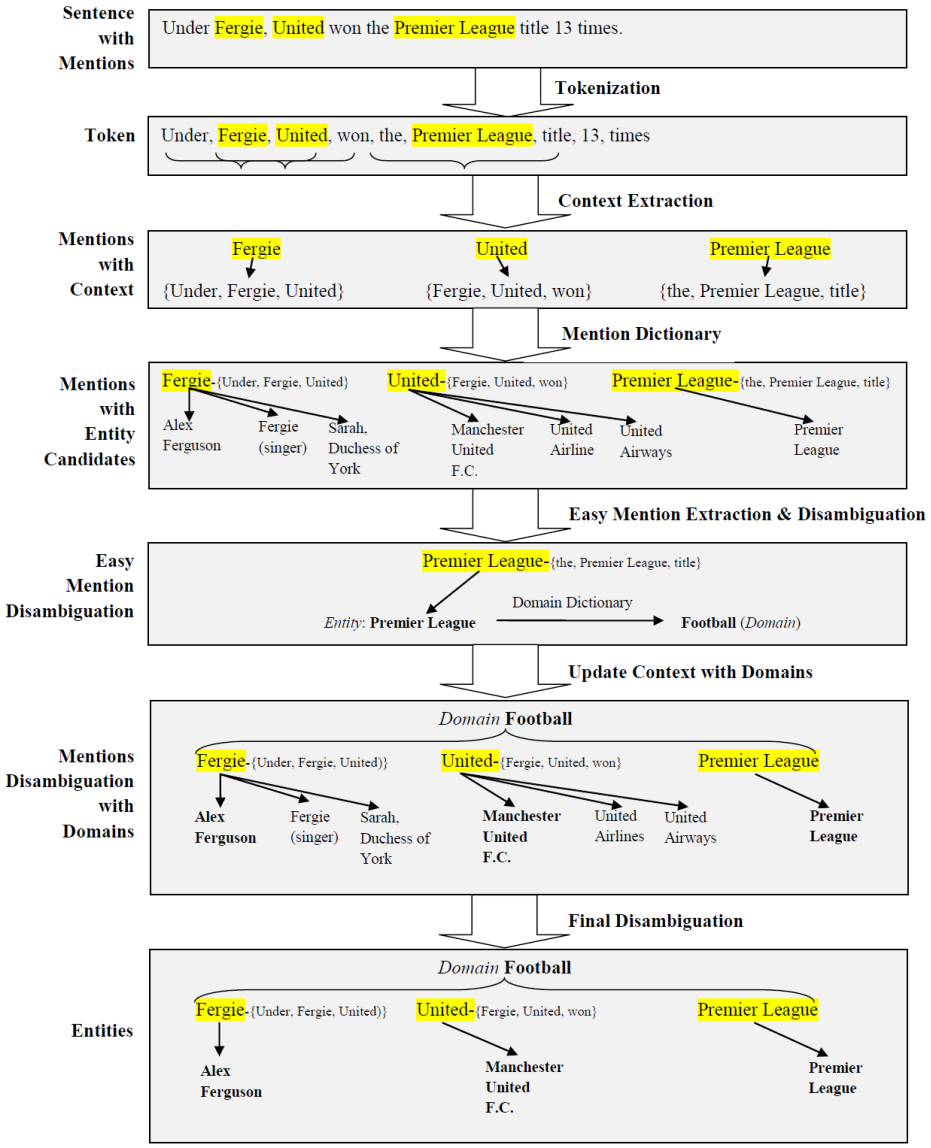


Figure 2: AIDA-*light* processing of sample sentence.

pages where many out-of-dictionary mentions occur, we apply Locality Sensitive Hashing (LSH) in combination with a min-wise permutation scheme [9, 15, 1] to cover more spelling variations among these mentions. That is, once an out-of-dictionary mention occurs, AIDA-*light* first attempts to find similar mentions for it via LSH. Second, all possible entity candidates of these similar mentions are assigned to this mention as candidate entities. Let us take the mention "*Northwest Fur Company*" as an example. This mention does not exist in the *means* relation of YAGO2, and thus there is no entity candidate for it. However, via LSH we are able to detect that "*Northwest Fur Company*" and the dictionary entry "*North West Fur Company*" are highly similar, such that AIDA-*light* is able to identify the entity `North_West_Company` as a candidate entity for this mention. KORE [12] is the first work that integrated LSH into an NED system to cluster key-phrases for an efficient form of similarity computation. Our work touches upon another important issue, which is to deal with the lack of dictionary resources in an NED system.

We remark that, according to the above definitions, two or more mentions, which are represented by identical tokens occurring at different locations in the input sequence may be mapped to different entities. That is, the same annotated name may very well be disambiguated to different meanings if this name occurs at different positions of the input text.

We next explain how we maintain the contexts of both mentions and candidate entities which form the basis for the actual disambiguation step.

3.1.1 Mention Context

The *mention context* is extracted from the annotated text snippet $\mathbb{T}$ that contains a mention M_j which we aim to disambiguate. For both efficiency and robustness reasons,

this mention context is limited to a sliding window $\mathbb{M}' = \langle M_{j-k}, \ldots, M_j, \ldots, M_{j+k} \rangle$ with $2\,k + 1$ mentions that surround M_j. Similarly, we consider also a sliding window $\mathbb{T}' = \langle T_{i-l}, \ldots, T_i, \ldots, T_{i+l} \rangle$ of $2\,l + 1$ tokens surrounding M_j, such that $M_j = NER(T_i)$.

Domains. Within such a mention context, each candidate entity $E \in \mathbb{E}$ may be assigned to (i.e., be mapped to) zero or more domains via a *domain function* $Dom : \mathbb{E} \rightarrow 2^{\mathbb{D}}$. This set of domains $\mathbb{D}$ consists of the collection of 46 manually selected classes [2] (such as "**Football**") of the *subclassOf* hierarchy of YAGO2 shown in Figure 3. As inherited from the original *subclassOf* relation, these 46 domains form a DAG structure.

In the first stage of the NED algorithm, both neighboring tokens and mentions that fall within their respective sliding windows are taken into account by our disambiguation algorithm. Later, as the algorithm proceeds from its first to its second stage, the context of a mention might hold extra information such as the domains that the neighboring mentions under this context are associated with. At the first stage, however, these domains are unknown, and the iterative mapping described in Section 4.2, updates the mention contexts after the first round by the set of chosen entities and their respective domains.

3.1.2 Entity Context

In analogy to the mention context, the *entity context* is used to describe each entity E out of the set of known entities $\mathbb{E}$ with respect to a given background KB such as YAGO2. We focus on two main components for this purpose. The first is a set of key tokens extracted from all the key phrases provided by AIDA for the respective entity. The second component consists of a set of Wikipedia categories which is again obtained from YAGO2 as the background KB.

Key Tokens. Each entity $E \in \mathbb{E}$ is assigned to zero or more key tokens of type *string* via a *token function* $Tok : \mathbb{E} \rightarrow 2^{string}$. These tokens are obtained by simplifying the key phrases provided by AIDA [14] for each entity in YAGO2. For example, the set consisting of the two key phrases { "*U.S. President*", "*President of the U.S.*" } in AIDA is reduced to the set of key tokens { "*president*", "*U.S.*" } in AIDA-*light*.

Wikipedia Categories. Additionally, each entity $E \in \mathbb{E}$ is assigned to zero or more Wikipedia categories via a *category function* $Cat : \mathbb{E} \rightarrow 2^{\mathbb{C}}$. These categories $\mathbb{C}$ are obtained from the *type* relation of YAGO2 (such as `wikicategory:English_footballers`) which thus form the leafs in the *type* system in the YAGO2 knowledge base.

Both of these components together form the basis for defining the pairwise coherence of two entities, or the pairwise similarity between a mention and an entity if the domains are known, respectively.

3.2 Basic Feature Functions

For the following steps, we consider only sets of candidate entities $\mathbb{E}_i = Dict(M_i)$. That is, we consider only for those mentions M_i that occur in the mention context $\mathbb{M}' = \langle M_{j-k}, \ldots, M_j, \ldots, M_{j+k} \rangle$ of a mention M_j that is about to be disambiguated. Moreover, these mentions need to match an entry in our mention dictionary as described earlier. Both these points help us to significantly reduce the amount of candidate entities for a given mention.

3.2.1 Prior

The function *prior* reflects the likelihood that a mention M_i refers to an entity $E_{i,j}$ with respect to the link structure of Wikipedia. It is thus defined as the relative frequency with which a link with anchor M_i points to a Wikipedia article representing entity $E_{i,j}$.

$$\begin{aligned} \forall M_i \in \mathbb{M}', \forall E_{i,j} \in \mathbb{E}_i := Dict(M_i) : \\ f_1(M_i, E_{i,j}) &:= prior(M_i, E_{i,j}) \\ &:= \frac{\text{count}(M_i \rightarrow E_{i,j})}{\text{count}(M_i)} \end{aligned} \quad (1)$$

The respective counts are obtained from a recent WikiMedia dump of English Wikipedia articles[3]. Each such count is normalized by the number of times M_i occurs as an anchor among all links in Wikipedia. It thus produces a real value in the range $[0, 1]$.

3.2.2 Mention-Name Similarity

The similarity between a mention (e.g., "*Obama*") and an entity name (e.g., `Barack_Obama`) is measured by the *Jaccard* similarity over 3-gram character sequences extracted from the respective mention and entity strings.

$$\begin{aligned} \forall M_i \in \mathbb{M}', \forall E_{i,j} \in \mathbb{E}_i := Dict(M_i) : \\ f_2(M_i, E_{i,j}) &:= Jaccard(M_i, E_{i,j}) \\ &:= \frac{|3\text{-}grams(M_i) \cap 3\text{-}grams(E_{i,j})|}{|3\text{-}grams(M_i) \cup 3\text{-}grams(E_{i,j})|} \end{aligned} \quad (2)$$

White-spaces and truncators (such as "_") are removed from the input strings before the 3-grams are generated. Hence, *Jaccard* also produces a real value between 0 and 1.

3.2.3 Context Similarity

The function *context similarity* defines how similar the token context $\mathbb{T}' = \langle T_{i-l}, \ldots, T_i, \ldots, T_{i+l} \rangle$ surrounding a mention M_i and the *key tokens* $Tok(E_{i,j})$ of an entity candidate $E_{i,j}$ are.

$$\begin{aligned} \forall M_i \in \mathbb{M}', \forall E_{i,j} \in \mathbb{E}_i := Dict(M_i) : \\ f_3(M_i, E_{i,j}) &:= Overlap(\mathbb{T}', Tok(E_{i,j})) \\ &:= \frac{|\mathbb{T}' \cap Tok(E_{i,j})|}{min(|\mathbb{T}'|, |Tok(E_{i,j})|)} \end{aligned} \quad (3)$$

It is thus estimated as the overlap coefficient of tokens extracted from the mention context $\mathbb{T}'$ and the key tokens $Tok(E_{i,j})$ (thus using word stems as tokens and excluding stopwords). It produces a real value between 0 and 1.

3.2.4 Surrounding Mention Coherence

This feature function measures the semantic coherence among mentions M_t within the context of a mention M_i which we aim to disambiguate. Thus, given the mention context $\mathbb{M}' = \langle M_{i-k}, \ldots, M_i, \ldots, M_{i+k} \rangle$ that is surrounding a mention M_i, this feature function computes the pairwise coherence between M_i and the remaining mentions $M_t \in \mathbb{M}$, $t \neq i$, based on their co-occurrence counts in Wikipedia.

[2] The 46 domains: *badminton, baseball, basketball, cricket, football, golf, table tennis, rugby, soccer, tennis, volleyball, cycling, skating, skiing, hockey, mountaineering, rowing, swimming, sub, diving, racing, athletics, wrestling, boxing, fencing, archery, fishing, hunting, bowling, agriculture, alimentation, architecture, computer science, engineering, medicine, veterinary, astronomy, biology, chemistry, earth, mathematics, physics, economy, fashion, industry, politics.*

[3] `http://dumps.wikimedia.org/enwiki/20130103/`

For example, if there is a pair of mentions "*Victoria*" and "*David Beckham*", then "*Victoria*" tends to be mapped to `Victoria_Beckham` and "*David Beckham*" to `David_Beckham`, respectively.

$$\forall M_i \in \mathbb{M}', \forall E_{i,j} \in \mathbb{E}_i := Dict(M_i) :$$
$$f_4(M_i, E_{i,j}) := Coherence(M_i, E_{i,j})$$
$$:= \frac{\prod_{M_t \in \mathbb{M}', t \neq i} \frac{co\text{-}occurrence\_count(M_t, E_{i,j})}{count(M_t)}}{\sum_{E_{i,u} \in \mathbb{E}_i} \prod_{M_t \in \mathbb{M}', t \neq i} \frac{co\text{-}occurrence\_count(M_t, E_{i,u})}{count(M_t)}} \quad (4)$$

This features thus resembles the conditional probability $P(E_{i,j}|M_t)$ that an entity $E_{i,j}$ occurs when we are given a surrounding mention M_t. In YAGO2, this probability is again estimated over an English Wikipedia dump by counting how many times entity $E_{i,j}$ and mention M_t occur together in an article. This count is normalized by the absolute number of times mention M_t occurs (i.e., using $count(M_t)$) in all articles. In order to avoid counts of zero, we apply a simple smoothing technique for our implementation by assigning 1 to unseen mention-entity pairs. This already produces a real value from 0 to 1. However, it is quite small in most cases. And thus, we apply a linear normalization to all values among the candidate set $\mathbb{E}_i$ of a mention.

3.2.5 Entity-Entity Context Coherence

The *entity-entity context coherence* function finally reflects the pairwise relatedness between the two entities via their key tokens sets $Tok(E_{i,j})$ and $Tok(E_{t,v})$.

$$\forall M_i, M_t \in \mathbb{M}', \forall E_{i,j} \in \mathbb{E}_i := Dict(M_i),$$
$$\forall E_{t,v} \in \mathbb{E}_t := Dict(M_t) :$$
$$f_6(E_{i,j}, E_{t,v}) := Overlap(Tok(E_{i,j}), Tok(E_{t,v}))$$
$$:= \frac{|Tok(E_{i,j}) \cap Tok(E_{t,v})|}{min(|Tok(E_{i,j})|, |Tok(E_{t,v})|)} \quad (5)$$

It is once more computed as the overlap coefficient of two *key tokens* sets. It produces a real value from 0 to 1.

3.3 Domain-Oriented Feature Functions

3.3.1 Domain Hierarchy

Despite the obvious usefulness of Wikipedia categories for estimating the coherence of two entities or the coherence of an entity and a domain, there—so far—has not been much work investing this particular issue. We thus propose to build hierarchies between Wikipedia categories (instances of the *type* relation) and domains (leafs in the *subclassOf* relation system). Both kinds of semantic relations are available in YAGO2 [13] via the *yagoWordnetDomains* and *wordnetDomainHierarchy* relations[4]. That is, a Wikipedia category is first mapped to its WordNet [6] type, and it is then traced back to a more generic domain in one of the 46 domains, such as "**Sports**" (including football, basketball, etc.), "**Science**" (including computer science, mathematics, etc.), and so on, which are second-level nodes in the *wordnetDomainHierarchy* relation in YAGO2. Note that, we manually removed overly generic (and thus noisy) domains, such as the "**Person**" domain, which is associated with the `wikicategory:Living_people` category in Wikipedia and thus spans multiple of the aforementioned domains.

[4] `http://www.mpi-inf.mpg.de/yago-naga/yago/downloads.html`

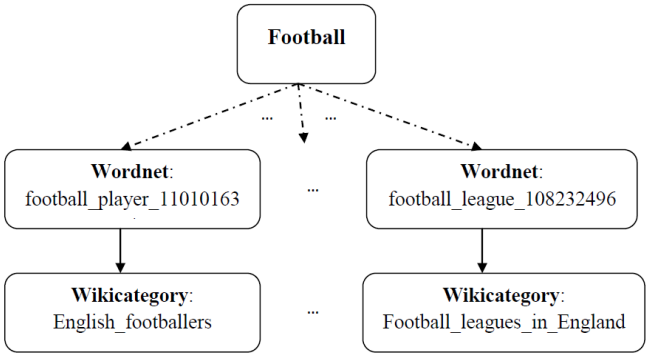


Figure 3: A Wikipedia category hierarchy for the "Football" domain.

Once these hierarchies (as shown in Figure 3) are built, the coherence between two such domains, or between a category and a domain, can be computed as the inverse of the length of the shortest path between them in this hierarchy. In addition, this feature may help to predict the domain of the input text when some of the mentions with low ambiguity are already fixed to entities, which results in a better local similarity estimation for the remaining, more ambiguous mentions. For example, if a number of "easy mentions" have been mapped to soccer players, clubs or leagues, we can predict that the text likely relates to the domain "**Football**".

3.3.2 Domain-Entity Coherence

The *domain-entity coherence* function reflects the (binary) relevance of assigning an entity to one of the predefined domains (e.g., "**Football**").

$$\forall M_i \in \mathbb{M}', \forall E_{i,j} \in \mathbb{E}_i := Dict(M_i), Domains\ \mathbb{D} :$$
$$f_5(M_i, E_{i,j}) := \begin{cases} 1 & \exists D \in \mathbb{D} \wedge E_{i,j} \in D \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

In case a chosen entity candidate $E_{i,j}$ belongs to one of our predefined domains $\mathbb{D}$, we are thus able to take the coherence between this entity candidate and the domain hierarchy into account. This feature function only produces a binary value of 0 or 1, which reflects whether there is a domain $D \in \mathbb{D}$ holding $E_{i,j}$, or not.

3.3.3 Entity-Entity Category Coherence

When using both YAGO2 or DBpedia 3.8 as background KB, Wikipedia categories (such as `wiki-category:English_footballers` or `wiki-category:FIFA_World_Cup`) may provide good hints on the coherence among pairs of entities, especially for long-tail entities with otherwise poor context information. The *entity-entity category coherence* function thus estimates the relatedness between the two entities via their assigned Wikipedia categories $Cat(E_{i,j})$ and $Cat(E_{t,v})$.

$$\forall M_i, M_t \in \mathbb{M}', \forall E_{i,j} \in \mathbb{E}_i := Dict(M_i),$$
$$\forall E_{t,v} \in \mathbb{E}_t := Dict(M_t) :$$
$$f_7(E_{i,j}, E_{t,v}) := Coherence(Cat(E_{i,j}), Cat(E_{t,v})) \quad (7)$$
$$:= \max_{\substack{C_{i,j,x} \in Cat(E_{i,j}) \\ C_{t,v,y} \in Cat(E_{t,v})}} \frac{1}{Distance(C_{i,j,x}, C_{t,v,y})}$$

Here, the function *Distance* of two categories is computed as the shortest path between them in a domain hierarchy $\mathbb{D}$.

It is computed as the maximum coherence over all pairs of Wikipedia categories the two candidate entities belong to. The coherence of a category pair can be computed by using the domain-hierarchy described above. It produces a real value from 0 to 1.

3.4 Objective Function

To aim for a robust disambiguation algorithm, we combine the seven individual feature functions described above into a single objective function (all functions except *prior* are computed at runtime). In doing so, we disambiguate the entire sequence of mentions occurring in an input document *jointly*. Thus let $\mathbb{M}_D = \langle M_1, \ldots, M_k \rangle$ denote this sequence of mentions occurring in the input document, and let $\mathbb{E}_D = \langle E_1, \ldots, E_k \rangle$, $E_i \in Dict(M_i)$, denote its corresponding set of disambiguated entities. We select exactly one entity candidate E_i for each mention $M_i \in \mathbb{M}_D$, $i = 1..k$ subject to the following optimization objective $NED : 2^{\mathbb{M}} \rightarrow 2^{\mathbb{E}}$.

$$NED(\langle M_1, \ldots, M_k \rangle) := \langle E_1, \ldots, E_k \rangle$$
$$:= \text{argmax}_{E_1 \in Dict(M_1) \times \cdots \times E_d \in Dict(M_k)} \sum_{\substack{j=1..5 \\ i=1..k}} p_j \, f_j(M_i, E_i) + \sum_{\substack{j=6..7 \\ t=1..k \\ v=1..k \\ t \neq v}} p_j \, f_j(E_t, E_v) \quad (8)$$

such that $\sum_{j=1..5} p_j = 1$ and $p_6 + p_7 = 1$.

The first five feature functions $f_1, ..., f_5$ thus reflect the likelihood of choosing an entity based on the respective mention and entity contexts, while functions f_6 and f_7 compute the pairwise coherences among the chosen entities.

We remark that, as described before for the feature functions, the mention context for each mention $M_i \in \mathbb{M}_D$ still is based on a sliding window of mentions $\mathbb{M}'$ and tokens $\mathbb{T}'$ surrounding that mention (see Section 3.1.1).

4. DISAMBIGUATION ALGORITHM

We construct a weighted, undirected *dependency graph* whose nodes are formed by all mentions and their candidate entities, and whose weighted edges are created by the feature functions described in the previous section [14]. Consequently, the objective function described in Section 3.4 is solved by finding the best subgraph where each mention is mapped onto only one entity candidate. This dense-subgraph problem is known to be NP-hard [21] as it generalizes the well-studied Steiner-tree problem. We thus adopt the greedy approximation algorithm proposed in [14] and extend it into a new, two-stage disambiguation algorithm.

4.1 Entity-Mention Dependency Graph

The mention-entity dependency graph (see Figure 1 for an example) is typically dense on the entity side, often having hundreds or thousands of nodes because there might be many candidate entities for common mentions (e.g., common first names, last names, etc.). Therefore, in order to stay efficient, we first propose to keep at most k of the best entity candidates for each mention in the graph. Thanks to the multi-phase computation (see the next subsection), which iteratively updates the context and thus gives a better initial mention-entity similarity estimation, this cut-off at the top k (even with fairly small values of k, e.g., $k = 5$) does not affect accuracy too much. Second, an entity node which is not the last remaining candidate entity of a mention (and hence a "*non-taboo*" node), and which holds the currently smallest weighted degree, is iteratively removed from the mention-entity graph as shown in Algorithm 1.

Algorithm 1 Collective Mapping Algorithm.

Require: Weighted dependency graph of mentions and entities.
Ensure: Best sub-graph with one edge per mention.

```
//build the graph
for each mention M do
    for each candidate entity E_i of mention M do
        sim(M, E_i) := Σ_{t=1..5} p_t f_t(M, E_i); //local similarity
    end for
    Keep the k best candidates, drop the others;
end for
for each candidate entity E_i of mention M do
    W_i := sim(M, E_i); //init weight
    for each entity E_j not generated from mention M do
        //compute coherence
        coh(E_i, E_j) := p_6 f_6(E_i, E_j) + p_7 f_7(E_i, E_j);
        W_i += coh(E_i, E_j); //update weight
    end for
end for
/*an entity is non-taboo if it is not the last candidate of
 a mention*/
while graph has non-taboo entity do
    Determine non-taboo entity node with lowest weight,
    remove it and all its incident edges;
    Recompute the weights;
end while
```

4.2 Two-Stage Computation

Current NED approaches either map each mention separately in a "*context-free*" manner, which cannot capture the intricate coherences among entities, or they map all mentions together in an expensive joint-mapping step. However, in practice the level of ambiguity of various mentions occurring in a text snippet often varies substantially. Often, there are mentions which are not very difficult to disambiguate because of the very few candidates (e.g., less than 4 candidates based on our experiments) that come into question for them. We thus propose to organize the mapping algorithm into 2-stage computation as shown in Algorithm 2.

For each stage of the disambiguation algorithm, we apply the collective mapping algorithm described in the previous subsection by using the 7 features functions from Section 4.1. The unknown context-related domains $\mathbb{D}'$ in the first stage does not affect the algorithm because the corresponding feature function f_6 only returns 0. The multi-phase computation not only helps to reduce the complexity of collective mapping algorithm but it also contributes to a better local similarity estimation for "highly ambiguous" mentions (in the second stage) by updating the mention contexts such as context-related domains or already chosen entities (from the first stage).

5. EXPERIMENTS

5.1 Setup

Test corpora. We conducted experiments for four different test corpora, with different characteristics:

Algorithm 2 Multi-Phase Computation.

Require: Sequence of mentions $\mathbb{M}'$;
1: Initialize context-related domains $\mathbb{D}' := \emptyset$;
2: Find sets $\mathbb{E}_i = Dict(M_i)$, $M_i \in \mathbb{M}'$, of "easy mentions" such that $|\mathbb{E}_i| \leq 4$;
3: Jointly disambiguate sets of candidate entities for these easy mentions using Algorithm 1 and let $\mathbb{M}'' \subseteq \mathbb{M}'$ denote the set of mentions disambiguated after the first stage;
4: **for** each domain $D \in \mathbb{D}$ **do**
5: **if** at least half of the entities chosen in the first stage relate to D **then**
6: add D to $\mathbb{D}'$;
7: **end if**
8: **end for**
9: Recompute the mention-entity similarities;
10: Disambiguate $\mathbb{M}' - \mathbb{M}''$ using Algorithm 1;

- **CoNLL-YAGO**: The CoNLL-YAGO testb corpus was originally used in [14]. It is based on the CoNLL 2003 dataset, which consists of 231 news articles (4485 mentions) with an average article length of 216 words. There are long-tail entities in form of short mentions, which challenges NED systems to get high precision on it. For example, it is challenging to map the mention *"Japan"* in the sentence *"SOCCER- Japan get lucky win."* to the right entity `Japan_national_football_team`, not to the prominent entity `Japan`.
- **WP**: The WP corpus [12] consisting of 2019 difficult sentences (10318 mentions) is a specific sample from Wikipedia articles, with emphasis on short context (52 words on average) and highly ambiguous mentions and long-tail entities. Let's take the text mentioning `AC/DC` band as an example:
"Johnson, Young, Rudd, Young and Williams, live in Tacoma, Washington, 31 August 2009."
These short person names with hundreds of entity candidates are hard for any current NED systems to deal with.
- **Wiki-links**: We randomly extracted a subset of the Wikilinks dataset [5], consisting of 10665 annotated articles (21643 mentions) with an average article length of 5344 words. They are web-pages in a number of websites including blogspot [6], livejournal [7], etc. Note that in this dataset, the annotated mentions are limited to href anchor texts that point to a Wikipedia page. This way, each article has only a marked-up mentions (often only one or two), although many additional entities are mentioned in the text. Moreover, most annotated mentions refer to prominent entities.
- **Wikipedia articles**: We randomly extract 153 Wikipedia featured articles [8], with a total of 19264 mentions and an average article length of 9487 words. This corpus covers a variety of topics such as football (`Arsenal_F.C` page), entertainment (`Batman` page), people (`John_Calvin` page), etc.

[5] `http://code.google.com/p/wiki-links/`
[6] `http://abdallahhouse.blogspot.de`
[7] `http://www.livejournal.com/`
[8] `http://en.wikipedia.org/wiki/Wikipedia:Featured_articles`

For each of these four NED tasks, ground truth is given, either by manual annotation or by the href links in Wikipedia (which, of course, are not given to the NED methods). In all cases, we consider only mentions that refer to individual entities like people, organizations, places, events, etc. We disregard text phrases that refer to general concepts such as "peace", "harmony", "logic", "statistics", etc. In so-called Wikification tasks, such as TAC KBP Entity Linking, all Wikipedia articles are considered as targets, including general concepts. For the Web of Linked Data and this paper's emphasis on NED, this does not make sense; hence our focus on individual entities. We implement this restriction by considering only mentions with ground-truth entities that are in both Wikipedia and YAGO (as YAGO does not include general concepts of the above kind).

Parameters. The parameters $p_1 \ldots p_7$ to combine AIDA-*light* 's feature functions were manually tuned with the goal of maximizing the precision of AIDA-*light* on CoNLL-YAGO testa training corpus. Plus, based on our experiments, we chose a sliding window of 11 sentences (5 sentences before, 5 sentences after and the sentence holding the mention) to extract the context of a mention.

Performance measures. Our measure for output quality is the per-mention precision: the fraction of mentions that are correctly mapped to the proper entities. When a system has a choice of abstaining on a difficult mention by reporting "null" although the proper entity is in the knowledge base, we count this as a false mapping. Alternatively, we could count precision only on the non-null entities, and additionally consider the correctly mapped fraction of all entities as recall. However, our notion of precision captures both of these aspects in a single metric.

Our measure for efficiency is the run-time, measured in single-threaded mode with cold caches on a server with 8 (+8 in hyper-threading mode) Intel Xeon CPUs at 2.4GHz and 48GB of memory.

Systems under comparison. We compared AIDA-*light* against the following state-of-the-art methods:
- **AIDA** [14, 12]: using rich features stored in a SQL database;
- **Spotlight** [17, 5]: statistical version without using any thresholds to get as many entities as possible.

We did not include TagMe [7] in our comparison, as it uses its own special method for mention recognition; so we could not control that it would process exactly the same mentions as the other methods. Besides, TagMe has been designed for very short texts like tweets—a use-case not considered here.

In the following, we first report on experiments with different AIDA-*light* configurations, then on our comparison against state-of-the-art baselines regarding output quality, and finally on run-time measurements.

5.2 AIDA-light in Different Configurations

In order to study the influence of the different assets of AIDA-*light*, we configured our system in different ways and ran it on the CoNLL data, which presumably is the most difficult one of our four test sets.

One possible concern about our architecture is that the multi-stage algorithm could suffer from early mistakes that propagate in the processing pipeline. Our experiments showed a precision of 96% for the "easy" mentions identified in stage 1. So this potential concern was not an issue.

A second concern was that using only the top K candidate entities for each mention is risky in potentially missing the proper entity. We compared two different settings in this regard:

- **Top 5**: Iteratively updating the context in our multi-phase computation is deactivated. The local similarity is computed with the lack of context-related domains.
- **Top 5 with domain coherence**: The local similarity is updated by the chosen entities for "easy mentions". That is, the entity-domain coherence is taken into account.

Table 1 shows the top-5 precision results for these two settings. When the correct entity of a mention was among the top-5 candidates, we count this stage as being successful, otherwise it is counted as false. The results demonstrate that the context-related domains have substantial benefit: 3% better precision. In general, the precision of 95.2% at this first stage is very high, especially considering that the CoNLL corpus has very difficult texts (with short contexts, long-tail entity names, metonyms, etc.).

Table 1: Top-5 precision of AIDA-*light* for top 5 entity candidates on CoNLL.

Method	Top5	Top5+Domain Coherence
CoNLL-YAGO	92.3%	95.2%

Finally, we report the results of AIDA-*light* with all mentions on CoNLL, in the following three configurations:

- **KW**: AIDA-*light* uses the baseline features including a bag of key tokens and surrounding mentions. The collective algorithm is applied in a single round to the entire mention set (deactivating the two-stage approach).
- **KW and easy mentions**: the multi-phase computation is activated. However, AIDA-*light* does not update the context (e.g. domain feature) after the first round.
- **KW, easy mentions, and domains**: All components in AIDA-*light* including multi-phase computation and domain features are turned on.

The results are shown in Table 2. Obviously, each additional asset helps AIDA-*light* in gaining output quality. Most notably, enabling "*easy mention*" disambiguation stage improves precision by almost 4%. Note that this is not at the expense of increased run-time.

5.3 Comparisons on NED Quality

In our experiments, Illinois Wikifier performed poorly with precision results around 70% or worse, probably because of code changes (and perhaps resulting bugs) compared to the version used in [20]. To avoid improper comparisons, we therefore discuss only the results of the remaining three competitors: AIDA-*light* , AIDA, and Spotlight.

Table 3: Precision over corpora (best method per row is in boldface; statistically significant improvements over Spotlight are marked with an asterisk).

Dataset	AIDA	AIDA-*light*	Spotlight
CoNLL-YAGO	82.5%*	**84.8%***	75.0%
WP	**84.7%***	84.4%*	63.8%
Wikipedia articles	**90.0%**	88.3%	89.6%
Wiki-links	80.3%	**85.1%***	80.7%

Table 3 shows the precision results for the three systems. AIDA-*light* outperforms the other two systems on CoNLL and Wiki-links. Especially, a paired t-test shows that AIDA-*light* is significantly better than Spotlight on both corpora (p-value < 0.01). Particularly on CoNLL, AIDA-*light* can handle many difficult entities such as football teams mentioned in the form of city names, national sport teams in the form of country names, etc. On Wiki-links, the point that brings 4% higher precision than others is AIDA-*light* 's strong ability to handle mentions that differ from the official names of entities (e.g. "*Northwest Fur Company*" for the entity `North_West_Company`). AIDA-*light* is slightly worse than AIDA on WP, but still much better than Spotlight (20% higher precision). On Wikipedia articles, the best system is AIDA (90.0%), followed by Spotlight (89.6%) and AIDA-*light* (88.3%). However, there is no significant difference among the three systems.

Overall, AIDA-*light* consistently exhibits high output quality, whereas Spotlight is inferior on very difficult input texts. AIDA is usually close in performance to AIDA-*light*, but clearly loses on Wiki-links because of the small number of mentions per document for this test case.

5.4 Comparisons on Run-Time Efficiency

The original AIDA system uses a SQL database as a backend. Therefore, it is an order of magnitude slower that the systems that use customized in-memory data structures. To avoid reporting apples vs. oranges, we exclude AIDA from the following run-time comparisons, leaving us with two competitors: AIDA-*light* and Spotlight.

Table 4: Average per-document run-time results.

Dataset	AIDA-*light*	Spotlight
CoNLL-YAGO	0.47s	0.51s
WP	0.05s	0.14s
Wikipedia articles	5.47s	4.22s
Wiki-links	0.18s	0.32s

Table 4 shows the average run-times per document for the four test corpora. Both systems are very fast. AIDA-*light* is faster on CoNLL, WP, and Wiki-links, and slightly slower on Wikipedia articles. The reason is that AIDA-*light* computes coherence including context coherence and type coherence over all entity pairs, and thus, it becomes slower to process large documents with hundreds of mentions.

Table 5 shows the run-times on the Wiki-links corpus in more detail. The reason for choosing Wiki-links is that it is the largest one among our corpora with 10665 documents. The table gives 90% confidence intervals for the per-document average run-time, and also the standard deviation. AIDA-*light* is much faster than Spotlight on average. However, they are almost the same on the 90% quantile, that is, the 10% longest or most difficult documents. The reason is the increasing complexity of AIDA-*light* when the number of mentions increases. Overall, however, AIDA-*light* clearly outperformed its competitor on this dataset.

6. CONCLUSION

AIDA-*light* is a new NED system that reconciles high output quality with fast run-time. Our experiments have shown that AIDA-*light* consistently performs as well as or better

Table 2: Precision results of AIDA-*light* in different configurations on CoNLL.

Settings	KW		KW+Easy Mentions		KW+Easy Mentions+Domain	
	Accuracy	Avg Running Time	Accuracy	Avg Running Time	Accuracy	Avg Running Time
CoNLL-YAGO	78.1%	0.42s	81.9%	0.42s	84.8%	0.47s

Table 5: Average run-times on Wiki-links (90% confidence intervals)

Method	Avg. Run-Time (ms)		90% Quantile (ms)	
	Mean	Standard Deviation	Mean	Standard Deviation
AIDA-*light*	182 ± 35	2250	1225 ± 352	7032
Spotlight	323 ± 44	2810	1234 ± 443	8831

than state-of-the-art competitors, over a variety of corpora including news (CoNLL), difficult and short contexts (WP), Web pages (Wiki-links), and Wikipedia articles.

By its two-stage algorithm, its simple but expressive features including thematic domains, and its low footprint, AIDA-*light* is geared for high-throughput usage at Web scale. Our measurements showed that AIDA-*light* can process a typical Web page (from the Wiki-links corpus) within 0.2 seconds on mid-range hardware. Its architecture easily allows it to scale out the processing of a large corpus across the nodes of a cluster or distributed platform. By its small memory consumption, AIDA-*light* runs well even on low-end nodes of such platforms. Thus, with a single-thread throughput of 5 pages/second, we can process, for example, the complete ClueWeb'12 collection (`lemurproject.org/clueweb12/`) with ca. 700 Million pages, in about one week (including NER) on a mid-range 32-node, 16 cores-per-node cluster.

Our future work includes further improvements of the multi-stage algorithm, and especially a deeper integration of the mention recognition (NER, currently implemented by the CRF-based Stanford Tagger) with the NED method.

7. REFERENCES

[1] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise Independent Permutations. *Journal of Computer and System Sciences 1998.*

[2] R. Bunescu and M. Pasca. Using Encyclopedic Knowledge for Named Entity Disambiguation. *EACL 2006.*

[3] M. Cornolti, P. Ferragina, and M. Ciaramita. A Framework for Benchmarking Entity-annotation Systems. *WWW 2013.*

[4] S. Cucerzan. Large-scale Named Entity Disambiguation Based on Wikipedia Data. *EMNLP-CoNLL 2007.*

[5] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes. Improving Efficiency and Accuracy in Multilingual Entity Extraction. *I-SEMANTICS 2013.*

[6] C. Fellbaum. WordNet: An Electronic Lexical Database. *MIT Press 1998.*

[7] P. Ferragina and U. Scaiella. Tagme: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). *CIKM 2010.*

[8] J. R. Finkel, T. Grenager, and C. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *ACL 2005.*

[9] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. *VLDB 1999.*

[10] X. Han and J. Zhao. Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge. *CIKM 2009.*

[11] T. Heath and C. Bizer. Linked Data: Evolving the Web into a Global Data Space. *Morgan Claypool 2011.*

[12] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum. Kore: Keyphrase Overlap Relatedness for Entity Disambiguation. *CIKM 2012.*

[13] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. *WWW 2011.*

[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. *EMNLP 2011.*

[15] P. Indyk and R. Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. *STOC 1998.*

[16] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective Annotation of Wikipedia Entities in Web Text. *KDD 2009.*

[17] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia Spotlight: Shedding Light on the Web of Documents. *I-SEMANTICS 2011.*

[18] D. Milne and I. H. Witten. Learning to Link with Wikipedia. *CIKM 2008.*

[19] L. Ratinov and D. Roth. Design Challenges and Misconceptions in Named Entity Recognition. *CoNLL 2009.*

[20] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and Global Algorithms for Disambiguation to Wikipedia. *HLT 2011.*

[21] M. Sozio and A. Gionis. The Community-search Problem and How to Plan a Successful Cocktail Party. *KDD 2010.*