J-NERD: Joint Named Entity Recognition and Disambiguation with Rich Linguistic Features

Abstract

Methods for Named Entity Recognition and Disambiguation (NERD) perform NER and NED in two separate stages. Therefore, NED may get penalized by NER false positives, and suffers in recall by false negatives. Conversely, NED does not fully exploit information computed by NER such as types of mentions. This paper presents J-NERD, a new approach to perform NER and NED *jointly*, by means of a probabilistic graphical model that captures mention spans, mention types, and the mapping of mentions to entities in a knowledge base. We present experiments with different kinds of texts from the CoNLL'03, ACE'05, and ClueWeb'09-FACC1 corpora. J-NERD consistently outperforms state-of-the-art competitors in end-to-end NERD precision, recall, and F1.

1 Introduction

Motivation: Methods for Named Entity Recognition and Disambiguation, NERD for short, typically proceed in two stages:

- At the NER stage, text spans of entity mentions are detected and tagged with coarse-grained types like person, organization, location, etc. This is typically performed by a trained Conditional Random Field (CRF) over word sequences (e.g., (Finkel et al., 2005)).
- At the NED stage, mentions are mapped to entities in a knowledge base (KB) based on contextual similarity measures and the semantic coherence of the selected entities (e.g., (Cucerzan, 2014; Hoffart et al., 2011; Ratinov et al., 2011)).

This two-stage approach has limitations. First, NER may produce false positives that can misguide NED. Second, NER may miss out on some entity mentions, and NED has no chance to compensate for these false negatives. Third, NED is not able to help NER, for example, by disambiguation "easy" mentions (e.g., of prominent entities with more or less unique names) and then using the entities and knowledge about them as enriched features for NER.

Example: Consider the following sentences:

David played for manu, real, and la galaxy.

His wife posh performed with the spice girls.

This is difficult for NER because of the absence of upper-case spelling, which is not untypical in social media, for example. Most NER methods will miss out on multi-word mentions or words that are also common nouns ("spice") or adjectives ("posh", "real"). Typically, NER would pass only the mentions "David", "manu", and "la" to the NED stage, which then is prone to many errors like mapping the first two mentions to any prominent people with first names David and Manu, and mapping the third one to the city of Los Angeles. With NER and NED performed jointly, the possible disambiguation of "la galaxy" to the soccer club can guide NER to tag the right mentions with the right types (e.g., recognizing that "manu" could be a short name for a soccer team), which in turn helps NED to map "David" to the right entity `David_Beckham`.

Contribution: This paper presents a novel kind of probabilistic graphical model for the joint recognition and disambiguation of named-entity mentions in natural-language texts. With this integrated approach to NERD, we aim to overcome the limitations of the two-stage NER/NED methods discussed above.

Our method, coined *J*-NERD , is based on a supervised, non-linear CRF that combines multiple per-sentence CRF's into an entity-coherence-aware global CRF. The global CRF detects mention spans, tags them with coarse-grained types, and maps them to entities in a single joint-

inference step based on Gibbs sampling. The *J*-NERD method comprises the following novel contributions:

- a tree-shaped CRF for each sentence, whose structure is derived from the dependency parse tree and thus captures linguistic context in a deeper way compared to prior work with CRF's for NER and NED;
- richer linguistic features not considered in prior work, harnessing dependency parse trees and verbal patterns that indicate mention types as part of their ***nsubj*** or ***nobj*** arguments;
- an inference method that maintains the uncertainty of both mention candidates (i.e., token spans) and entity candidates for competing mention candidates and makes joint decisions, as opposed to fixing mentions before reasoning on their disambiguation.

We present experiments with three major datasets: the CoNLL'03 collection of newswire articles, the ACE'05 corpus of news and blogs, and the ClueWeb'09-FACC1 corpus of web pages. Baselines that we compare *J*-NERD with include AIDA-light (Nguyen et al., 2014), Spotlight (Daiber et al., 2013), and TagMe (Ferragina and Scaiella, 2010), and the recent joint NER/NED method of Durrett and Klein (2014). *J*-NERD consistently outperforms these competitors in terms of both precision and recall. The *J*-NERD code and all experimental data are publicly available at the URL `anonymized-for-doubleblind-review`.

2 Related Work

NER: Detecting the boundaries of text spans that denote named entities has been mostly addressed by supervised CRF's over word sequences (McCallum and Li, 2003; Finkel et al., 2005). The work of Ratinov and Roth (2009) improved these techniques by additional features from context aggregation and external lexical sources (gazetteers, etc.). Passos et al. (2014) harnessed skip-gram features and external dictionaries for further improvement. An alternative line of NER techniques is based on dictionaries of name-entity pairs, including nicknames, short-hand names, and paraphrases (e.g., "the first man on the moon"). The work of Ferragina (Ferragina and Scaiella, 2010) and Mendes (Mendes et al., 2011) are examples of dictionary-based NER. The work of Spitkovsky and Chang (2012) is an example for a large-scale dictionary that can be harnessed by such methods.

An additional output of the CRF's are type tags for the recognized word spans, typically limited to coarse-grained types like *Person*, *Organization*, and *Location* (and also *Miscellaneous*). The most widely used tool of this kind is the Stanford NER Tagger (Finkel et al., 2005). Many NED tools use the Stanford NER Tagger for their first stage of detecting mentions.

Mention Typing: The specific NER of inferring semantic types has been further refined and extended by various works on fine-grained typing (e.g., politicians, musicians, singers, guitarists) for entity mentions and general noun phrases (Fleischman and Hovy, 2002; Rahman and Ng, 2010; Ling and Weld, 2012; Yosef et al., 2012; Nakashole et al., 2013). Most of this work is based on supervised classification, using linguistic features from mentions and their surrounding text. One exception is the work of Nakashole et al. (2013) which is based on text patterns that connect entities of specific types, acquired by sequence mining from the Wikipedia full-text corpus. In contrast to our work, these are simple surface patterns, and the task addressed here is limited to typing noun phrases that likely denote emerging entities that are not yet registered in a KB.

NED: Methods and tools for NED go back to the seminal works of (Dill et al., 2003; Bunescu and Pasca, 2006; Cucerzan, 2007; Milne and Witten, 2008). More recent advances led to open-source tools like the Wikipedia Miner Wikifier (Milne and Witten, 2013), the Illinois Wikifier (Ratinov et al., 2011), Spotlight (Mendes et al., 2011), Semanticizer (Meij et al., 2012) TagMe (Ferragina and Scaiella, 2010; Cornolti et al., 2014), and AIDA (Hoffart et al., 2011) with its improved variant AIDA-light (Nguyen et al., 2014). We choose some, namely, Spotlight, TagMe and AIDA-light, as baselines for our experiments. These are the best-performing, publicly available systems for news and web texts. Most of these methods combine contextual similarity measures with some form of considering the coherence among a selected set of candidate entities for the disambiguation. The latter aspect can be cast into a variety of computational models, like graph algorithms (Hoffart et al., 2011), integer linear programming (Ratinov et al., 2011), or probabilistic graphical models (Kulkarni et al., 2009). All these methods use the Stanford NER Tagger or dictionary-based

matching for their NER stages. Kulkarni et al. (2009) uses an ILP or LP solver (with rounding) for the NED inference, which is computationally expensive.

Note that some of the NED tools aim to link not only named entities but also general concepts (e.g. "world peace") for which Wikipedia has articles. In this paper, we solely focus on proper entities.

Joint NERD: There is little prior work on performing NER and NED jointly. (Sil and Yates, 2013; Durrett and Klein, 2014) are the most notable methods. Sil and Yates (2013) first compiles a liberal set of mention and entity candidates, and then performs joint ranking of the candidates. Durrett and Klein (2014) presents a CRF model for coreference resolution, mention typing, and mention disambiguation. Our model is also based on CRF's, but distinguishes itself from prior work in three ways: 1) tree-shaped per-sentence CRF's derived from dependency parse trees, as opposed to merely having connections among mentions and entity candidates; 2) linguistic features about verbal phrases from dependency parse trees; 3) maintaining candidates for both mentions and entities and jointly reasoning on their uncertainty. Our experiments include comparisons with the method of (Durrett and Klein, 2014).

There are also benchmarking efforts on measuring the performance for end-to-end NERD (Cornolti et al., 2013; Carmel et al., 2014; Usbeck et al., 2015), as opposed to assessing NER and NED separately. However, to the best of our knowledge, none of the participants in these competitions considered integrating NER and NED.

3 Overview of the *J*-NERD Method

For labeling the token sequence $\langle tok_1, \ldots, tok_m \rangle$ with mention boundaries, NER types, and NED entities, we have devised different kinds of *graphical models* (Koller et al., 2007). While state-of-the-art methods like the Stanford NER tagger employ a linear-chain CRF (Sutton and McCallum, 2012) for this task, we use more sophisticated *tree-shaped models* whose structure is obtained from dependency parse trees. The per-sentence CRF's are then combined into a global model by adding cross-sentence dependencies whenever the same token sequence appears as mention candidates in several sentences. Figure 3 gives an example of a global CRF for two sentences.

These CRF models use a variety of features. Some of these are fairly standard for NER/NED, whereas others are novel contributions of this paper:

- Standard features include lexico-syntactic properties of tokens like POS tags, matches in dictionaries/gazetteers, and similarity measures between token strings and entity names. Also, entity-entity coherence is an important feature for NED – not exactly a standard feature, but used in some prior works.
- Features about the topical domain of an input text (e.g., politics, sports, football, etc.) are obtained by a classifier based on "easy mentions": those mentions for which the NED decision can be made with very high confidence without advanced features. The use of domains for NED was introduced by (Nguyen et al., 2014). Here we further extend this technique by harnessing domain features for joint inference on NER and NED.
- The most advanced feature group captures typed dependencies from the sentence parsing. Such features have not been used in prior work.

The feature space of *J*-NERD is presented in detail in Section 4; the graphical models and their learning and inference methods are further discussed in Section 5.

In the rest of this section we introduce various building blocks that *J*-NERD uses for preprocessing inputs and for computing features.

3.1 Language Processing

We employ the Stanford CoreNLP tool suite (`nlp.stanford.edu/software/corenlp.shtml`) for processing input documents. This includes tokenization, sentence detection, POS tagging, lemmatization, and dependency parsing. All of these provide features for our CRF model. In particular, we harness *dependency types* between noun phrases (de Marneffe et al., 2006), like ***nsubj***, ***dobj***, ***prep_in***, ***prep_for***, etc.

3.2 Entity Repository and Name-Entity Dictionary

We utilize a large knowledge base, namely, YAGO2 (Hoffart et al., 2013), as an entity repositiory and as a major source of a dictionary of name-entity pairs (aliases incl. paraphrases). For the latter, we import the YAGO2 $means$ and $hasName$ relations, a total of more than 6 Mil-

lion name-entity pairs (for ca. 3 Million distinct entities). We also derive NER-type-specific phrase dictionaries. To this end, we also import supporting phrases from GATE (Cunningham et al., 2011), e.g., "Mr.", "Mrs.", "Dr.", "President", etc. for the type *Person*, "city", "river", "park", etc. for the type *Location*, "company", "institute", "Inc.", "Ltd.", etc. for the type *Organization*.

Context Statistics: To compute values for the features described in Section 4, we obtain statistics for different kinds of *contexts* for tokens and candidate entities. We distinguish two kinds of contexts: i) based on other tokens, and ii) based on patterns of parsing dependencies; these are represented as a bag-of-words or a bag-of-linguistic-patterns, respectively, each with tf-idf scores. Here tf denotes token frequencies and idf denotes inverse document frequencies (Croft et al., 2009). The tf values are obtained from the input document at hand; the idf values are estimated from a large Wikipedia text dump.

Token Context of Tokens. For a given token tok_i, all tokens tok_j in the same input document form the token context and are associated with their tf-idf scores. Thus, all tokens in the same document have identical token contexts.

Linguistic Context of Tokens. For all parsing dependencies, in which a token tok_i occurs, we treat the dependency type and the other argument of the dependency as linguistic patterns and compute their frequencies in the document (tf) and their inverse document frequencies in the Wikipedia corpus (idf). In the example sentence "*David played for manu, real, and la galaxy*", the linguistic context of the token "manu" consists of the Stanford parser dependencies ***prep_for****[played, manu]*, ***conj_and****[manu, real]*, and ***conj_and****[manu, galaxy]*. This leads to the patterns ***prep_for****[played, ∗]*, ***conj_and****[∗, real]*, and ***conj_and****[∗, galaxy]* with wildcards ∗, for which we compute tf-idf scores.

Token Context of Entities. For each candidate entity ent_i, we extract all tokens from keyphrases associated with ent_i and compute tf-idf scores for these tokens. The keyphrases are distilled from Wikipedia link anchor texts (Hoffart et al., 2011) and are part of the YAGO2 knowledge base. For example, the entity David Beckham has keyphrases such as "player of the year", "champions league final", "Manchester United", etc. Thus, we compute tf-idf statistics for tokens like "player", "year", "champions", etc.

Linguistic Context of Entities. For a candidate entity ent_i, we extract all linguistic patterns (with tf-idf scores) where ent_i occurs from the Wikipedia corpus. The result of parsing the Wikipedia corpus by a dependency tool is saved in our database.

3.3 Mention Candidates & Entity Candidates

To determine the candidate mentions in a token sequence, we first perform exact-match lookups of all sub-sequences against the name-entity dictionary. As an option (and by default), this can be limited to sub-sequences that are tagged as noun phrases by the Stanford parser. For higher recall, we then add partial-match lookups where a token sub-sequence matches only some but not all tokens of an entity name in the dictionary. For example, for the sentence "*David played for manu, real and la galaxy*", we obtain "David", "manu", "real", "la galaxy", "la", and "galaxy" as candidate mentions. This process yields features; our CRF model then learns how to determine the actual mention boundaries.

The entity candidates then are simply all entities that are associated with at least one of the candidate mentions. As we include highly ambiguous mentions and the knowledge base contains thousands of candidate entities for some mentions, we use pruning heuristics to restrict the candidate space. For each candidate mention, we consider only the top k (using $k = 20$ in our experiments) highest ranked entities. The ranking is based on the string similarity between the mention and the primary entity name, the prior popularity of the entity, and the local context similarity (feature functions f_8, f_9, f_{10} in Section 4).

For each candidate mention, we add a virtual entity `out-of-kb` (out of knowledge base) to the entity candidate space, to prepare for the possible situation that the mention actually denotes an emerging or long-tail entity that is not contained in the knowledge base at all. We compute the token context of a mention-specific `out-of-kb` entity, for a given mention token tok_i, based on the method of (Hoffart et al., 2014). First, we form the set union of the token contexts of all candidate entities for tok_i, and subtract this union from the token context of tok_i. Second, we compute tf-idf scores for the remaining tokens, using the idf estimates from Wikipedia (as for all other tokens) and

setting tf to 1 (as the `out-of-kb` entity is not observable).

4 Feature Space

We define feature templates f_1–f_{17} for computing the NER type and the NED label of a token tok_i that denotes or is part of a mention. The boundaries of a mention, i.e., its token span, are then trivially derived by combining adjacent tokens with the same NER type label (and disregarding all tokens with label "other").

Let pos_i be the POS tag of tok_i, dic_i is the NER tag from the dictionary lookup of tok_i, and dep_i is the parsing dependency that connects tok_i with another token. We write $sur_i = \langle tok_{i-1}, tok_i, tok_{i+1} \rangle$ to refer to the sequence of tokens surrounding tok_i. If lemmatization is enabled, tok_i can be replaced by lem_i. Next, let C_i be the set of candidate entities for all possible mentions that contain token tok_i. Finally, let d be the domain which the input text is classified into (see Section 3)

For a given training corpus (e.g., the CoNLL-YAGO2 training set), the feature templates are expanded into concrete features, considering also background dictionaries and knowledge-base statistics. Some of these are Boolean features, others are real-valued. The Boolean features $(f_2, f_3, f_4, f_5, f_6, f_7, f_{14}, f_{15}, f_{16})$ capture the presence or absence of features like tokens, POS tags, dependency types, dictionary entries in a given input document on which we want to run end-to-end NERD. The real-valued features $(f_1, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{17})$ capture similarity and relatedness measures between tokens, domains, or entities; these measures are precomputed on background resources like dictionaries and knowledge bases. These features are particularly crucial for NED, which needs to cope with many thousands of possible output labels (the entities for the mentions in a text).

4.1 Standard Features

Token-Type Prior. This feature f_1 captures a prior probability of tok_i being of NER type $type_j$. These probabilities are estimated from NER-annotated corpora. In our experiments, we use the training subsets of different test corpora, where training and test data are disjoint. For example, we may have a prior $f_1(\text{“}Ltd.\text{”}, \textit{ORG}) = 0.8$.

Current POS. The feature template $f_2(tok_i, pos_i, type_j)$ generates binary features for all training-corpus tokens tok_i with part-of-speech tag pos_i and NER label $type_j$. For the same token, multiple features may be generated: for example, if token "real" occurs in the training corpus in the phrase "real (*JJ*) madrid (*NN*)", this generates features $f_{2_1}(tok_i, \textit{JJ}, \textit{ORG})$ and $f_{2_2}(tok_i, \textit{JJ}, \textit{LOC})$, etc. When later observing token tok_i in an input document, the feature $(tok_i, pos_i, type_j)$ is set to 1 if tok_i has POS tag pos_i and $type_j$ is one of the corresponding NER types in the training corpus. In the rest of this section, binary features are generated from feature templates analogously.

In-Dictionary. The feature template $f_3(tok_i, dic_i, type_j)$ generates binary features if tok_i is in the name-entity dictionary for some entity of NER type $type_j$.

Uppercase. The feature template $f_4(tok_i, type_j)$ generates binary features if tok_i appears in uppercase form and is NER-labeled as $type_j$ in the training corpus.

Surrounding POS. The feature template $f_5(tok_i, sur_i, type_j)$ generates binary features if token tok_i and the POS tag sequence of its surrounding tokens sur_i appear in the training corpus where tok_i has NER label $type_j$.

Surrounding Tokens. The feature template $f_6(tok_i, sur_i, type_j)$ generates binary features if token tok_i has NER type $type_j$ given that tok_i appears with surrounding tokens sur_i in an NER-annotated training corpus. This could possibly lead to a huge number of features. For tractability, we thus ignore sequences that only occur once in the training corpus.

Surrounding In-Dictionary. The feature template $f_7(tok_i, sur_i, type_j)$ performs dictionary lookups for surrounding tokens in sur_i. Similar to f_6, it generates binary features if tok_i and the dictionary lookup sequence of its surrounding tokens sur_i appear in the training corpus where tok_i has NER type $type_j$.

Token-Entity Prior. The real-valued feature f_8 captures a prior probability of tok_i being entity ent_j. The probabilities are estimated from the occurrence frequencies of name-entity pairs in the background corpus, harnessing link anchor texts in Wikipedia. For example, we may have a prior $f_8(\text{“}Beckham\text{”}, \texttt{David\_Beckham}) = 0.7$ as he is more popular (today) than his wife Victoria. On the other hand, $f_8(\text{“}David\text{”}, \texttt{David\_Beckham})$ would be

lower than $f_8(\text{“}David\text{”}, \texttt{David\_Bowie})$, for example, as this still active pop star is more frequently and prominently mentioned than the retired football player.

Token-Entity n-Gram Similarity. The real-valued feature f_9 measures the Jaccard similarity of character-level n-grams a name in the dictionary that includes tok_i and the primary (i.e., full and most frequently used) name of an entity ent_j. For example, for $n = 2$ the value of $f_9(\text{“}Becks\text{”}, \texttt{David\_Beckham})$ is $\frac{3}{11}$. In experiments we set $n = 3$.

Token-Entity Token Contexts. The real-valued feature f_{10} measures the weighted overlap similarity between the token contexts (*tok-cxt*) of token tok_i and entity ent_j. We use a weighted generalization of the standard overlap coefficient, *WO*, between two sets X, Y of weighted elements, X_k and Y_k.

$$WO(X,Y) = \frac{\sum_k \min(X_k, Y_k)}{\min(\sum_k X_k, \sum_k Y_k)}$$

For our setting of token contexts, the weights are tf-idf scores, hence:

$$f_{10}(tok_i, ent_j) = WO\Big(\text{tok-cxt}(tok_i), \text{tok-cxt}(ent_j)\Big)$$

Entity-Entity Token Coherence. The real-valued feature f_{11} measures the coherence between the token contexts of two entity candidates ent_i and ent_j.

$$f_{11}(tok_i, ent_j) = WO\Big(\text{tok-cxt}(ent_i), \text{tok-cxt}(ent_j)\Big)$$

For example, entities David_Beckham and Manchester_United are highly coherent as they share many tokens in their contexts, such as “champions”, “league”, “premier”, “cup”, etc. Thus, they should be chosen together.

4.2 Domain Features

We use WordNet *domains*, created by (Miller, 1995; Magnini and Cavagli, 2000; Bentivogli et al., 2004), to construct a hierarchical taxonomy of 46 domains such as *Politics*, *Economy*, *Sports*, *Science*, *Medicine*, *Biology*, *Art*, *Music*, etc. We combine the domains with semantic *types* (classes of entities) provided by YAGO2, by assigning them to their respective domains. This is based on the manual assignment of WordNet synsets to domains by (Magnini and Cavagli, 2000; Bentivogli et al., 2004), and extends to additional types in YAGO2. For example, *Singer* is assigned to *Music*, and *Football Player* to *Football*, a sub-domain of *Sports*. These types include the standard NER types *Person* (*PERS*), *Organization* (*ORG*), *Location* (*LOC*), and *Miscellaneous* (*MISC*) which are further refined by the YAGO2 *subclassOf* hierarchy. In total, the 46 domains are enhanced with ca. 350,000 types imported from YAGO2.

J-NERD classifies input texts onto domains by means of “*easy mentions*”. An easy mention is a match in the name-entity dictionary for which there are at most three candidate entities (Nguyen et al., 2014). Although the mention set is not explicitly given before running NERD, *J*-NERD still can extract “*easy mentions*” from the entirety of all mention candidates. Let C^* be the set of candidate entities for easy mentions in the input document. For each domain d, we compute the *coherence* of the easy mentions $M^* = \{m_1, m_2, \dots\}$

$$\text{coh}(M^*) = \frac{|C^* \cap C^d|}{|C^*|}$$

where C^d is the set of all entities under domain d. We classify the document into the domain with the highest coherence score.

Although, the mentions and their entities may be inferred incorrectly, the domain classification still tends to work very reliably as it aggregates over all “easy” mention candidates. The following feature templates exploit domains.

Entity-Domain Coherence. This feature captures the coherence between an entity candidate ent_j and the domain d which the input text is classified into. That is, $f_{12}(d, ent_j) = 1$ if $d \in \text{dom}(ent_j)$. Otherwise, the feature value is 0.

Entity-Entity Type Coherence. This feature computes the relatedness between the Wikipedia categories of two candidate entities $ent_i \in C_i$, $ent_j \in C_j$.

$$f_{13}(ent_i, ent_j) = \max_{\substack{c_u \in \text{cat}(ent_i) \\ c_v \in \text{cat}(ent_j)}} \text{rel}(c_u, c_v)$$

where the function $\text{rel}(c_u, c_v)$ computes the reciprocal length of the shortest path between categories c_u, c_v in the domain taxonomy (Nguyen et al., 2014). Recall that our domain taxonomy contains hundred thousands of Wikipedia categories integrated in the YAGO2 type hierarchy.

4.3 Linguistic Features

Linguistic Pattern from Dependency Parsing. Recall that we obtain dependency-parsing patterns by using Wikipedia as a large background corpus. Here we harness that Wikipedia contains many mentions with explicit links to entities and that the

knowledge base provides us with the NER types for these entities.

Typed-Dependency. The feature template $f_{14}(tok_i, dep_i, type_j)$ generates binary features if the background corpus contains the pattern $dep_i = deptype(arg1, arg2)$ where tok_i is either $arg1$ or $arg2$ and tok_i is labeled with NER type $type_j$. For example, with token "manu" in our example sentence "*David played for manu, real, and la galaxy*", a feature generated from a similar sentence in Wikipedia would be $f_{14_1}(tok_i, \textbf{\textit{prep\_for}}(\text{“}played\text{”}, *), ORG)$.

Typed-Dependency/POS. This feature template $f_{15}(tok_i, pos_i, dep_i, type_j)$ captures linguistic patterns that combine parsing dependencies (like in f_{14}) and POS tags (like in f_2), learned from an annotated training corpus. It generates binary features if tok_i appears in the dependency pattern dep_i with POS tag pos_i and this configuration also occurs in the training data with NER label $type_j$.

Typed-Dependency/In-Dictionary. The feature template $f_{16}(tok_i, dic_i, dep_i, type_j)$ captures linguistic patterns that combine parsing dependencies (like in f_{14}) and dictionary lookups (like in f_3), learned from an annotated training corpus. It generates binary features if tok_i appears in the dependency pattern dep_i and has an entry dic_i in the name-entity dictionary for some entity with NER label $type_j$.

Token-Entity Linguistic Contexts. The real-valued feature f_{17} measures the weighted overlap between the linguistic contexts (*ling-cxt*) of token tok_i and entity ent_j.

$$f_{17}(tok_i, ent_j) = WO\Big(\text{ling-cxt}(tok_i), \text{ling-cxt}(ent_j)\Big)$$

5 Graphical Models

The CRF models that *J*-NERD uses for its inference are initially constructed on a per-sentence basis. These local CRF's are then combined into a global model by adding *non-local links* to capture cross-sentence dependencies (Finkel et al., 2005) among mentions in different sentences (as illustrated in Figure 3).

5.1 Linear-Chain Model

In the *local setting*, *J*-NERD works on each sentence $S = \langle tok_1, \ldots, tok_m \rangle$ separately. We construct a linear-chain CRF (see Figure 1) by introducing an observed variable x_i for each token tok_i that represents a proper word. For each x_i, we additionally introduce a variable y_i representing the combined NERD labels. As in any CRF, the x_i,y_i and y_i,y_{i+1} pairs are connected via factors, whose weights we obtain from the feature functions described in Section 4.

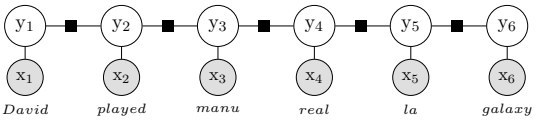

Figure 1: Linear-chain CRF.

5.2 Tree Model

The factor graph for the tree model (see Figure 2) extends the linear-chain model by adding a factor linking each pair of labels y_i,y_j whenever these tokens have a typed dependency obtained from the Stanford parser. Figure 2 shows an example for the tree-shaped model.

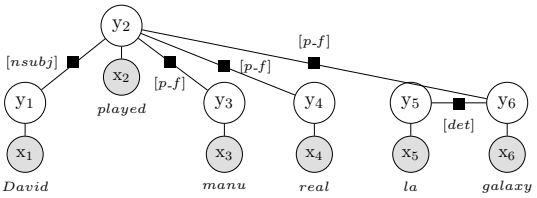

Figure 2: Tree model. ($[p\_f]$ is $[prep\_for]$)

5.3 Global Models

Following (Finkel et al., 2005) for the *global setting*, we consider an entire input text, consisting of multiple sentences $S_1, \ldots, S_n = \langle tok_1, \ldots, tok_m \rangle$, for the construction of both the *linear-chain model* and the *tree model*. As shown in Figure 3, cross-sentence edges among pairs of labels y_i,y_j are introduced for candidate sets C_i,C_j that share at least one candidate entity, such as "David" and "David Beckham". Additionally, we introduce factors for all pairs of tokens in adjacent mentions within the same sentence, such as "David" and "manu".

5.4 Inference & Learning

For a given sequence of observed variables $\langle x_1, \ldots, x_m \rangle$, let L denote a sequence of NERD labels $\langle y_1, \ldots, y_m \rangle$, where each y_i consists of an NER type $type_i$ and an NED label ent_i. Our *inference objective* is to find the most probable sequence L^*. This goal is expressed by the following function: $L^* = \arg\max_{y_1 \ldots y_m}$

$$\exp\left(\sum_{t=1}^{m}\sum_{k=1}^{K} \lambda_k \, feature_k(y_t, y_{prev(t)}, x_1 \ldots x_m)\right)$$

where

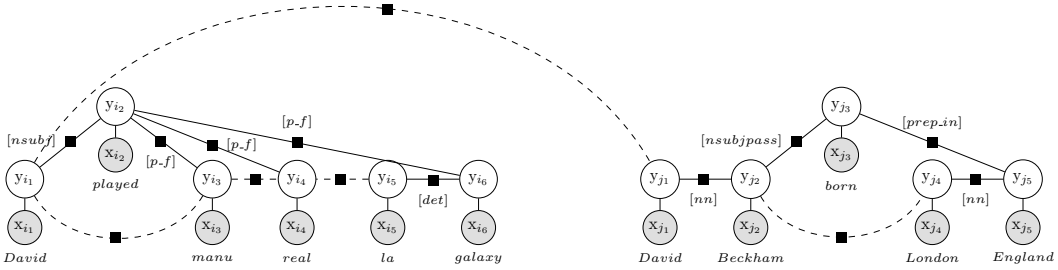

Figure 3: Global model, linking two tree models. ($[p\_f]$ is $[prep\_for]$)

- $feature_{1..K}$ are features generated from feature templates $f_{1..17}$ of Section 4,
- $prev(t)$ is the index of the label y_j that y_t depends on, i.e., the previous index $(t-1)$ in linear-chain models or the parent index in tree models,
- and λ_k are feature weights, i.e., model parameters to be learned.

The number of generated features, K, depends on the training corpus and the choice of the CRF model. For the CoNLL-YAGO2 training set, the tree models have $K = 1,767$ parameters. Given a trained model, exact inference with respect to the above objective function can be efficiently performed by variants of the Viterbi algorithm (Sutton and McCallum, 2012) for the local models, both linear and tree models. For the global models, however, exact solutions are computationally intractable. Therefore, we employ Gibbs sampling to approximate the solution.

As for model parameters, *J*-NERD learns the feature weights λ_k from the training data by maximizing a respective conditional likelihood function (Sutton and McCallum, 2012), using a variant of the L-BFGS optimization algorithm (Liu and Nocedal, 1989). We do this for each local CRF model (linear and tree models), and apply the same learned weights to the corresponding global models. Our implementation uses the RISO toolkit[1] for belief networks.

6 Experiments

6.1 Setup

6.1.1 Test Data Collections

Our evaluation is mainly based on the CoNLL-YAGO2 corpus of newswire articles. Additionally, we report on experiments with an extended version of the ACE-2005 corpus and a large sample of the entity-annotated ClueWeb'09-FACC1 Web crawl.

CoNLL-YAGO2 is derived from the CoNLL-YAGO corpus (Hoffart et al., 2011)[2] by removing tables where mentions in table cells do not have linguistic context, a typical example being sports results. The resulting corpus contains 1,244 documents with 20,924 mentions including 4,774 `out-of-kb` entities. Ground-truth entities in YAGO2 are provided by (Hoffart et al., 2011). For consistent ground-truth, we derived the NER types from the NED ground-truth entities, this way fixing some errors in the original annotations related to metonymy (e.g., labeling the mentions in "*India beats Pakistan 2:1*" incorrectly as *LOC*, whereas the entities are the sports teams of type *ORG*). This makes the dataset not only cleaner but also more demanding, as metonymous mentions are among the most difficult cases.

For the evaluation we use the "testb" subset of CoNLL-YAGO, which – after the removal of tables – has 231 documents with 5,616 mentions including 1,131 `out-of-kb` entities. The other 1,045 documents with a total of 17,870 mentions (including 4,057 `out-of-kb` mentions) are used for training.

ACE is an extended variant of the ACE 2005 corpus[3], with additional NED labels by (Bentivogli et al., 2010). We consider only proper entities and exclude mentions of general concepts such as "revenue", "world economy", "financial crisis" etc., as they do not correspond to individual entities in a knowledge base. This reduces the number of mentions, but gives the task a crisp focus. We disallow overlapping mention spans and consider only maximum-length mentions, following the rationale of the ERD Challenge 2014. The test set contains 117 documents with 2,958 mentions.

ClueWeb contains two randomly sampled subsets of the ClueWeb'09-FACC1[4] corpus with Freebase annotations:

[1] `http://riso.sourceforge.net/`

[2] `https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/aida/downloads/`

[3] `http://projects.ldc.upenn.edu/ace/`

[4] `http://lemurproject.org/clueweb09/FACC1/`

- **ClueWeb**: 1,000 documents (24,289 mentions) each with at least 5 entities .
- **ClueWeb**$_{long-tail}$: 1,000 documents (49,604 mentions) each with at least 3 long-tail entities.

We consider an entity to be "long-tail" if it has at most 10 incoming links in the English Wikipedia. Note that these web documents are very different in style from the news-centric articles in CoNLL and ACE. Also note that the entity markup is automatically generated, but with emphasis on high precision. So the data captures only a small subset of the potential entity mentions, and it may contain a small fraction of false entities.

In addition to these larger test corpora, we ran experiments with several small datasets used in prior works: KORE (Hoffart et al., 2012), MSNBC (Cucerzan, 2007), and a subset of AQUAINT (Milne and Witten, 2008). Each of these has only a few hundred mentions, but they exhibit different characteristics. The findings on these datasets are fully in line with those of our main experiments; hence no explicit results are presented here.

In all these test datasets, the ground-truth considers only individual entities and excludes general concepts, such as "climate change", "harmony", "logic", "algebra", etc. These proper entities are identified by the intersection of Wikipedia articles and YAGO2 entities. This way, we focus on NERD. Systems that are designed for the broader task of "Wikification" are not penalized by their (typically lower) performance on inputs other than proper entity mentions.

6.1.2 Methods under Comparison

We compare ***J*-NERD** in its four variants (*linear* vs. *tree* and *local* vs. *global*) to various state-of-the-art NER/NED methods.

For NER (i.e., mention boundaries and types) we use the recent version 3.4.1 of the Stanford NER Tagger[5] (Finkel et al., 2005) as a baseline. This version has NER benchmark results on CoNLL'03 that are as good as those reported in Ratinov and Roth (2009) and Passos et al. (2014). We *retrained* this model by using the same corpus-specific training data that we use for *J*-NERD .

For NED, we compared *J*-NERD against the following methods for which we obtained open-source software or could call a web service:

- **Berkeley-entity** (Durrett and Klein, 2014) is a joint model for coreference resolution, NER and NED with linkage to Wikipedia.
- **AIDA-light** (Nguyen et al., 2014) is an optimized variant of the AIDA system (Hoffart et al., 2011), based on YAGO2. It uses the Stanford tool for NER.
- **TagMe** (Ferragina and Scaiella, 2010) is a Wikifier that maps mentions to entities or concepts in Wikipedia. It uses a Wikipedia-derived dictionary for NER.
- **Spotlight** (Mendes et al., 2011) links mentions to entities in DBpedia. It uses the LingPipe dictionary-based chunker for NER.

Some systems use confidence thresholds to decide on when to map a mention to `out-of-kb`. For each dataset, we used withheld data to tune these system-specific thresholds. Figure 4 illustrates the sensitivity of the thresholds for the CoNLL-YAGO2 dataset.

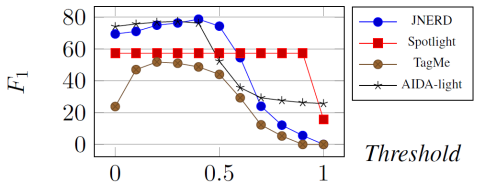

Figure 4: F_1 for varying confidence thresholds.

6.1.3 Evaluation Measures

We evalute the output quality at the NER level alone and for the end-to-end NERD task. We do not evaluate NED alone, as this would require giving a ground-truth set of mentions to the systems under test to rule out that NER errors affect NED. Most competitors do not have interfaces for such a controlled NED-only evaluation.

Each test collection has ground-truth annotations (G) consisting of text spans for mentions, NER types of the mentions, and mapping mentions to entities in the KB or to `out-of-kb`. Recall that the `out-of-kb` case captures entities that are not in the KB at all. Let X be the output of system X: detected mentions, NER types, NED mappings. Following the ERD 2014 Challenge (Carmel et al., 2014), we define precision and recall of X for end-to-end NERD as:

$$Prec(X) = \frac{|X \text{ agrees with } G|}{|X|}$$

$$Rec(X) = \frac{|X \text{ agrees with } G|}{|G|}$$

where agreement means that X and G overlap in the text spans (i.e. have at least one token

[5] `nlp.stanford.edu/software/CRF-NER.shtml`

in common) for a mention, have the same NER type, and have the same mapping to an entity or `out-of-kb`. The F_1 score of X is the harmonic mean of precision and recall.

For evaluating mention boundary detection alone, we consider only the overlap of text spans; for evaluating NER completely, we consider both mention overlap and agreement on NER type.

6.2 Results for CoNLL-YAGO2

Our first experiment on CoNLL-YAGO2 is comparing the four CRF variants of *J*-NERD for three tasks: mention boundary detection, NER typing and end-to-end NERD. Then, the best model of *J*-NERD is compared against various baselines and a pipelined configuration of our method. Finally, we test the influence of different features groups.

6.2.1 Experiments on CRF Variants

Table 1: Experiments on CoNLL-YAGO2.

Perspective	Variants	Prec	Rec	F_1
Mention Boundary Detection	*J*-NERD$_{linear\text{-}local}$	94.2	89.6	91.8
	J-NERD$_{tree\text{-}local}$	94.4	89.4	91.8
	J-NERD$_{linear\text{-}global}$	95.1	90.3	92.6
	J-NERD$_{tree\text{-}global}$	**95.8**	**90.6**	**93.1**
NER Typing	*J*-NERD$_{linear\text{-}local}$	87.8	83.0	85.3
	J-NERD$_{tree\text{-}local}$	89.5	82.2	85.6
	J-NERD$_{linear\text{-}global}$	88.6	83.4	85.9
	J-NERD$_{tree\text{-}global}$	**90.4**	**83.8**	**86.9**
End-to-End NERD	*J*-NERD$_{linear\text{-}local}$	71.8	74.9	73.3
	J-NERD$_{tree\text{-}local}$	75.1	74.5	74.7
	J-NERD$_{linear\text{-}global}$	77.6	74.8	76.1
	J-NERD$_{tree\text{-}global}$	**81.9**	**75.8**	**78.7**

Table 1 shows that all variants perform very well on boundary detection and NER typing, with small differences only. For end-to-end NERD, however, *J*-NERD$_{tree\text{-}global}$ outperforms all other variants by a large margin. This results in achieving the best F_1 score of 78.7%, which is 2.6% higher than *J*-NERD$_{linear\text{-}global}$. We performed a paired t-test between these two variants, and obtained a p-value of 0.01. The local variants of *J*-NERD lose around 4% of F_1 because they do not capture the coherence among mentions in different sentences.

In the rest of our experiments, we therefore focus on *J*-NERD$_{tree\text{-}global}$ and the task of end-to-end NERD.

6.2.2 Comparison of Joint vs. Pipelined Models and Baselines

In this subsection, we demonstrate the benefits of joint models against pipelined models including state-of-the-art baselines. In addition to the competitors introduced in 6.1.2, we add a pipelined configuration of *J*-NERD , named *P*-NERD. That is, we first run *J*-NERD in NER mode (only considering NER features $f_{1..7}$ and $f_{14..16}$). The best sequence of NER labels is then given to *J*-NERD to run in NED mode (only considering NED features $f_{8..13}$ and f_{17}).

Table 2: Comparison between joint models and pipelined models on end-to-end NERD.

Method	Prec	Rec	F_1
P-NERD	80.1	75.1	77.5
J-NERD	**81.9**	75.8	**78.7**
AIDA-light	78.7	**76.1**	77.3
TagMe	64.6	43.2	51.8
SpotLight	71.1	47.9	57.3

The results are shown in Table 2. *J*-NERD achieves the highest precision of 81.9% for end-to-end NERD, outperforming all competitors by a large margin. This results in achieving the best F_1 score of 78.7%, which is 1.2% higher than *P*-NERD and 1.4% higher than AIDA-light. Note that (Nguyen et al., 2014) reported higher precision for AIDA-light, but that experiment did not consider `out-of-kb` entities which pose an extra difficulty in our setting. Also, the test corpora – CoNLL-YAGO2 vs. CoNLL-YAGO – are not quite comparable (see above).

TagMe and Spotlight are clearly inferior on this dataset (more than 20% lower in F_1 than *J*-NERD). It seems these systems are more geared for efficiency and coping with popular and thus frequent entities, whereas the CoNLL-YAGO2 dataset contains very difficult test cases.

For the best F_1 score of *J*-NERD, we performed a paired t-test against the other methods' F_1 values and determined a p-value of 0.075.

We also compared the NER performance of *J*-NERD against the state-of-the-art method for NER alone, the Stanford NER Tagger version 3.4.1. For mention boundary detection, *J*-NERD achieved an F_1 score of 93.1% versus 93.4% by Stanford NER and 92.9% by *P*-NERD. For NER typing, *J*-NERD achieved an F_1 score of 86.9% versus 86.8% by Stanford NER and 86.3% by *P*-NERD. So we could not outperform the best prior method for NER alone, but achieved very competitive results. Here, we do not really leverage any form of joint inference (combining CRF's across sentences is used in Stanford NER, too), but harness rich features on domains, entity candidates, and linguistic dependencies.

6.2.3 Influence of Features

To analyze the influence of the features, we performed an additional ablation study on the global *J*-NERD tree model, which is the best variant of *J*-NERD , as follows:

- *Standard features* only include features introduced in Section 4.1.
- *Standard and domain features* exclude the linguistic features $f_{14}, f_{15}, f_{16}, f_{17}$.
- *Standard and linguistic features* excludes the domain features f_{12} and f_{13}.
- *All features* is the full-fledged *J*-NERD$_{tree\text{-}global}$ model.

Table 3: Feature Influence on CoNLL-YAGO2.

Perspective	Setting	F_1
NER Typing	Standard features	85.1
	Standard and domain features	85.7
	Standard and linguistic features	86.4
	All features	**86.9**
End-to-End NERD	Standard features	74.3
	Standard and domain features	76.4
	Standard and linguistic features	76.6
	All features	**78.7**

Table 3 shows the results, demonstrating that linguistic features are crucial for both NER and NERD. For example, in the sentence *"Woolmer played 19 tests for England"*, the mention "England" refers to an organization (the English cricket team), not to a location. The dependency-type feature ***prep_for****[play, England]* is a decisive cue to handle such cases properly. Domain features help in NED to eliminate, for example, football teams when the domain is cricket.

6.3 End-to-End NERD on ACE

For comparison to the recently developed Berkeley-entity system (Durrett and Klein, 2014), the authors of that system provided us with detailed results for the entity-annotated ACE'2005 corpus, which allowed us to discount non-entity (so-called "*NOM-type*") mappings (see Subsection 6.1.1). All other systems, including the best *J*-NERD method, were run on the corpus under the same conditions.

Table 4: NERD results on ACE.

Method	Prec	Rec	F_1
P-NERD	68.2	60.8	64.2
J-NERD	**69.1**	**62.3**	**65.5**
Berkeley-entity	65.6	61.8	63.7
AIDA-light	66.8	59.3	62.8
TagMe	60.6	43.5	50.7
SpotLight	68.7	29.6	41.4

J-NERD outperforms *P*-NERD and Berkeley-entity: F_1 scores are 1.3% and 1.8% better, respectively, with a t-test p-value of 0.05 (Table 4). Following these three best-performing systems, AIDA-light also achieves decent results. The other systems show substantially inferior performance.

The performance gains that *J*-NERD achieves over Berkeley-entity can be attributed to two factors. First, the rich linguistic features of *J*-NERD help to correctly cope with some difficult cases, e.g., when common nouns are actually names of people. Second, the coherence features of global *J*-NERD help to properly couple decisions on related entity mentions.

6.4 End-to-End NERD on ClueWeb

The results for ClueWeb are shown in Table 5. Again, *J*-NERD outperforms all other systems with a t-test p-value of 0.05. The differences between *J*-NERD and fast NED systems such as TagMe or SpotLight become smaller as the number of prominent entities (i.e., prominent people, organizations and locations) is higher on ClueWeb than on CoNLL-YAGO2.

Table 5: NERD results on ClueWeb.

Dataset	Method	Prec	Rec	F_1
ClueWeb	*P*-NERD	80.9	67.1	73.3
	J-NERD	**81.5**	**67.5**	**73.8**
	AIDA-light	80.2	66.4	72.6
	TagMe	78.4	60.5	68.3
	SpotLight	79.7	57.1	66.5
ClueWeb$_{long-tail}$	*P*-NERD	81.2	64.4	71.8
	J-NERD	**81.4**	**65.1**	**72.3**
	AIDA-light	81.2	63.7	71.3
	TagMe	78.4	58.3	66.9
	SpotLight	81.2	56.3	66.5

7 Conclusions

We have shown that coupling the tasks of NER and NED in a joint CRF model is beneficial. Our *J*-NERD method outperforms strong baselines on a variety of test datasets. The strength of *J*-NERD comes from three novel assets. First, our tree CRF's capture the structure of dependency parse trees, and we couple multiple of such tree models across sentences. Second, we harness non-standard features about domains and novel features for linguistic patterns derived from parsing. Third, our joint inference maintains uncertain candidates for both mentions and entities and makes decisions as late as possible. In our future work, we plan to explore use cases for joint NERD, especially for content analytics over news streams and social media.

References

Luisa Bentivogli, Pamela Forner, Bernardo Magnini, and Emanuele Pianta. 2004. Revising the Wordnet Domains Hierarchy: Semantics, Coverage and Balancing. In *Proceedings of the Workshop on Multilingual Linguistic Ressources, MLR '04*, pages 101–108. ACL.

Luisa Bentivogli, Pamela Forner, Claudio Giuliano, Alessandro Marchetti, Emanuele Pianta, and Kateryna Tymoshenko. 2010. Extending English ACE 2005 Corpus Annotation with Ground-truth Links to Wikipedia. In *The People's Web Meets NLP: Collaboratively Constructed Semantic Resources '10*, pages 19–27. COLING.

Razvan Bunescu and Marius Pasca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *EACL '06*, pages 9–16. ACL.

David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. 2014. ERD'14: Entity Recognition and Disambiguation Challenge. In *SIGIR '14*, page 1292. ACM.

Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A Framework for Benchmarking Entity-annotation Systems. In *WWW '13*, pages 249–260. ACM.

Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Hinrich Schütze, and Stefan Rüd. 2014. The SMAPH System for Query Entity Recognition and Disambiguation. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation, ERD '14*, pages 25–30. ACM.

Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company.

Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *EMNLP-CoNLL '07*, pages 708–716. ACL.

Silviu Cucerzan. 2014. Name Entities Made Obvious: The Participation in the ERD 2014 Evaluation. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation, ERD '14*, pages 95–100. ACM.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, et al. 2011. *Text Processing with GATE*. University of Sheffield.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *Proceedings of the 9th International Conference on Semantic Systems, I-SEMANTICS '13*, pages 121–124. ACM.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC '06*, pages 449–454. ELRA.

Stephen Dill, Nadav Eiron, David Gibson, et al. 2003. SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. In *WWW '03*, pages 178–186. ACM.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. In *TACL '14*. ACL.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *CIKM '10*, pages 1625–1628. ACM.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *ACL '05*, pages 363–370. ACL.

Michael Fleischman and Eduard Hovy. 2002. Fine Grained Classification of Named Entities. In *COLING '02*, pages 1–7. ACL.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *EMNLP '11*, pages 782–792. ACL.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: Keyphrase Overlap Relatedness for Entity Disambiguation. In *CIKM '12*, pages 545–554. ACM.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *AI vol. 194*, pages 28–61.

Johannes Hoffart, Yasemin Altun, and Gerhard Weikum. 2014. Discovering Emerging Entities with Ambiguous Names. In *WWW '14*, pages 385–396. ACM.

Daphne Koller, Nir Friedman, Lise Getoor, and Benjamin Taskar. 2007. Graphical Models in a Nutshell. In *An Introduction to Statistical Relational Learning*. MIT Press.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective Annotation of Wikipedia Entities in Web Text. In *KDD '09*, pages 457–466. ACM.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained Entity Recognition. In *AAAI '12*. AAAI Press.

Dong C. Liu and Jorge Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming vol. 45*, pages 503–528.

Bernardo Magnini and Gabriela Cavagli. 2000. Integrating subject field codes into wordnet. In *Proceedings of the international conference on Language resources and evaluation, LREC '00*, pages 1413–1418. ELRA.

Andrew McCallum and Wei Li. 2003. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *HLT-NAACL '03*, pages 188–191. ACL.

Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. 2012. Adding Semantics to Microblog Posts. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM '12*, pages 563–572. ACM.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems, I-SEMANTICS '11*, pages 1–8. ACM.

George A. Miller. 1995. Wordnet: A lexical database for english. pages 39–41. ACM.

David Milne and Ian H. Witten. 2008. Learning to Link with Wikipedia. In *CIKM '08*, pages 509–518. ACM.

David Milne and Ian H. Witten. 2013. An Open-source Toolkit for Mining Wikipedia. *Artificial Intelligence vol. 194*, pages 222–239.

Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained Semantic Typing of Emerging Entities. In *ACL '13*, pages 1488–1497. ACL.

Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. 2014. AIDA-light: High-Throughput Named-Entity Disambiguation. In *Proceedings of the Workshop on Linked Data on the Web, LDOW '14*. CEUR-WS.org.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL '14*, pages 78–86. ACL.

Altaf Rahman and Vincent Ng. 2010. Inducing Fine-grained Semantic Classes via Hierarchical and Collective Classification. In *COLING '10*, pages 931–939. ACL.

Lev Ratinov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *CoNLL '09*, pages 147–155. ACL.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *HLT '11*, pages 1375–1384. ACL.

Avirup Sil and Alexander Yates. 2013. Re-ranking for Joint Named-Entity Recognition and Linking. In *CIKM '13*, pages 2369–2374. ACM.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A Cross-Lingual Dictionary for English Wikipedia Concepts. In *Proceedings of the international conference on Language resources and evaluation, LREC '12*. ELRA.

Charles A. Sutton and Andrew McCallum. 2012. An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning vol. 4*, pages 267–373.

Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, et al. 2015. GERBIL – general entity annotation benchmark framework. In *WWW '15*. ACM.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical Type Classification for Entity Names. In *COLING '12*, pages 1361–1370. ACL.