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

Dat Ba Nguyen[1], **Martin Theobald**[2], **Gerhard Weikum**[1]
[1] Max Planck Institute for Informatics
[2] University of Ulm
{datnb,weikum}@mpi-inf.mpg.de
martin.theobald@uni-ulm.de

Abstract

Methods for Named Entity Recognition and Disambiguation (NERD) perform NER and NED in two separate stages. Therefore, NED may be penalized with respect to precision by NER false positives, and suffers in recall from NER 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 disambiguating "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.

Transactions of the Association for Computational Linguistics, vol. 4, pp. 215–229, 2016. Action Editor: Hwee Tou Ng.
Submission batch: 7/2015; Revision batch: 1/2016; 3/2016; Published 5/2016.

Our method, called *J*-NERD[1], is based on a supervised, non-linear graphical model that combines multiple per-sentence models into an entity-coherence-aware global model. The global model detects mention spans, tags them with coarse-grained types, and maps them to entities in a single joint-inference step based on the Viterbi algorithm (for exact inference) or Gibbs sampling (for approximate inference). The *J*-NERD method comprises the following novel contributions:

- a tree-shaped model 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 ***dobj*** 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.

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 and Scaiella (2010) and Mendes et al. (2011) are examples of dictionary-based NER. The work of Spitkovsky and Chang (2012) is an example of 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 in their first stage of detecting mentions.

Mention Typing: The specific NER task 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 these works are 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, those 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 work of Dill et al. (2003), Bunescu and Pasca (2006), Cucerzan (2007), and 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,

[1]The *J*-NERD source is available at the URL `http://download.mpi-inf.mpg.de/d5/tk/jnerd-tacl.zip`.

publicly available systems for news and web texts. Most of these methods combine contextual similarity measures with some form of consideration for the coherence among a selected set of candidate entities for 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), and Durrett and Klein (2014) are the most notable methods. Sil and Yates (2013) first compile a liberal set of mention and entity candidates, and then perform joint ranking of the candidates. Durrett and Klein (2014) present 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) the maintaining of 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 *J*-NERD Factor Graph Model

3.1 Overview

To label a sequence of *input tokens* $\langle x_1, \dots, x_m \rangle$ with a sequence of *output labels* $\langle y_1, \dots, y_m \rangle$, consisting of NER types and NED entities, we devise a family of linear-chain and tree-shaped *probabilistic graphical models* (Koller et al., 2007). We employ these models to compactly encode a multivariate probability distribution over *random variables* $\mathcal{X} \cup \mathcal{Y}$, where $\mathcal{X}$ denotes the set of input tokens x_i we may observe, and $\mathcal{Y}$ denotes the set of output labels y_i we may associate with these tokens. By writing $\mathbf{x}$, we denote an *assignment of tokens* to $\mathcal{X}$, while $\mathbf{y}$ denotes an *assignment of labels* to $\mathcal{Y}$. In our running example, "David" is the first token x_1 with the desired label y_1 = `PER:David_Beckham` where `PER` denotes the NER type *Person* and `David_Beckham` is the entity of interest. Consecutive tokens with identical labels are considered to be entity mentions. For example, for $x_5 = la$ and $x_6 = galaxy$, the output would ideally be $y_5 =$ `ORG:Los_Angeles_Galaxy` and $y_6 =$ `ORG:Los_Angeles_Galaxy`, denoting the soccer club. Upfront these are merely candidate labels, though. Our method may alternatively consider the labels $y_5 =$ `LOC:Los_Angeles` and $y_6 =$ `MISC:Samsung_Galaxy`. This would yield incorrect output with two single-token mentions and improper entities.

The *feature templates* f_1–f_{17} we describe in detail in Section 4 each take the possible assignments $\mathbf{x}, \mathbf{y}$ of tokens and labels, respectively, as input and give a binary value or real number as output. Binary values denote the presence or absence of a feature (e.g., a particular token); real-valued ones typically denote frequencies of observed features.

For tractability, probabilistic graphical models are typically constrained by making conditional independence assumptions, thus imposing structure and locality on $\mathcal{X} \cup \mathcal{Y}$. In our models, we postulate that the following conditional independence assumptions hold:

$$p(y_i \mid \mathbf{x}, \mathbf{y}) = p(y_i \mid \mathbf{x}, y_{prev(i)})$$

That is, the label y_i for the i^{th} token directly depends only on the label $y_{prev(i)}$ of some previous token at position $prev(i)$ and potentially on all input tokens. The case where $prev(i) = i - 1$ is the standard setting for a linear-chain CRF, where the label of a token depends only on the label of its preceding token. We generalize this approach to considering $prev(i)$ tokens based on the edges of a dependency parse tree and $prev(i)$ tokens derived from co-references in preceding sentences.

By the Hammersley-Clifford Theorem, such a graphical model can be factorized into a product form where each factor captures a subset $A \subseteq \mathcal{X} \cup \mathcal{Y}$ of the random variables. Typically, each factor considers only those $\mathcal{X}$ and $\mathcal{Y}$ variables that are coupled by a conditional (in-)dependence assumptions, with overlapping A sets of different factors. The probability distribution encoded by the graphical model can then be expressed as follows:

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \prod_{A} \mathcal{F}_A(\mathbf{x}_A, \mathbf{y}_A)$$

Here, $\mathcal{F}_A(\mathbf{x}_A, \mathbf{y}_A)$ denotes the *factors* of the model, each of which is of the following form:

$$\mathcal{F}_A(\mathbf{x}_A, \mathbf{y}_A) = \exp \left\{ \sum_k \lambda_k \, f_{A,k}(\mathbf{x}_A, \mathbf{y}_A) \right\}$$

The normalization constant $Z = \sum_{\mathbf{x},\mathbf{y}} \prod_A \mathcal{F}_A(\mathbf{x}_A, \mathbf{y}_A)$ ensures that this distribution sums up to 1, while λ_k are the parameters of the model, which we aim to learn from various annotated background corpora.

Our inference objective then is to find the *most probable sequence of labels* $\mathbf{y}^*$ when given the token sequence $\mathbf{x}$ as evidence:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y} \,|\, \mathbf{x})$$

That is, in our setting, we fix $\mathbf{x} = \langle tok_1, \ldots, tok_m \rangle$ to the observed token sequence, while $\mathbf{y} = \langle y_1, \ldots, y_m \rangle$ ranges over all possible sequences of associated labels. In our approach, which we hence coined *J*-NERD, each y_i label represents a combination of NER type and NED entity.

State-of-the-art NER methods, such as the Stanford NER Tagger, employ *linear-chain factor graph*, known as Conditional Random Fields (CRF's) (Sutton and McCallum, 2012). We also devise more sophisticated *tree-shaped factor graphs* whose structure is obtained from the dependency parse trees of the input sentences. These per-sentence models are optionally combined into a *global factor graph* by adding also cross-sentence dependencies (Finkel et al., 2005). These cross-sentence dependencies are added whenever overlapping sets of entity candidates (i.e., potential co-references) are detected among the input sentences. Figure 3 gives an example of such a global graphical model for two sentences.

The search space of candidate labels for our models depends on the candidates for mention spans (with the same NER type) and their NED entities. We use pruning heuristics to restrict this space: candidate spans for mentions are derived from dictionaries, and we consider only the top-20 entity candidates for each candidate mention. For a given sentence, this typically leads to a few thousand candidate labels over which the CRF inference runs. The candidates are determined independently for each sentence.

3.2 Features

These models employ a variety of feature templates that generate the factors of the joint probability distribution. Some of the features are fairly standard for NER/NED, whereas others are novel.

- 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 third feature group captures typed dependencies from the sentence parsing. To our knowledge, these have not been used in prior work on NER and NED.

The NER types that we consider are the standard types `PER` for person, `LOC` for location and `ORG` for organization. All other types that, for example, the Stanford NER Tagger would mark, are collapsed into a type `MISC` for miscellaneous. These include labels like date and money (which are not genuine entities anyway) and also entity types like events and creative works such as movies, songs, etc. (which are disregarded by the Stanford NER Tagger). We add two dedicated tags for tokens to express the case when no meaningful NER type or NED entity can be assigned. For tokens that should not be labeled as a named entity at all (e.g., "played" in our example), we use the tag `Other`. For tokens with a valid NER type, we add the virtual entity `Out-of-KB` (for "out of knowledge base") to its entity candidates, to prepare for the possible situation where the token (and its surrounding tokens) actually denotes an emerging or long-tail entity that is not contained in the knowledge base.

3.3 Linear-Chain Model

In the local models, *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 that represents the combined NERD label. As in any CRF, the x_i, y_i and y_i, y_{i+1} pairs are connected via factors $\mathcal{F}(\mathbf{x}, \mathbf{y})$, whose weights we obtain from the feature functions described in Section 4.

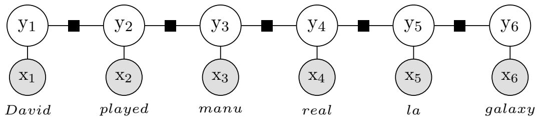

Figure 1: Linear-chain model (CRF).

3.4 Tree Model

The factor graph for the tree-shaped model is constructed in a similar way. However, here we add a factor that links a pair of labels y_i, y_j if their respective tokens tok_i, tok_j are connected via a typed dependency which we obtain from the Stanford parser. Figure 2 shows an example of such a tree model. Thus, while the linear-chain model adds factors between labels of adjacent tokens only based on their positions in the sentence, the tree model adds factors based on the dependency parse tree to enhance the coherence of labels across tokens.

3.5 Global Models

For global models, we consider an entire input text consisting of multiple sentences $S_1, \ldots, S_n = \langle tok_1, \ldots, tok_m \rangle$, for augmenting either one of the

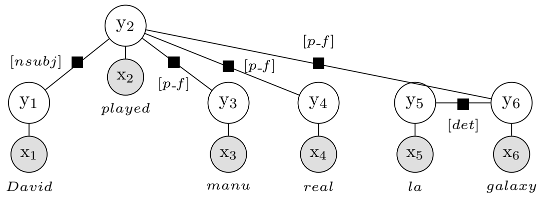

Figure 2: Tree model ($[p\_f]$ is $[prep\_for]$).

linear-chain model or *tree-shaped 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".

3.6 Inference & Learning

Our *inference objective* is to find the most probable sequence of NERD labels $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$ according to the objective function we defined in Section 3. Instead of considering the actual distribution

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \prod_A \exp \left\{ \sum_k \lambda_k \, f_{A,k}(\mathbf{x}_A, \mathbf{y}_A) \right\}$$

for this purpose, we aim to maximize an equivalent objective function as follows. Each factor A in our model couples a label variable y_t with a variable $y_{prev(t)}$: either its immediately preceding token in the same sentence, or a parsing-dependency-linked token in the same sentence, or a co-reference-linked token in a different sentence. Each of these factors has its feature functions, and we can regroup these features on a per-token basis given the log-linear nature of the objective function. This leads to the following optimization problem which has its maximum for the same label assignment as the original problem:

$$\mathbf{y}^* = \arg\max_{y_1 \ldots y_m} \exp \left(\sum_{t=1}^{m} \sum_{k=1}^{K} \lambda_k \, feature_k(y_t, y_{prev(t)}, x_1 \ldots x_m) \right)$$

where

- $prev(t)$ is the index of label y_j on which y_t depends,
- $feature_{1..K}$ are the feature functions generated from templates f_1–f_{17} of Section 4,

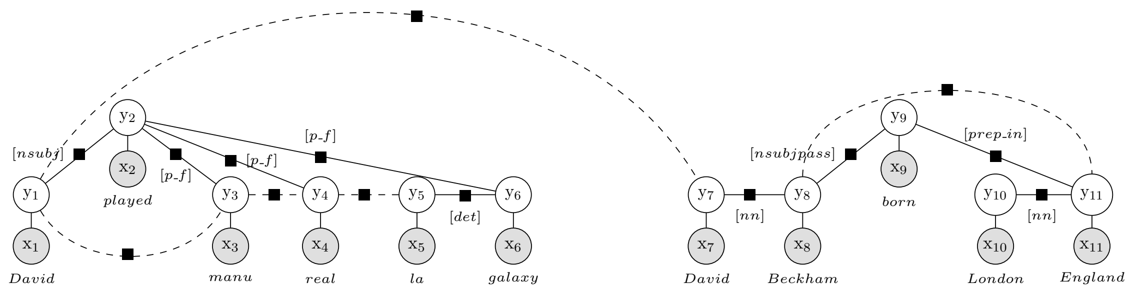

Figure 3: Global model, linking two tree models ($[p\_f]$ is short for $[prep\_for]$).

- and λ_k are the feature weights, i.e., the model parameters to be learned.

The actual number of generated features, K, depends on the training corpus and the choice of the graphical 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 in the linear-chain and tree-shaped cases. For the global models, however, exact solutions are computationally intractable. Therefore, we employ Gibbs sampling (Finkel et al., 2005) to approximate the solution.

As for the 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 model (linear-chain and tree models), and apply the same learned weights to the corresponding global models. Our implementation uses the RISO toolkit[2] for belief networks.

4 Feature Templates

We define feature templates for detecting the combined NER/NED labels of token that denote or are part of an entity mention. Once these labels are determined, the actual boundaries of the mentions, i.e., their token spans, are trivially derived by combining adjacent tokens with the same label (and disregarding all tokens with the tag `Other`).

Language Preprocessing. We employ the Stanford CoreNLP tool suite[3] for processing input documents. This includes tokenization, sentence detection, POS tagging, lemmatization, and dependency parsing. All of these provide features for our graphical model. In particular, we harness *dependency types* between noun phrases (de Marneffe et al., 2006), like ***nsubj***, ***dobj***, ***prep_in***, ***prep_for***, etc.

In the following, we introduce the complete set of feature templates f_1 through f_{17} used by our method. Templates are instantiated based on the observed input and the candidate space of possible labels for this input, and guided by distant resources like knowledge bases and dictionaries. Templates f_1, f_8–f_{13}, f_{17} generate real numbers as values derived from frequencies in training data; all other templates generate binary values denoting presence or absence of certain features. The generated feature values depend on the assignment of input tokens to variables $x_i \in \mathcal{X}$. In addition, our graphical models often consider only a specific subset of *candidate labels* as assignments to the output variables $y_i \in \mathcal{Y}$. Therefore, we formulate the feature-generation process as a set of *feature functions* that depend on both (per-factor subsets of) $\mathcal{X}$ and $\mathcal{Y}$.

Table 1 illustrates the feature generation by the set of active feature functions for the token "manu" in our running example, using three different candidate labels.

Entity Repository and Name-Entity Dictionary. Many feature templates harness a knowledge base, namely, YAGO2 (Hoffart et al., 2013), as an entity repository and as a dictionary of name-to-entity pairs (i.e., aliases and paraphrases). We import the YAGO2 *means* and *hasName* relations, a total of more than 6 Million name-entity pairs (for ca. 3 Million distinct entities). We derive additional

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

[3]`http://stanfordnlp.github.io/CoreNLP/`

Table 1: Positive features (value set to true or real number $>$ 0) for the token "manu" (x_3) with candidate labels ORG:Manchester_United_F.C. (y_3), PER:Manu_Chao (y'_3) and Other (y''_3). The domain is *Football* and the linguistic pattern is ***prep_for****[played, *]*.

Feature	y_3	y'_3	y''_3
f_1: Token-Type Prior	✓	✓	✓
f_2: Current POS	✓	✓	✓
f_3: In-Dictionary	✓	✓	
f_4: Uppercase			
f_5: Surrounding POS	✓	✓	✓
f_6: Surrounding Tokens	✓	✓	✓
f_7: Surrounding In-Dictionary	✓	✓	✓
f_8: Token-Entity Prior	✓	✓	
f_9: Token-Entity *n*-Gram Similarity	✓	✓	
f_{10}: Token-Entity Token Contexts	✓	✓	
f_{11}: Entity-Entity Token Coherence	✓	✓	
f_{12}: Entity-Domain Coherence		✓	
f_{13}: Entity-Entity Type Coherence	✓	✓	
f_{14}: Typed-Dependency	✓		✓
f_{15}: Typed-Dependency/POS	✓		✓
f_{16}: Typed-Dependency/In-Dictionary	✓		
f_{17}: Token-Entity Linguistic Contexts	✓		

NER-type-specific phrase dictionaries from supporting phrases of GATE (Cunningham et al., 2011), e.g., "Mr.", "Mrs.", "Dr.", "President", etc. for the type PER; "city", "river", "park", etc. for the type LOC; "company", "institute", "Inc.", "Ltd.", etc. for the type ORG.

Pruning the Candidate Space. To reduce the dimensionality of the generated feature space and to make the factor-graph inference tractable, we use pruning techniques based on the knowledge base and the dictionaries. To determine if a token can be a mention or part of a mention, 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 when 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. For each such candidate mention, we look up the knowledge base for entities and consider only the best n (using $n = 20$ in our experiments) highest ranked candidate entities. The ranking is based on the string similarity between the mention and the entity name, the prior popularity of the entity, and the local context similarity (using feature functions f_8, f_9, f_{10} described in Subsection 4.1).

4.1 Standard Features

For the following definitions of the feature templates, let pos_i denote the POS tag of tok_i, dic_i denote the NER tag from the dictionary lookup of tok_i, and dep_i denote the parsing dependency that connects tok_i with another token. Further, we write $sur_i = \langle tok_{i-1}, tok_i, tok_{i+1} \rangle$ to refer to the sequence of tokens surrounding tok_i. As for the possible labels, we denote by $type_i$ and ent_i an NER type and candidate entity for the current token tok_i, respectively.

Token-Type Prior. Feature $f_1(type_i, tok_i)$ captures a prior probability for tok_i being of NER type $type_i$. These probabilities are estimated from an NER-annotated training corpus. In our experiments, we used training subsets of different test corpora such as CoNLL. For example, we may thus obtain a prior of $f_1(\texttt{ORG}, \text{"}Ltd.\text{"}) = 0.8$.

Current POS. Template $f_2(type_i, tok_i)$ generates a binary feature function if token tok_i occurs in the training corpus with POS tag pos_i and NER label $type_i$. For example, $f_2(\texttt{PER}, \text{"}David\text{"}) = 1$ if the current token "David" has occurred with POS tag *NNP* and NER label PER in the training corpus. For combinations of tokens with POS tags and NER types that do not occur in the training corpus, no actual feature function is generated from the template (i.e., the value of function would be 0). For the rest of this section, we assume that all binary feature functions are generated from their feature templates in an analogous manner.

In-Dictionary. Template $f_3(type_i, tok_i)$ generates a binary feature function if the current token tok_i occurs in the name-to-entity dictionary for some entity of NER label $type_i$.

Uppercase. Template $f_4(type_i, tok_i)$ generates a binary feature function if the current token tok_i appears in upper-case form and additionally has the NER label $type_i$ in the training corpus.

Surrounding POS. Template $f_5(type_i, tok_i)$ generates a binary feature function if the current token

tok_i and the POS sequence of its surrounding tokens sur_i both appear in the training corpus, where tok_i also has the NER label $type_i$.

Surrounding Tokens. Template $f_6(type_i, tok_i)$ generates a binary feature function if the current token tok_i has NER label $type_i$, given that tok_i also appears with surrounding tokens sur_i in the training corpus. When instantiated, this template could possibly lead to a huge number of feature functions. For tractability, we thus ignore sequences that occur only once in the training corpus.

Surrounding In-Dictionary. Template $f_7(type_i, tok_i)$ performs dictionary lookups for surrounding tokens in sur_i. Similar to f_6, it generates a binary feature function if the current token tok_i and the dictionary lookups of its surrounding tokens sur_i appear in the training corpus, where tok_i also has NER label $type_i$.

Token-Entity Prior. Feature $f_8(ent_i, tok_i)$ captures a prior probability of tok_i having NED label ent_i. These probabilities are estimated from co-occurrence frequencies of name-to-entity pairs in the background corpus, thus harnessing link-anchor texts in Wikipedia. For example, we may have a prior of $f_8(\texttt{David\_Beckham}, \text{“}\textit{Beckham}\text{”}) = 0.7$, as David is more popular (today) than his wife Victoria. On the other hand, $f_8(\texttt{David\_Beckham}, \text{“}\textit{David}\text{”})$ may be lower than $f_8(\texttt{David\_Bowie}, \text{“}\textit{David}\text{”})$, for example, as this still active pop star is more frequently and prominently mentioned than the retired football player.

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

Token-Entity Token Contexts. Feature $f_{10}(ent_i, tok_i)$ measures the weighted overlap similarity between the token contexts (*tok-cxt*) of token tok_i and entity ent_j. Specifically, we use a weighted generalization of the standard overlap coefficient, *WO*, between two sets X, Y of weighted elements, $X_k \in X$ and $Y_k \in Y$:

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

We set the weights to be tf-idf scores, and hence we obtain:

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

Entity-Entity Token Coherence. Feature $f_{11}(ent_i, ent_j)$ measures the coherence between the token contexts of two entity candidates ent_i and ent_j:

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

f_{11} allows us to establish cross-dependencies among labels in our graphical model. For example, the two 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 mapped jointly.

4.2 Domain Features

We use WordNet *domains*, created by Miller (1995), Magnini and Cavagli (2000), and Bentivogli et al. (2004), to construct a taxonomy of 46 domains, including *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, introduced by Magnini and Cavagli (2000), and 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* (`PER`), *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 into domains by means of “*easy mentions*”. An easy mention is a match in the name-to-entity dictionary for which there exist at most three candidate entities (Nguyen et al., 2014). Although the mention boundaries are not explicitly provided as input, *J*-NERD still can extract these easy mentions from the entirety of all mention candidates.

In the following, let C^* be the set of candidate entities for the "easy" mentions in the input document. For each domain d (see Section 3), 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. Template $f_{12}(ent_i, tok_i)$ generates a binary feature function that captures the coherence between an entity candidate ent_i of token tok_i and the domain d which the input text is classified into. That is, $f_{12}(ent_i, tok_i) = 1$ if $d \in \text{dom}(ent_i)$. Otherwise, the feature value is 0.

Entity-Entity Type Coherence. Feature $f_{13}(ent_i, ent_j)$ computes the relatedness between the Wikipedia categories of two candidate entities $ent_i \in C_i$ and $ent_j \in C_j$, where C_i, C_j denote the two sets of candidate entities associated with tok_i, tok_j, respectively:

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

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 a few hundred thousands of Wikipedia categories integrated in the YAGO2 type hierarchy.

4.3 Linguistic Features

Recall that we harvest 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. Template $f_{14}(type_i, tok_i)$ generates a binary feature function if the background corpus contains the pattern $dep_i = deptype(arg1, arg2)$ where the current token tok_i is either $arg1$ or $arg2$, and tok_i is labeled with NER label $type_i$.

Typed-Dependency/POS. Template $f_{15}(type_i, tok_i)$ 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 the current token tok_i appears in the dependency pattern dep_i with POS tag pos_i and this combination also occurs in the training data under NER label $type_i$.

Typed-Dependency/In-Dictionary. Template $f_{16}(type_i, tok_i)$ 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 a binary feature function if the current token tok_i appears in the dependency pattern dep_i and has an entry dic_i in the name-to-entity dictionary for some entity with NER label $type_i$.

Token-Entity Linguistic Contexts. Feature $f_{17}(ent_i, tok_i)$ measures the weighted overlap between the linguistic contexts (*ling-cxt*) of token tok_i and candidate entity ent_i:

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

5 Experiments

5.1 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)[4] by removing tables where mentions in table cells do not have linguistic context; a typical example is 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 a consistent ground-

[4] `https://www.mpi-inf.mpg.de/yago-naga/yago/`

truth set, we derived the NER types from the NED ground-truth entities, 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 our 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[5], 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[6] corpus with Freebase annotations:

- **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 smaller datasets used in prior work: 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 of 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.

5.2 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[7] (Finkel et al., 2005) and the recent version 2.8.4 of the Illinois Tagger[8] (Ratinov and Roth, 2009) as baselines. These systems have NER benchmark results on CoNLL'03 that are as good as the result reported in 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) uses 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

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

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

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

[8] `http://cogcomp.cs.illinois.edu/`

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.

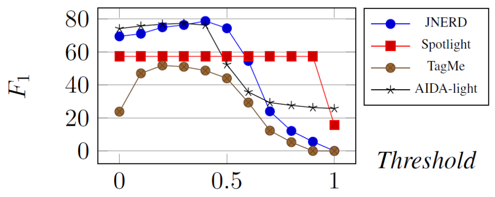

Figure 4: F_1 for varying confidence thresholds.

5.3 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 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:

$$Prec(X) = \quad |X \text{ agrees with } G|/|X|$$
$$Rec(X) = \quad |X \text{ agrees with } G|/|G|$$

where agreement means that X and G overlap in the text spans (i.e., have at least one token 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 the mention-boundary detection alone, we consider only the overlap of text spans; for evaluating NER completely, we consider both mention overlap and agreement based on the assigned NER types.

5.4 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.

5.4.1 Experiments on CRF Variants

Table 2 compares the different CRF variants. All CRFs have the same features, but differ in their factors. Therefore, some features are not effective for the linear model and the tree model. For the linear CRF, the parsing-based linguistic features and the cross-sentence features do not contribute; for the tree CRF, the cross-sentence features are not effective.

Table 2: Experiments on CoNLL-YAGO2.

Perspective	Variants	Prec	Rec	F$_1$
Mention Boundary Detection	*J*-NERD$_{\textit{linear-local}}$	94.2	89.6	91.8
	J-NERD$_{\textit{tree-local}}$	94.4	89.4	91.8
	J-NERD$_{\textit{linear-global}}$	95.1	90.3	92.6
	J-NERD$_{\textit{tree-global}}$	**95.8**	**90.6**	**93.1**
NER Typing	*J*-NERD$_{\textit{linear-local}}$	87.8	83.0	85.3
	J-NERD$_{\textit{tree-local}}$	89.5	82.2	85.6
	J-NERD$_{\textit{linear-global}}$	88.6	83.4	85.9
	J-NERD$_{\textit{tree-global}}$	**90.4**	**83.8**	**86.9**
End-to-End NERD	*J*-NERD$_{\textit{linear-local}}$	71.8	74.9	73.3
	J-NERD$_{\textit{tree-local}}$	75.1	74.5	74.7
	J-NERD$_{\textit{linear-global}}$	77.6	74.8	76.1
	J-NERD$_{\textit{tree-global}}$	**81.9**	**75.8**	**78.7**

We see that all variants perform very well on boundary detection and NER typing, with small differences only. For end-to-end NERD, however, *J*-NERD$_{\textit{tree-global}}$ 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$_{\textit{linear-global}}$. 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 focus on *J*-NERD$_{\textit{tree-global}}$ and the task of end-to-end NERD.

5.4.2 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 Section 5.2, we add a pipelined configuration of *J*-NERD , coined *P*-NERD. That is, we first run *J*-NERD in NER mode (thus 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 3: 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 3. *J*-NERD achieves the highest precision of 81.9% for end-to-end NERD, outperforming all competitors by a significant 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). These systems are more geared towards 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 obtained 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 and the Illinois Tagger 2.8.4 (Table 4). For mention boundary detection, *J*-NERD achieved an F_1 score of 93.1% versus 93.4% by Stanford NER, 93.3% by Illinois Tagger, 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, 85.3% by Illinois Tagger, 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.

Table 4: Experiments on NER against state-of-the-art NER systems.

Perspective	Variants	Prec	Rec	F$_1$
Mention Boundary Detection	*P*-NERD	95.6	90.5	92.9
	J-NERD	**95.8**	90.6	93.1
	Stanford NER	95.6	**91.3**	**93.4**
	Illinois Tagger	95.5	91.2	93.3
NER Typing	*P*-NERD	89.6	83.4	86.3
	J-NERD	**90.4**	83.8	**86.9**
	Stanford NER	89.3	**84.5**	86.8
	Illinois Tagger	87.5	83.2	85.3

5.4.3 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$_{\textit{tree-global}}$ model.

Table 5: 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 5 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.

5.5 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 5.1). All other systems, including the best *J*-NERD method, were run on the corpus under the same conditions.

Table 6: 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 6). 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 more of the 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.

5.6 End-to-End NERD on ClueWeb

The results for ClueWeb are shown in Table 7. 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 7: 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

6 Conclusions

We have shown that coupling the tasks of NER and NED in a joint CRF-like 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-shaped models capture the structure of dependency parse trees, and we couple multiple such tree models across sentences. Second, we harness non-standard features about domains and novel features based on 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 more use cases for joint NERD, especially for content analytics over news streams and social media.

7 Acknowledgments

We would like to thank Greg Durrett for helpful discussions about entity disambiguation on ACE. We also thank the anonymous reviewers, and our action editors Lillian Lee and Hwee Tou Ng for their very thoughtful and helpful comments.

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.

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, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 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, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. 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. *Artificial Intelligence*, 194:28–61.

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*, 45(3):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 *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. *Communications of the ACM*, 38(11):39–41.

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*, 194:222–239.

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 *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*, 4(4):267–373.

Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. 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.