Structural Reranking Models for Named Entity Recognition

Truc-Vien T. NGUYEN a and Alessandro MOSCHITTI b

a CIMeC - University of Trento
Corso Bettini 31, 38068 Rovereto (TN), Italy
b DISI - University of Trento
Via Sommarive 5, 38123 Povo (TN), Italy

Abstract. We present a method for incorporating global features in named entity recognizers using reranking techniques and the combination of two state-of-the-art NER learning algorithms: conditional random fields (CRFs) and support vector machines (SVMs). The reranker employs two kinds of features: flat and structured features. The former are generated by a polynomial kernel encoding entity features whereas tree kernels are used to model dependencies amongst tagged candidate examples. The experiments on two standard corpora in two languages, i.e. the Italian EVALITA 2009 and the English CoNLL 2003 datasets, show a large improvement on CRFs in F-measure, i.e., from 80.34% to 84.33% and from 84.86% to 87.99%, respectively. Our analysis reveals that (i) both kernels provide a comparable improvement over the CRFs baseline; and (ii) their combination improves CRFs much more than the sum of the individual contributions, suggesting an interesting synergy.

Keywords: named entity recognition, reranking, kernel methods, conditional random fields

1. Introduction

Research in statistical natural language processing has shown that reranking is a promising approach for enhancing system accuracy. This method first employs a probabilistic model to generate a list of top-N candidates and then reranks such list with additional features (see for example, (Moschitti et al., 2006; Moschitti et al., 2008; Dinarelli et al., 2009; Nguyen et al., 2010; Johansson and Moschitti, 2010; Moschitti et al., 2012; Severyn and Moschitti, 2012). The approach is appealing as it can flexibly incorporate arbitrary features into a learning model. These features help in discriminating good from bad hypotheses and consequently can be used to select the best one. Various algorithms have been applied for reranking in NLP applications, including parsing (Collins, 2000; Collins and Duffy, 2002; Charniak and Johnson, 2005; Huang, 2008), name tagging (Collins, 2002; Collins and Duffy, 2002), machine translation (Shen et al., 2004) and opinion detection (Johansson and Moschitti, 2010). Such work has exploited the discriminative property as the main criterion of the reranking algorithm.

Recent research (Moschitti et al., 2012; Severyn and Moschitti, 2012) has shown that reranking models improve when coupled with kernel methods (Collins and Duffy, 2001; Moschitti, 2004; Moschitti, 2006). The main contribution of the latter is the possibility to extract a huge amount of features along with their dependencies from the ranking hypotheses. Indeed, while feature-based learning algorithms only involve the dot product between feature vectors, kernel methods allow for a higher generalization by replacing the dot product with a function between pairs of linguistic objects. Such functions are a kind of similarity measure satisfying certain properties. For example, tree kernels (Collins and Duffy, 2001) encode grammatical information in learning algorithms by computing the number sub-trees shared by two syntactic trees (associated with the classification instances). Similarly, sequence kernels (Lodhi et al., 2002) count the number of common sub-sequences shared by two input strings.
The use of reranking kernels for NER is promising as previous algorithms, i.e., described in (Collins, 2002; Collins and Duffy, 2002), only targeted the entity detection (and not entity classification) task. Besides, kernel methods offer a natural way to exploit linguistic properties. Finally, the NER hypotheses can be generated from an entire sentence such that it can be possible to capture important global semantic features.

In this paper, we describe the use of kernel methods for performing effective reranking of NER hypotheses provided by basic systems for both Italian and English languages over well-known corpora. The key aspect of our method regards the use of structured and flat features for discriminating between candidate hypotheses. For this purpose, we apply tree kernels to a tree structure encoding sentence annotation, which, in turn, refers to a tagged sequences. We also combine such model with a polynomial kernel applied to vectors of innovative global features.

Our main contribution is to show that tree kernels can be used to define general and effective features modeling syntactic but also semantic information for NER reranking. Our study demonstrates that the composite kernel is very effective for reranking named-entity sequences. Without the need of producing and heuristically combining learning models like previous work on NER, our composite kernel not only captures most of the flat features but also efficiently exploits structured features. More interestingly, this kernel yields significant improvement when applied to two corpora of two different languages. The evaluation on the Italian corpus shows that our approach outperforms the best-reported methods whereas on the English data it reaches the state-of-the-art.

In the remainder of this paper, Section 2 reports on the related work, Section 3 introduces the tree kernel theory, Section 4 describes the datasets and the baseline performance associated with them, Section 5 describes our reranking models, Section 6 illustrates out experiments and finally Section 7 derives the conclusions.

2. Related Work

Named-entities (NEs) are objects that can be referred by names (Chinchor and Robinson, 1998), such as people, organizations, and locations. NEs are essential for defining the semantics of a document. The research on NER has been promoted by the Message Understanding Conferences (MUCs, 1987-1998), the shared task of the Conference on Natural Language Learning (CoNLL, 2002-2003), and the Automatic Content Extraction program (ACE, 2002-2005). Figure 1 shows a text from the CoNLL 2003 corpus, where all named entities are in bold.

The MUC and ACE programs were held in order to exchange research findings on techniques for extracting a variety of levels of information from official data. However, most early works were based on hand-crafted rules. The use of machine learning for NER was firstly proposed in (Bikel et al., 1997; Leek, 1997) and consisted in a simple Hidden Markov Model (HMM). Since then, various IE learning algorithms have been developed. In the last decade, most effort has been focused on developing machine learning algorithms to reduce effort and time required by domain experts for engineering rules.

Existing approaches for NER using machine learning fall into two types. Simpler NER systems employing one learning algorithm only. These include, e.g., Maximum Entropy (Bender et al., 2003; Chieu and Ng, 2003; Curran and Clark, 2003), Hidden Markov Models (Zhou and Su, 2002), Perceptron (Carreras et al., 2003a), Adaboost (Carreras et al., 2003b), Conditional Random Fields (McCallum and Li, 2003) and Support Vector Machines (Mayfield et al., 2003).

The second type uses multiple learning algorithms used in classifier committees. For example, (Florian et al., 2003) present a classifier combination for NER in which four diverse models are combined under different conditions.

2.1. Reranking in Statistical Natural Language Processing

As mentioned in the introduction, discriminative rerankers have been used for many NLP applications, including parsing (Collins, 2000; Collins and Duffy, 2002; Charniak and Johnson, 2005; Huang, 2008), name tagging (Collins, 2002; Collins and Duffy, 2002), machine translation (Shen et al., 2004) and opinion detection (Johansson and Moschitti, 2010; Johansson and Moschitti, 2013). Their algorithm is very simple: first a basic classifier is used to generate a set of automatic annotation hypotheses of limited size, and then a second classifier, i.e., the reranker, is used to select the best hypothesis. The reranker is a simple binary classifier of hypothesis pairs but can encode complex dependencies, e.g., thanks to structural kernels (e.g., see (Moschitti, 2008)). In particular, tree, sequence and linear kernels can be applied to struc-
tural and feature-vector representations of the hypothesis pairs to train the reranker. For example, we have used them for reranking the output of (i) a multi-label hierarchical text classifiers (Moschitti et al., 2012); (ii) a question answering system (Severyn and Moschitti, 2012); (iii) a concept labeling and segmentation model for speech (Dinarelli et al., 2009); and (iv) a semantic role labeling system (Moschitti et al., 2006; Moschitti et al., 2008).

A typical approach to discriminative reranking using kernel methods is the preference reranker. The task concerns the selection of the correct hypothesis candidate $H_i$ from a candidate set $\mathcal{H} = \{H_1, H_2, \ldots, H_k\}$. This task is reduced to a binary classification problem by training a model on hypothesis pairs. The member order of the latter determines the positive or negative label of the data. For example, if $H_1$ is better than $H_2$, we can choose to have the following order $\langle H_1, H_2 \rangle$, consequently, $\langle H_2, H_1 \rangle$ will be a negative example. This training set can then be used to train a binary classifier. At classification time, pairs are not formed (since the correct candidate is not known); instead, the standard one-versus-all binarization method is applied.

The kernels are then engineered to implicitly represent the differences between the objects in the pairs. If we have a valid kernel $K$ over $\mathcal{H}$, we can construct a preference kernel $P_K : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{R}$ as follows:

$$P_K(\langle H_1, H_2 \rangle, \langle H_3, H_4 \rangle) = K(H_1, H_3) + K(H_2, H_4) - K(H_1, H_4) - K(H_2, H_3) \quad (1)$$

It is easy to show that $P_K$ is also a valid Mercer’s kernel (Shawe-Taylor and Cristianini, 2004). This makes it possible to use kernel methods to train the reranker. The next section describes tree kernels in detail to demonstrate how they can generate effective features to represent trees. The latter are used to represent NER hypotheses in our reranker.

3. Tree Kernels

Tree kernels represent trees in terms of their substructures (called tree fragments). Such fragments form a feature space, which, in turn, is mapped into a vector space. Tree kernels measure the similarity between pair of trees by counting the number of fragments in common. There are three important characterizations of fragment type: the SubTrees (STs), the Syntactic Trees (SYTs) and the Partial Trees (PTs).

3.1. Tree Fragment Types

An ST is defined by taking any node along with its descendants. An SYT is a more general structure, which does not necessarily include all the descendants. One strong constraint is that an SYT (and also an ST) must be generated by applying the same grammatical rule set that generated the original tree, as pointed out in (Collins and Duffy, 2001). The rules impose grammatical constraints on the children of a node, which cannot be separated (otherwise the grammar rule generating an SYT with split children may be constituted by a new rule and thus would be different from the initial set of rules). A Partial Tree (PT) is a more general form of substructure obtained by relaxing constraints over the SYTs, i.e., children can be separated to generate PTs. Figure 2 shows the overall fragment set of the ST, SYT and PT kernels for the syntactic parse tree of the sentence fragment: *gives a talk*. In general, the tree kernel impact depends on the specific application, which should suggest the appropriate kernel type.

To carry out efficient computation of this high dimensional representation, we follow the algorithm introduced in (Collins and Duffy, 2001). We enumerate all tree fragments that occur in the training data from 1 to $n$. Thus, each tree is represented by an $n$-dimensional vector where the $i$’th component counts the number of occurrences of the $i$’th tree fragment. Let us define the function $c_i(T)$ to be the number of
occurrences of the \( i \)'th tree fragment in tree \( T \), so that \( T \) is now represented as \( \tilde{c}(T) = (c_1(T), c_2(T), \ldots, c_n(T)) \).

We then examine the inner product between two trees \( T_1 \) and \( T_2 \) under this representation, i.e., \( K(T_1, T_2) = \tilde{c}(T_1) \cdot \tilde{c}(T_2) \). To compute \( K \), we define (i) the set of nodes in trees \( T_1 \) and \( T_2 \) as \( N_1 \) and \( N_2 \) respectively; and (ii) the indicator function \( I_i(n) \) to be 1 if subtree is seen rooted at node \( n \) and 0 otherwise. It follows that \( c_1(T_1) = \sum_{n \in N_1} I_i(n) \) and \( c_1(T_2) = \sum_{n \in N_2} I_i(n) \). With some simple algebra we have:

\[
\tilde{c}(T_1) \cdot \tilde{c}(T_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_{i} I_i(n_1) I_i(n_2) \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta(n_1, n_2)
\]

where we define \( \Delta(n_1, n_2) = \sum_i I_i(n_1) I_i(n_2) \). Note that \( \Delta(n_1, n_2) \) can be computed in polynomial time and determines the kernel type.

3.2. Syntactic Tree Kernel

To compute STK, we need to define \( \Delta(n_1, n_2) \) as follows:

- If the productions at \( n_1 \) and \( n_2 \) are different \( \Delta(n_1, n_2) = 0 \).

- If the productions at \( n_1 \) and \( n_2 \) are the same, and \( n_1 \) and \( n_2 \) are pre-terminals, then \( \Delta(n_1, n_2) = 1 \).

- Else if the productions at \( n_1 \) and \( n_2 \) are the same and \( n_1 \) and \( n_2 \) are not pre-terminals,

\[
\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j))), \quad (2)
\]

where \( nc(n_1) \) is the number of children of \( n_1 \) in the tree \( (nc(n_1) = nc(n_2) \) since the productions at \( n_1 / n_2 \) are the same) and \( ch(n_1, j) \) is the \( j \)'th child of \( n_1 \).

Note that \( \Delta(n_1, n_2) \) counts the number of common subtrees rooted at both \( n_1 \) and \( n_2 \). From the identity \( \tilde{c}(T_1) \cdot \tilde{c}(T_2) = \sum_{n_1, n_2} \Delta(n_1, n_2) \), and the recursive definition of \( \Delta(n_1, n_2) \), it follows that STK can be calculated in \( O(|N_1||N_2|) \) time.

3.3. Partial Tree Kernel (PTK)

The computation of PTFs is carried out by the \( \Delta \) function defined as follows:

1. if the node labels of \( n_1 \) and \( n_2 \) are different then \( \Delta(n_1, n_2) = 0 \).
2. else:

\[
\Delta(n_1, n_2) = 1 + \sum_{\tilde{I}_j} \prod_{j=1}^{l(\tilde{I}_1)} \Delta(ch(n_1, \tilde{I}_j), ch(n_2, \tilde{I}_j))
\]
where $\bar{t}_1 = \langle k_1, k_2, k_3, \ldots \rangle$ and $\bar{t}_2 = \langle k'_1, k'_2, k'_3, \ldots \rangle$ are index sequences associated with the ordered child sequences $ch(n_1, \cdot)$ of $n_1$ and $ch(n_2, \cdot)$ of $n_2$, respectively. $\bar{t}_{j1}$ and $\bar{t}_{j2}$ point to the $j$-th child in the corresponding sequence, and $l(\cdot)$ returns the sequence length, i.e., the number of children in the sequence.

Furthermore, we add two decay factors: $\mu$ for the depth of the tree and $\lambda$ for the length of the child subsequences with respect to the original sequence, which accounts for gaps. Hence, the PTK expression for $\Delta(n_1, n_2) = $

$$
\mu \left( \lambda^2 + \sum_{\bar{t}_{j1}, \bar{t}_{j2}}^{\bar{t}_{11}, \bar{t}_{21}} (\lambda^{d(\bar{t}_{j1})} + d(\bar{t}_{j2}) \prod_{j=1}^{l(\bar{t}_{j1})} \Delta(ch(n_1, \bar{t}_{j1}), ch(n_2, \bar{t}_{j2}))) \right) (3)
$$

where $d(\bar{t}_{j1}) = \bar{t}_{j1+1} - \bar{t}_{j1} + 1$ and $d(\bar{t}_{j2}) = \bar{t}_{j2+1} - \bar{t}_{j2} + 1$. This way, we penalize both larger trees and child subsequences with gaps. Eq. 3 is more general than Eq. 2. Indeed, if we only consider shared subsequences that contain all the node children (no child of a node is skipped), we actually obtain the STK kernel. The computational complexity of PTK is $O(pp^2 |n_{T1}| \times |n_{T2}|)$ (Moschitti, 2006), where $p$ is the largest subsequent of children that we want to consider and $\rho$ is the maximal outdegree observed in the two trees. However, as shown in (Moschitti, 2006), the average running time again tends to be linear in the number of nodes for natural language syntactic trees.

3.4. The Polynomial Kernel

The polynomial kernel between two candidate tagged sequences is defined as:

$$
K(x, y) = (1 + \bar{x}_1 \cdot \bar{x}_2)^2,
$$

where $\bar{x}_1$ and $\bar{x}_2$ are two feature vectors extracted from the two sequences with the global feature template.

3.5. Kernel Engineering

Kernel engineering can be carried out by combining basic kernels with additive or multiplicative operators or by designing specific data objects (vectors, sequences and tree structures) for the target tasks.

It is worth noting that well-known kernels applied to new structures produce completely new kernels as shown hereafter. Let $K(t_1, t_2) = \phi(t_1) \cdot \phi(t_2)$ be a basic kernel, where $t_1$ and $t_2$ are two trees. If we map $t_1$ and $t_2$ into two new structures $s_1$ and $s_2$ with a mapping $\phi_M(\cdot)$, we obtain: $K(s_1, s_2) = \phi(s_1) \cdot \phi(s_2) = \phi(\phi_M(t_1)) \cdot \phi(\phi_M(t_2)) = \phi'(t_1) \cdot \phi'(t_2) = K'(t_1, t_2)$, which is a noticeably different kernel induced by the mapping $\phi' = \phi \circ \phi_M$.

3.6. The role of tree kernels in Reranking

Eq. 1 allow us to use any kernel $K$ for defining the preference reranker. Therefore, we can use tree kernels, such as PTK or STK, to describe the similarity between two hypotheses. In Section 5.2, we will describe a tree representation for hypotheses, which can be used as input of tree kernels. The main idea is that tree kernels can measure the similarity between such hypotheses in terms of their subtrees. Such similarity can learn the order of hypotheses according to their accuracy. For example, Eq. 1 imposes that if $H_3$ is similar to $H_2$ and $H_2$ is similar to $H_4$, then $H_3$ is more accurate than $H_2$ will suggest that $H_3$ is more accurate than $H_4$.

In the next section we present the first step of reranking, i.e., the basic NER used for building the hypotheses along with the baseline results.

4. Datasets and Baseline for Named Entity Recognition

A robust NER system is expected to be adaptable to different domains and languages. Therefore, we experimented with two datasets: i) the well-known CoNLL 2003 English shared task corpus; and ii) the EVALITA 2009 Italian corpus. Statistics about such datasets are shown in tables 1 and 2. In the following sections we describe the datasets and the typical accuracy that baseline NERs attain on them.

4.1. Datasets

The CoNLL English dataset. The CoNLL 2003 English dataset is created within the shared task of CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003). It is a collection of news wire articles from the Reuters Corpus, annotated with four entity types: Person (PER), Location (LOC), Organization (ORG) and Miscellaneous name (MISC). The training and the development datasets are news feeds from August 1996, while the test set contains news feeds from December 1996. Accordingly, the named entities in the test dataset are considerably different from those that appear in the training or the development set.
The EVALITA Italian dataset. The EVALITA 2009 Italian dataset is based on I-CAB, the Italian Content Annotation Bank (Magnini et al., 2006), annotated with four entity types: Person (PER), Organization (ORG), Geo-Political Entity (GPE) and Location (LOC). The training data, taken from the local newspaper “L’Adige”, consists of 525 news stories, which belong to five categories: News Stories, Cultural News, Economic News, Sports News and Local News. Test data, on the other hand, consists of completely new texts, taken from the same newspaper and consists of 180 news stories.

4.2. The baseline algorithm

We selected Conditional Random Fields (CRFs) (Lafferty et al., 2001) as the baseline model. These are a probabilistic framework for labeling and segmenting sequence data. They present several advantages over other purely generative models such as Hidden Markov models (HMMs) by relaxing the independence assumptions required by HMMs. Besides, HMMs and other discriminative Markov models are prone to the label bias problem, which is effectively solved by CRFs.

The named-entity recognition (NER) task is framed as assigning label sequences to a set of observation sequences. We follow the IOB notation where the NE tags have the format B-TYPE, I-TYPE or O, which mean that the word is a beginning, a continuation, or not part of an entity at all. For example, the following sentence is labeled with the tags above:

Il/O presidente/O della/O Fifa/B-ORG Sepp/B-PER Blatter/I-PER affermando/O che/O il/O torneo/O era/O stato/O ottimo/O (FIFA president Sepp Blatter says that the tournament was excellent)

For our experiments, we used CRF++\(^1\) to build our recognizer, which is a model trained discriminatively with the unigram and bigram features. These are extracted from a window of \(k\) words centered in the target word \(w\) (i.e. the one we want to classify with the B, O, I tags). The features used for each token are described below.

Basic features. The basic features are small, primitive units, consisting of the word itself with some features directly derived from it:

1. \(w_i\) for \(i = 1, \ldots, n\) is the \(i\)’th word
2. \(l_i\) is the word in lower-case
3. \(p_{1i}, p_{2i}, p_{3i}, p_{4i}\) are four prefixes of \(w_i\) of size 1, 2, 3 or 4.
4. \(s_{1i}, s_{2i}, s_{3i}, s_{4i}\) are four suffixes of \(w_i\) as before.
5. \(f_i\) is the part-of-speech of \(w_i\)
6. \(g_i\) is the orthographic feature that tests whether a word contains all upper case, initial upper case, all lower-case letters.

\(^1\)http://crfpp.sourceforge.net
Table 3
CRFs results on the CoNLL test set

<table>
<thead>
<tr>
<th>Category</th>
<th>Pr</th>
<th>Re</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>85.37</td>
<td>84.35</td>
<td>84.86</td>
</tr>
<tr>
<td>LOC</td>
<td>90.25</td>
<td>88.61</td>
<td>89.42</td>
</tr>
<tr>
<td>MISC</td>
<td>79.81</td>
<td>74.51</td>
<td>77.07</td>
</tr>
<tr>
<td>ORG</td>
<td>80.02</td>
<td>77.85</td>
<td>78.92</td>
</tr>
<tr>
<td>PER</td>
<td>87.94</td>
<td>90.92</td>
<td>89.41</td>
</tr>
</tbody>
</table>

Table 4
SVMs results on the CoNLL test set

<table>
<thead>
<tr>
<th>Category</th>
<th>Pr</th>
<th>Re</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>84.76</td>
<td>84.18</td>
<td>84.47</td>
</tr>
<tr>
<td>LOC</td>
<td>87.99</td>
<td>88.6</td>
<td>88.29</td>
</tr>
<tr>
<td>MISC</td>
<td>79.22</td>
<td>75.76</td>
<td>77.45</td>
</tr>
<tr>
<td>ORG</td>
<td>80.96</td>
<td>76.81</td>
<td>78.83</td>
</tr>
<tr>
<td>PER</td>
<td>87.3</td>
<td>90.85</td>
<td>89.04</td>
</tr>
</tbody>
</table>

7. $k_i$ is a feature that tests whether a token is a word, a number, a symbol, or a punctuation mark.

8. $o_i$ is the gazetteer feature. We simply look up $w_i$ in our knowledge base.

**Combined features.** A bi-gram feature is derived from the combination of two consecutive units. To allow the highest generalization, the consecutive units can be lexical words or any of their basic features as described in the previous section. For example, the text “presidential palace” can have one feature as “presidential/palace” and another feature as “JJ/NN” where JJ and NN are the part-of-speech of presidential and palace.

1. The combinations $w_{i-2}/w_{i-1}$, $w_{i-1}/w_i$, and $w_i/w_{i+1}$
2. Similar to the first, but use $l_i$ instead of $w_i$
3. Similar to the first, but use $p1_i$, $p2_i$, $p3_i$, $p4_i$ instead of $w_i$
4. Similar to the first, but use $s1_i$, $s2_i$, $s3_i$, $s4_i$ instead of $w_i$
5. Similar to the first, but use $f_i$ instead of $w_i$
6. Similar to the first, but use $o_i$ instead of $w_i$
7. The combinations $w_{i-2}/f_{i-2}$, $w_{i-1}/f_{i-1}$, $w_i/f_i$, and $w_{i+1}/f_{i+1}$
8. The combinations $f_{i-2}/o_{i-2}$, $f_{i-1}/o_{i-1}$, $f_i/o_i$, and $f_{i+1}/o_{i+1}$
9. The combinations $w_{i-2}/o_{i-2}$, $w_{i-1}/o_{i-1}$, $w_i/o_i$, and $w_{i+1}/o_{i+1}$

The gazetteer lists are built with names imported from different sources. For English, the geographic features are imported from NIMA’s GEnet Names Server (GNS) and The Alexandria Digital Library (ADL) gazetteer. The company data is included with all the publicly traded companies listed in Google directory and the European business directory. For Italian, the generic proper nouns are extracted from Wikipedia and various Italian sites. Moreover, the gazetteer lists for Italian are extracted from La Repubblica (Baroni et al., 2004), a large corpus of Italian newspaper text by using rule-based approach with patterns tuned specifically for each NE class.

4.3. Baseline Results

We trained the NER classifier on the two datasets. The Italian system participated in the EVALITA 2009 NER task (Nguyen et al., 2009). In addition to the base CRF classifier we trained another classifier with SVMs. Although this performed worse than the base model, we reported its results for completeness.

Table 5 and 6 show the final performance on the Italian test set with CRFs and SVMs. We found that,
with the same set of features, the accuracy of the NE classifiers trained with two models is rather competitive. Moreover, the NE classes, GPE and PER, achieve a rather good F1, while the recognition of ORG and LOC seems more problematic. This is in line with previous work according to which ORG seems to be the most difficult category to be learned. The lack of resources (the gazetteer for LOC is the smallest) may be another important reason for such low accuracy.

5. Reranking Models for Named Entity Recognition

In this section we show our most important contribution, i.e., the definition of a reranker for the outcome of a NER applied to an entire sentence. One important characteristic of such model is the use of structural kernels (i.e., tree kernels) to learn the reranking classifier. We also combine them with a polynomial kernel applied to traditional and innovative NER features.

5.1. Reranking Strategy

We first train a CRFs model to generate 10-best candidates per sentence, along with their probabilities. Each candidate is then represented by a semantic tree together with a feature vector. We set our reranking task as a binary classification problem where examples are pairs of hypotheses \( < H_i, H_j > \). As an example, let us consider the sentence:

“South African Breweries Ltd bought stakes in the Lech and Tychy brewers”

tagged by CRFs in three different ways:

\[
H_1 : \text{B-ORG I-ORG I-ORG I-ORG O O O B-ORG O B-ORG O}
\]

\[
H_2 : \text{B-MISC I-MISC B-ORG I-ORG O O O B-ORG I-ORG I-ORG O}
\]

\[
H_3 : \text{B-ORG I-ORG I-ORG O O O B-ORG O B-LOC O}
\]

where the first candidate is the correct sequence whereas B-ORG, I-ORG, B-LOC, O are the NE tags generated according to the IOB notation (as described in Section 4.2). Using such tagged sequences, we build the following pairs of hypotheses:

\[+1 < H_i, H_j >, +1 < H_i, H_k >, -1 < H_j, H_i >, \text{ and } -1 < H_k, H_i >,\]

where we assign the positive label to the first two pairs and negative one to the others as we assume that positive instances must have the correct hypothesis \( H_i \), in the first position. Then a binary classifier based on SVMs and kernel methods can be trained on such pairs to discriminate between the best hypothesis, i.e. \( H_i \), and the others. At testing time the hypothesis receiving the highest score is selected (Collins and Duffy, 2001).
5.2. Representation of Tagged Sequences in Semantic Trees

We provide a representation for hypotheses based on structures and global features. As in the case of NER, an input candidate is a sequence of word/tag pairs $s = \{w_1/t_1 \ldots w_n/t_n\}$, where $w_i$ is the $i$'th word and $t_i$ is the $i$'th NE tag for that word. The first representation we consider is a tree structure. For example, Figure 3 shows the semantic tree for $H_i$. In such tree representation: (a) the words are the leaf nodes; (b) their BIO tags are the father nodes and (c) the NER category is the father of BIO tags. A final fake root node connects all the NEs of the sentence. If a word is not an NE, it is simply connected to the null node.

This structure allows for best exploiting tree kernels between competing candidates. Indeed, in such kernel space, the inner product counts the number of common subtrees, i.e., the kernel computed on pairs of sentences containing similar sequences of NE tags are likely to have higher score. For example, the similarity between $H_i$ and $H_k$ will be higher than the similarity between $H_j$ and $H_k$. This is reasonable since $H_i$ and $H_k$ have the highest $F_1$.

It is worth noting that another useful modification is the flexibility of incorporating diverse and arbitrary features into this tree structure by adding them as children of the entity tag nodes. These features can be exploited efficiently with PTK, which relaxes the constraints of production rules.

5.3. Global features

In addition to the structural representation, we use traditional feature vectors. Their advantage is that they can specify some global and unstructured information about the NEs of the entire sentence. In particular, we use novel features described hereafter.

Mixed n-grams features. In previous work, some global features have been used, e.g., (Collins, 2002; Collins and Duffy, 2002) but the employed algorithm just exploited arbitrary information regarding word types and linguistic patterns. In contrast, we define and study diverse features by also considering the n-grams patterns preceding and following the target entity.

Complementary context. In supervised learning, NER systems often suffer from low recall, which is caused by lack of both resource and context. For example, a word like “Arkansas” may not appear in the training set and, in the test set, there may not be enough context to infer its NE tag. In such cases, neither global features defined in (Chieu and Ng, 2002) nor aggregated contexts (Chieu and Ng, 2003) can help.

To overcome this deficiency, we employed the following unsupervised procedure: first, the baseline NER is applied to a target unlabeled corpus. Second, we associate each word of the corpus with the most frequent NE category assigned in the previous step. Finally, the above tags are used as features during the training of the improved NER and also for building the feature representation for a new classification instance. This way, for any unknown (i.e., never seen in the training set) word $w$ of the test set, we can rely on...
the most probable NE category as feature. That is, we use the most frequent category assigned to \( w \) by a basic NER over a reference corpus. The advantage is that we derived it by using the average over many possible contexts of \( w \), which are in the different instances of the unlabeled corpus.

The unlabeled corpus for Italian was collected from La Repubblica \(^6\) and it contains over 20 millions words, whereas the one for English was collected mainly from The New York Times \(^7\) and BBC news stories \(^8\) with more than 35 millions words.

**Head word.** As the head word of an entity plays an important role in information extraction (Surdeanu et al., 2003; Bunescu and Mooney, 2005), it is included in the global set together with its orthographic representation. More in detail, our global features are the following:

1. \( w_i \) for \( i = 1 \ldots n \) is the \( i \)th word
2. \( t_i \) is the NE tag of \( w_i \)
3. \( g_i \) is the gazetteer feature of the word \( w_i \)
4. \( f_i \) is the most frequent NE tag seen in a large corpus of \( w_i \)
5. \( h_i \) is the head word of the entity. We normally set the head word of an entity as its last word. However, when a preposition exists in the entity string, its head word is set as the last word before the preposition. For example, the head word of the entity “University of Pennsylvania” is “University”.

6. Mixed \( n \)-grams features of the words and their gazetteer/frequent tag before/after the start/end of an entity. In addition to the normal \( n \)-grams solely based on words, we mix words with gazetteer/frequent tag seen from a large corpus and create mixed \( n \)-grams features.

Table 7 shows the full set of global features in our reranking framework. These features are anchored for each entity instance and adapted to entity categories. For example, the entity string (first feature) of the entity “United Nations” with entity type “ORG” is “ORG United Nations”. This helps to discriminate different entities with the same surface forms. Moreover, they can be combined with \( n \)-gram patterns to learn and explicitly push the score of the correct sequence above the score of competing sequences.

**An example of feature extraction.** Given the sentence “Eric Furda, dean of admissions at the University of Pennsylvania, is taking reader questions.” with the target entity “University of Pennsylvania”, Table 8 describes each feature with an example in context using the following primitives:

- \( w_1w_2w_3 \ldots w_e = \) University of Pennsylvania (the entity string)
- \( g_{e-1} = \) det (the gazetteer feature of the word the is a determiner)
- \( g_e = \) ORG PREFIX (the gazetteer feature of the word University is an organization prefix)
- \( g_{e+1} = \) prep (the gazetteer feature of the word of is a preposition)
- \( f_{e-1} = O \) (the most frequent NE tag of the word the is “O”, i.e., the null tag)
- \( f_e = \) ORG (the most frequent NE tag of the word University is an organization)

---

\(^6\)http://www.repubblica.it/
\(^7\)http://www.nytimes.com/
\(^8\)http://news.bbc.co.uk/
We point out that we used the ontology and knowledge base from the KIM ontology (Kiryakov et al., 2004) as our gazetteer. The KIM ontology contains about 300 classes, 100 attributes and relations. The KIM knowledge base (KB) contains about 7,500 entities with more than 110,000 aliases. Figure 4 shows an excerpt from the KIM ontology.

Table 7: Global features with the polynomial kernel for reranking

- $f_2 = 0$ (the most frequent NE tag of the word)
- $f_3 = 1$ (if the text is a noun, matches with LOCATION)
- $f_4 = 1$ (if the text is a noun, matches with PERSON and CITY, which lie in two different branches in the ontology)

Mixed trigrams of the words/frequent-tag features before the start of the entity

Mixed trigrams of the words/frequent-tag features after the end of the entity

Mixed trigrams of the word/gazetteer features before the start of the entity

Mixed trigrams of the word/gazetteer features after the end of the entity

Features before the start of the entity

Features after the end of the entity

Merged trigrams of the words/frequent-tag features

Merged trigrams of the gazetteer feature

Similar to $f_2$ but using the frequent-tag instead of the gazetteer feature

Similar to $f_2$ but using the frequent-tag instead of the gazetteer feature

The head word of the entity (see Section 3.5)

The most frequent NE tag feature (start of the word)

The most frequent NE tag feature (end of the word)

The most frequent NE tag feature (end of the word)

The most frequent NE tag feature (end of the word)

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased

Indicates whether the head word is lowercased
the class ENTITY as the parent class for both PERSON and CITY.

5.4. Reranking with Composite Kernel

In this section we describe our novel tagging kernels based on diverse global features as well as semantic trees for reranking candidate tagged sequences. As mentioned in the previous section, we can engineer kernels by combining trees and entity kernels.

5.5. The Tagging Kernels

In our reranking framework, we incorporate the probability from the original model with the tree structure as well as the feature vectors. Let us consider the following notations:

- \( K(x, y) = \text{L}(x) \cdot \text{L}(y) \) is the basic kernel where \( \text{L}(x) \) is the log probability of a candidate tagged sequence \( x \) under the BIO-tag probability model.
- \( TK(x, y) = t(x) \cdot t(y) \) is the partial tree kernel applied to our semantic tree.
- \( FK(x, y) = f(x) \cdot f(y) \) is the polynomial kernel applied to the global features.

The tagging kernels between two tagged sequences are defined as follows:

1. \( CTK = \alpha \cdot K + (1 - \alpha) \cdot TK \)
2. \( CFK = \beta \cdot K + (1 - \beta) \cdot FK \)
3. \( CTFK = \gamma \cdot K + (1 - \gamma) \cdot (TK + FK) \),

where \( \alpha, \beta, \gamma \) are parameters weighting the two participating terms. Experiments on the validation set showed that these combinations yield the best performance with \( \alpha = 0.2 \) for both languages, \( \beta = 0.4 \) for English and \( \beta = 0.3 \) for Italian, \( \gamma = 0.24 \) for English and \( \gamma = 0.2 \) for Italian.

6. Experiments

In these experiments, we compare the baseline performance with the different reranking models presented in the previous section.

6.1. Experimental setup

As a baseline we trained the CRF classifier on the entire training set (11,227 sentences in the Italian and 14,987 sentences in the English corpus). In developing a reranking strategy for both English and Italian, the training data was split into 5 sections, and in each case the baseline classifier was trained on 4/5 of the data, then used to decode the remaining 1/5.

The top 10 hypotheses together with their log probabilities were computed for each training sentence. Similarly, a model trained on the whole training data was used to produce 10 hypotheses for each sentence in the development set. For the reranking experiments, we applied different kernel setups to the two corpora described in Section 3. We enriched the feature set of the base CRF classifier (presented in section 4.3) with the most frequent tag added to each token.
6.2. Reranking Results

Tables 9 and 10 present the reranking results on the test data of both corpora. The results show a 20.29% relative improvement in F-measure for Italian and 21.79% for English.

CFK based on unstructured features achieves higher accuracy than CTK based on structured features. However, the huge amount of subtrees generated by the PT kernel may limit the expressivity of some structural features, e.g., many fragments may only generate noise. This problem is less important with the polynomial kernel where global features are tailored for individual entities. In any case, the experiments demonstrate that both tagging kernels CTK and CFK improve on the CRFs baseline in both languages. This suggests that structured and unstructured features are effective in discriminating between competing NE annotations.

Furthermore, the combination of the two tagging kernels on both standard corpora shows a large improvement in F-measure from 80.34% to 84.33% for Italian and from 84.86% to 88.16% for English. This suggests that these two kernels, corresponding to two kinds of features, complement each other.

To better collocate our results with previous work, we report the state of the art for NER on the Italian (Zanoli et al., 2009; Gesmundo, 2009) and the English (Ratinov and Roth, 2009; Finkel et al., 2005) datasets, in the last two rows (in italic) of each table, where the very last row shows the highest results. This demonstrates that our model outperforms the best Italian NER system and it is close to the state-of-art model for English, which exploits many complex features. Also note that we are very close to the F1 achieved by the best system of CoNLL 2003, i.e. 88.8.

7. Conclusion

In this paper, we have studied the use of discriminative reranking for named entity recognition. The major contribution with respect to previous work is the use of structural kernels for representing hypotheses of the NE annotation over an entire sentence. In particular, we first use a basic NER, e.g., built with CRFs or SVMs, to generate global hypotheses of NE annotation. Then, we apply a reranker, i.e., a classifier based on SVMs and tree kernels, to select the best hypothesis. Such reranker can better select the most accurate hypothesis since it can use features describing the global view of the sentence annotation and its label dependencies. In particular, tree kernels are
applied to a tree representation of the NE annotation. This way, all the possible tree fragments generated by the kernel correspond to features encoding global dependencies between the NE labels and words of the sentence.

Another interesting contribution of our paper is the combination of tree kernels with global features encoded in traditional vectors. This joint model is rather effective as it can capture two different important aspects of the sentence annotation: tree kernels provide structural dependencies between NEs, whereas the vectors can describe global properties without structure, which are less sparse. Additionally, we used innovative features, e.g., the most frequent tag associated with words in an external corpus and the KIM gazetteer.

The comparative results on two well-known benchmarks in Italian and English language suggest that the partial tree kernel applied to our structural representations is rather effective. Indeed, we could improve the state of the art of the Italian NER and achieve almost the same accuracy of very complex model for English. Again, this improvement is due to the richer/global features available to the reranker, which always improve the basic model. Nevertheless, (Ratinov and Roth, 2009), applying very accurate and complex manual feature engineering, provide a higher accuracy for English. This can be due to several factors, e.g., we did not re-implement their features. It is worth noting that their system can be used as basic input of our reranker, thus our model may improve it.

Finally, as our reranking kernels are rather efficient and effective, in the future, we would like to use them for other tasks, e.g., reranking sentence-level relation extraction or coreference resolution annotations.

Acknowledgement

This research has been supported by the EC framework FP7/2007-2013 under the grants #247758: ETERNALS – Trustworthy Eternal Systems via Evolving Software, Data and Knowledge, and #288024: LiMoSINE – Linguistically Motivated Semantic aggregation engiNes.

References


Nancy Chinchor and Patricia Robinson. 1998. Muc-7 named entity task definition. In The MUC.


Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role label-


