Re-Ranking Algorithms for Name Tagging

Heng Ji
Dept. of Computer Science
hengji@cs.nyu.edu

Cynthia Rudin
Center for Neural Science and Courant Institute of Mathematical Sciences
New York University
New York, N.Y. 10003
rudin@nyu.edu

Ralph Grishman
Dept. of Computer Science
grishman@cs.nyu.edu

Abstract

Integrating information from different stages of an NLP processing pipeline can yield significant error reduction. We demonstrate how re-ranking can improve name tagging in a Chinese information extraction system by incorporating information from relation extraction, event extraction, and coreference. We evaluate three state-of-the-art re-ranking algorithms (MaxEnt-Rank, SVMRank, and p-Norm Push Ranking), and show the benefit of multi-stage re-ranking for cross-sentence and cross-document inference.

1 Introduction

In recent years, re-ranking techniques have been successfully applied to enhance the performance of NLP analysis components based on generative models. A baseline generative model produces N-best candidates, which are then re-ranked using a rich set of local and global features in order to select the best analysis. Various supervised learning algorithms have been adapted to the task of re-ranking for NLP systems, such as MaxEnt-Rank (Charniak and Johnson, 2005; Ji and Grishman, 2005), SVMRank (Shen and Joshi, 2003), Voted Perceptron (Collins, 2002; Collins and Duffy, 2002; Shen and Joshi, 2004), Kernel Based Methods (Henderson and Titov, 2005), and RankBoost (Collins, 2002; Collins and Koo, 2003; Kudo et al., 2005).

These algorithms have been used primarily within the context of a single NLP analysis component, with the most intensive study devoted to improving parsing performance. The re-ranking models for parsing, for example, normally rely on structures generated within the baseline parser itself. Achieving really high performance for some analysis components, however, requires that we take a broader view, one that looks outside a single component in order to bring to bear knowledge from the entire NL analysis process. In this paper we will demonstrate the potential of this approach in enhancing the performance of Chinese name tagging within an information extraction application.

Combining information from other stages in the analysis pipeline allows us to incorporate information from a much wider context, spanning the entire document and even going across documents. This will give rise to new design issues; we will examine and compare different re-ranking algorithms when applied to this task.

We shall first describe the general setting and the special characteristics of re-ranking for name tagging. Then we present and evaluate three re-ranking algorithms – MaxEnt-Rank, SVMRank and a new algorithm, p-Norm Push Ranking – for this problem, and show how an approach based on multi-stage re-ranking can effectively handle features across sentence and document boundaries.

2 Prior Work

2.1 Ranking

We will describe the three state-of-the-art supervised ranking techniques considered in this work. Later we shall apply and evaluate these algorithms for re-ranking in the context of name tagging.

Maximum Entropy modeling (MaxEnt) has been extremely successful for many NLP classifi-

Workshop on Computationally Hard Problemsand Joint Inference in Speech and Language Processing, pages 49–56,
New York City, New York, June 2006.

cation tasks, so it is natural to apply it to re-ranking problems. (Charniak and Johnson, 2005) applied MaxEnt to improve the performance of a state-of-art parser; also in (Ji and Grishman, 2005) we used it to improve a Chinese name tagger.

Using SVMRank, (Shen and Joshi, 2003) achieved significant improvement on parse re-ranking. They compared two different sample creation methods, and presented an efficient training method by separating the training samples into subsets.

The last approach we consider is a boosting-style approach. We implement a new algorithm called p-Norm Push Ranking (Rudin, 2006). This algorithm is a generalization of RankBoost (Freund et al. 1998) which concentrates specifically on the top portion of a ranked list. The parameter "p" determines how much the algorithm concentrates at the top.

2.2 Enhancing Named Entity Taggers

There have been a very large number of NE tagger implementations since this task was introduced at MUC-6 (Grishman and Sundheim, 1996). Most implementations use local features and a unifying learning algorithm based on, e.g., an HMM, MaxEnt, or SVM. Collins (2002) augmented a baseline NE tagger with a re-ranker that used only local, NE-oriented features. Roth and Yih (2002) combined NE and semantic relation tagging, but within a quite different framework (using a linear programming model for joint inference).

3 A Framework for Name Re-Ranking

3.1 The Information Extraction Pipeline

The extraction task we are addressing is that of the Automatic Content Extraction (ACE)[1] evaluations. The 2005 ACE evaluation had 7 types of entities, of which the most common were PER (persons), ORG (organizations), LOC (natural locations) and GPE ('geo-political entities' – locations which are also political units, such as countries, counties, and cities). There were 6 types of semantic relations, with 18 subtypes. Examples of these relations are "the CEO of Microsoft" (an *organization-affiliation* relation), "Fred's wife" (a *personal-social* relation), and "a military base in Germany" (a *located* relation). And there were 8 types of events, with 33 subtypes, such as "Kurt Schork died in Sierra Leone yesterday" (a *Die* event), and "Schweitzer founded a hospital in 1913" (a *Start-Org* event).

To extract these elements we have developed a Chinese information extraction pipeline that consists of the following stages:

- Name tagging and name structure parsing (which identifies the internal structure of some names);
- Coreference resolution, which links "mentions" (referring phrases of selected semantic types) into "entities": this stage is a combination of high-precision heuristic rules and maximum entropy models;
- Relation tagging, using a K-nearest-neighbor algorithm to identify relation types and subtypes;
- Event patterns, semi-automatically extracted from ACE training corpora.

3.2 Hypothesis Representation and Generation

Again, the central idea is to apply the baseline name tagger to generate N-Best multiple hypotheses for each sentence; the results from subsequent components are then exploited to re-rank these hypotheses and the new top hypothesis is output as the final result.

In our name re-ranking model, each hypothesis is an NE tagging of *the entire sentence*. For example, "`<PER>John</PER> was born in <GPE>New York</GPE>.`" is one hypothesis for the sentence "John was born in New York".

We apply a HMM tagger to identify four named entity types: Person, GPE, Organization and Location. The HMM tagger generally follows the Nymble model (Bikel et al, 1997), and uses best-first search to generate N-Best hypotheses. It also computes the "margin", which is the difference between the log probabilities of the top two hypotheses. This is used as a rough measure of confidence in the top hypothesis. A large margin indicates greater confidence that the first hypothesis is correct. The margin also determines the number of hypotheses (N) that we will store. Using cross-validation on the training data, we determine the value of N required to include the best

[1] The ACE task description can be found at http://www.itl.nist.gov/iad/894.01/tests/ace/

hypothesis, as a function of the margin. We then divide the margin into ranges of values, and set a value of N for each range, with a maximum of 30.

To obtain the training data for the re-ranking algorithm, we separate the name tagging training corpus into k folders, and train the HMM name tagger on k-1 folders. We then use the HMM to generate N-Best hypotheses $H = \{h_1, h_2, ..., h_N\}$ for each sentence in the remaining folder. Each h_i in H is then paired with its NE F-measure, measured against the key in the annotated corpus.

We define a "crucial pair" as a pair of hypotheses such that, according to F-Measure, the first hypothesis in the pair should be more highly ranked than the second. That is, if for a sentence, the F-Measure of hypothesis h_i is larger than that of h_j, then (h_i, h_j) is a crucial pair.

3.3 Re-Ranking Functions

We investigated the following three different formulations of the re-ranking problem:

- Direct Re-Ranking by Score
  For each hypothesis h_i, we attempt to learn a scoring function $f: H \rightarrow R$, such that $f(h_i) > f(h_j)$ if the F-Measure of h_i is higher than the F-measure of h_j.
- Direct Re-Ranking by Classification
  For each hypothesis h_i, we attempt to learn $f: H \rightarrow \{-1, 1\}$, such that $f(h_i) = 1$ if h_i has the top F-Measure among H; otherwise $f(h_i) = -1$. This can be considered a special case of re-ranking by score.
- Indirect Re-Ranking Function
  For each "crucial" pair of hypotheses (h_i, h_j), we learn $f: H \times H \rightarrow \{-1, 1\}$, such that $f(h_i, h_j) = 1$ if h_i is better than h_j; $f(h_i, h_j) = -1$ if h_i is worse than h_j. We call this "indirect" ranking because we need to apply an additional decoding step to pick the best hypothesis from these pair-wise comparison results.

4 Features for Re-Ranking

4.1 Inferences From Subsequent Stages

Information extraction is a potentially symbiotic pipeline with strong dependencies between stages (Roth and Yih, 2002&2004; Ji and Grishman, 2005). Thus, we use features based on the output of four subsequent stages – name structure parsing, relation extraction, event patterns, and coreference analysis – to seek the best hypothesis.

We included ten features based on name structure parsing to capture the local information missed by the baseline name tagger such as details of the structure of Chinese person names.

The relation and event re-ranking features are based on matching patterns of words or constituents. They serve to correct name boundary errors (because such errors would prevent some patterns from matching). They also exert selectional preferences on their arguments, and so serve to correct name type errors. For each relation argument, we included a feature whose value is the likelihood that relation appears with an argument of that semantic type (these probabilities are obtained from the training corpus and binned). For each event pattern, a feature records whether the types of the arguments match those required by the pattern.

Coreference can link multiple mentions of names provided they have the same spelling (though if a name has several parts, some may be dropped) and same semantic type. So if the boundary or type of one mention can be determined with some confidence, coreference can be used to disambiguate other mentions, by favoring hypotheses which support more coreference. To this end, we incorporate several features based on coreference, such as the number of mentions referring to a name candidate.

Each of these features is defined for individual name candidates; the value of the feature for a hypothesis is the sum of its values over all names in the hypothesis. The complete set of detailed features is listed in (Ji and Grishman, 2006).

4.2 Handling Cross-Sentence Features by Multi-Stage Re-Ranking

Coreference is potentially a powerful contributor for enhancing NE recognition, because it provides information from other sentences and even documents, and it applies to all sentences that include names. For a name candidate, 62% of its coreference relations span sentence boundaries. However, this breadth poses a problem because it means that the score of a hypothesis for a given

sentence may depend on the tags assigned to the same names in other sentences.[2]

Ideally, when we re-rank the hypotheses for one sentence *S*, the other sentences that include mentions of the same name should already have been re-ranked, but this is not possible because of the mutual dependence. Repeated re-ranking of a sentence would be time-consuming, so we have adopted an alternative approach. Instead of incorporating coreference evidence with all other information in one re-ranker, we apply two re-rankers in succession.

In the first re-ranking step, we generate new rankings for all sentences based on name structure, relation and event features, which are all sentence-internal evidence. Then in a second pass, we apply a re-ranker based on coreference between the names in each hypothesis of sentence *S* and the mentions in the top-ranking hypothesis (from the first re-ranker) of all other sentences.[3] In this way, the coreference re-ranker can propagate globally (across sentences and documents) high-confidence decisions based on the other evidence. In our final MaxEnt Ranker we obtained a small additional gain by further splitting the first re-ranker into three separate steps: a name structure based re-ranker, a relation based re-ranker and an event based re-ranker; these were incorporated in an incremental structure.

4.3 Adding Cross-Document Information

The idea in coreference is to link a name mention whose tag is locally ambiguous to another mention that is unambiguously tagged based on local evidence. The wider a net we can cast, the greater the chance of success. To cast the widest net possible, we have used cross-document coreference for the test set. We cluster the documents using a cross-entropy metric and then treat the entire cluster as a single document.

We take all the name candidates in the top *N* hypotheses for each sentence in each cluster *T* to construct a "query set" *Q*. The metric used for the clustering is the cross entropy *H(T, d)* between the distribution of the name candidates in *T* and document *d*. If *H(T, d)* is smaller than a threshold then we add *d* to *T*. *H(T, d)* is defined by:

$$H(T,d) = -\sum_{x \in Q} prob(T,x) \times \log prob(d,x).$$

We built these clusters two ways: first, just clustering the test documents; second, by augmenting these clusters with related documents retrieved from a large unlabeled corpus (with document relevance measured using cross-entropy).

5 Re-Ranking Algorithms

We have been focusing on selecting appropriate ranking algorithms to fit our application. We choose three state-of-the-art ranking algorithms that have good generalization ability. We now describe these algorithms.

5.1 MaxEnt-Rank

5.1.1 Sampling and Pruning

Maximum Entropy models are useful for the task of ranking because they compute a reliable ranking probability for each hypothesis. We have tried two different sampling methods – single sampling and pairwise sampling.

The first approach is to use each single hypothesis h_i as a sample. Only the best hypothesis of each sentence is regarded as a positive sample; all the rest are regarded as negative samples. In general, absolute values of features are not good indicators of whether a hypothesis will be the best hypothesis for a sentence; for example, a co-referring mention count of 7 may be excellent for one sentence and poor for another. Consequently, in this single-hypothesis-sampling approach, we convert each feature to a Boolean value, which is true if the original feature takes on its maximum value (among all hypotheses) for this hypothesis. This does, however, lose some of the detail about the differences between hypotheses.

In pairwise sampling we used each pair of hypotheses (h_i, h_j) as a sample. The value of a feature for a sample is the difference between its values for the two hypotheses. However, considering all pairs causes the number of samples to grow quadratically ($O(N^2)$) with the number of hypotheses, compared to the linear growth with best/non-best sampling. To make the training and

2 For in-document coreference, this problem could be avoided if the tagging of an entire document constituted a hypothesis, but that would be impractical … a very large N would be required to capture sufficient alternative taggings in an N-best framework.

3 This second pass is skipped for sentences for which the confidence in the top hypothesis produced by the first re-ranker is above a threshold.

test procedures more efficient, we prune the data in several ways.

We perform pruning by beam setting, removing candidate hypotheses that possess very low probabilities from the HMM, and during training we discard the hypotheses with very low F-measure scores. Additionally, we incorporate the pruning techniques used in (Chiang 2005), by which any hypothesis with a probability lower than α times the highest probability for one sentence is discarded. We also discard the pairs very close in performance or probability.

5.1.2 Decoding

If *f* is the ranking function, the MaxEnt model produces a probability for each un-pruned "crucial" pair: *prob(f(h_i, h_j) = 1)*, i.e., the probability that for the given sentence, h_i is a better hypothesis than h_j. We need an additional decoding step to select the best hypothesis. Inspired by the caching idea and the multi-class solution proposed by (Platt et al. 2000), we use a dynamic decoding algorithm with complexity O(n) as follows.

We scale the probability values into three types: CompareResult *(h_i, h_j)* = "better" if *prob(f(h_i, h_j) = 1)* > δ_1, "worse" if *prob(f(h_i, h_j) = 1)* < δ_2, and "unsure" otherwise, where $\delta_1 \geqslant \delta_2$.[4]

Prune

```
for i = 1 to n
   Num = 0;
   for j = 1 to n and j≠i
        If CompareResult(h_i, h_j) = "worse"
            Num++;
   if Num> β then discard h_i from H
```

Select

```
Initialize: i = 1, j = n
while (i<j)
    if CompareResult(h_i, h_j) = "better"
       discard h_j from H;
       j--;
    else if CompareResult(h_i, h_j) = "worse"
       discard h_i from H;
       i++;
    else break;
```

Output

If the number of remaining hypotheses in H is 1, then output it as the best hypothesis; else propagate all hypothesis pairs into the next re-ranker.

5.2 SVMRank

We implemented an SVM-based model, which can theoretically achieve very low generalization error. We use the SVMLight package (Joachims, 1998), with the pairwise sampling scheme as for MaxEnt-Rank. In addition we made the following adaptations: we calibrated the SVM outputs, and separated the data into subsets.

To speed up training, we divided our training samples into *k* subsets. Each subset contains *N(N-1)/k* pairs of hypotheses of each sentence.

In order to combine the results from these different SVMs, we must calibrate the function values; the output of an SVM yields a distance to the separating hyperplane, but not a probability. We have applied the method described in (Shen and Joshi, 2003), to map SVM's results to probabilities via a sigmoid. Thus from the k^{th} SVM, we get the probability for each pair of hypotheses:

$$prob(f_k(h_i, h_j) = 1),$$

namely the probability of h_i being better than h_j. Then combining all *k* SVMs' results we get:

$$Z(h_i, h_j) = \prod_k prob(f_k(h_i, h_j) = 1).$$

So the hypothesis h_i with maximal value is chosen as the top hypothesis:

$$\arg\max_{h_i}(\prod_j Z(h_i, h_j)).$$

5.3 P-Norm Push Ranking

The third algorithm we have tried is a general boosting-style supervised ranking algorithm called p-Norm Push Ranking (Rudin, 2006). We describe this algorithm in more detail since it is quite new and we do not expect many readers to be familiar with it.

The parameter "p" determines how much emphasis (or "push") is placed closer to the top of the ranked list, where p≥1. The p-Norm Push Ranking algorithm generalizes RankBoost (take p=1 for RankBoost). When p is set at a large value, the rankings at the top of the list are given higher priority (a large "push"), at the expense of possibly making misranks towards the bottom of the list.

[4] In the final stage re-ranker we use $\delta_1 = \delta_2$ so that we don't generate the output of "unsure", and one hypothesis is finally selected.

Since for our application, we do not care about the rankings at the bottom of the list (i.e., we do not care about the exact rank ordering of the bad hypotheses), this algorithm is suitable for our problem. There is a tradeoff for the choice of p; larger p yields more accurate results at the very top of the list for the training data. If we want to consider more than simply the very top of the list, we may desire a smaller value of p. Note that larger values of p also require more training data in order to maintain generalization ability (as shown both by theoretical generalization bounds and experiments). If we want large p, we must aim to choose the largest value of p that allows generalization, given our amount of training data. When we are working on the first stage of re-ranking, we consider the whole top portion of the ranked list, because we use the rank in the list as a feature for the next stage. Thus, we have chosen the value p_1=4 (a small "push") for the first re-ranker. For the second re-ranker we choose p_2=16 (a large "push").

The objective of the p-Norm Push Ranking algorithm is to create a scoring function $f: H \rightarrow R$ such that for each crucial pair (h_i, h_j), we shall have $f(h_i) > f(h_j)$. The form of the scoring function is $f(h_i) = \sum \alpha_k g_k(h_i)$, where g_k is called a weak ranker: $g_k : H \rightarrow [0,1]$. The values of α_k are determined by the p-Norm Push algorithm in an iterative way.

The weak rankers g_k are the features described in Section 4. Note that we sometimes allow the algorithm to use both g_k and $g'_k(h_i)=1-g_k(h_i)$ as weak rankers, namely when g_k has low accuracy on the training set; this way the algorithm itself can decide which to use.

As in the style of boosting algorithms, real-valued weights are placed on each of the training crucial pairs, and these weights are successively updated by the algorithm. Higher weights are given to those crucial pairs that were misranked at the previous iteration, especially taking into account the pairs near the top of the list. At each iteration, one weak ranker g_k is chosen by the algorithm, based on the weights. The coefficient α_k is then updated accordingly.

6 Experiment Results

6.1 Data and Resources

We use 100 texts from the ACE 04 training corpus for a blind test. The test set included 2813 names: 1126 persons, 712 GPEs, 785 organizations and 190 locations. The performance is measured via Precision (P), Recall (R) and F-Measure (F).

The baseline name tagger is trained from 2978 texts from the People's Daily news in 1998 and also 1300 texts from ACE training data.

The 1,071,285 training samples (pairs of hypotheses) for the re-rankers are obtained from the name tagger applied on the ACE training data, in the manner described in Section 3.2.

We use OpenNLP[5] for the MaxEnt-Rank experiments. We use SVM^{light} (Joachims, 1998) for SVMRank, with a linear kernel and the soft margin parameter set to the default value. For the p-Norm Push Ranking, we apply 33 weak rankers, i.e., features described in Section 4. The number of iterations was fixed at 110, this number was chosen by optimizing the performance on a development set of 100 documents.

6.2 Effect of Pairwise Sampling

We have tried both single-hypothesis and pairwise sampling (described in section 5.1.1) in MaxEnt-Rank and p-Norm Push Ranking. Table 1 shows that pairwise sampling helps both algorithms. MaxEnt-Rank benefited more from it, with precision and recall increased 2.2% and 0.4% respectively.

Model		P	R	F
MaxEnt-Rank	Single Sampling	89.6	90.2	89.9
	Pairwise Sampling	91.8	90.6	91.2
p-Norm Push	Single Sampling	91.4	89.6	90.5
	Pairwise Sampling	91.2	90.8	91.0

Table 1. Effect of Pairwise Sampling

6.3 Overall Performance

In Table 2 we report the overall performance for these three algorithms. All of them achieved improvements on the baseline name tagger. MaxEnt yields the highest precision, while p-Norm Push Ranking with $p_2 = 16$ yields the highest recall.

A larger value of "p" encourages the p-Norm Push Ranking algorithm to perform better near the top of the ranked list. As we discussed in section

[5] http://maxent.sourceforge.net/index.html

5.3, we use $p_1 = 4$ (a small "push") for the first re-ranker and $p_2 = 16$ (a big "push") for the second re-ranker. From Table 2 we can see that $p_2 = 16$ obviously performed better than $p_2 = 1$. In general, we have observed that for $p_2 \leqslant 16$, larger p_2 correlates with better results.

Model	P	R	F
Baseline	87.4	87.6	87.5
MaxEnt-Rank	91.8	90.6	91.2
SVMRank	89.5	90.1	89.8
p-Norm Push Ranking (p_2=16)	91.2	90.8	91.0
p-Norm Push Ranking (p_2=1, RankBoost)	89.3	89.7	89.5

Table 2. Overall Performance

The improved NE results brought better performance for the subsequent stages of information extraction too. We use the NE outputs from MaxEnt-Ranker as inputs for coreference resolver and relation tagger. The ACE value[6] of entity detection (mention detection + coreference resolution) is increased from 73.2 to 76.5; the ACE value of relation detection is increased from 34.2 to 34.8.

6.4 Effect of Cross-document Information

As described in Section 4.3, our algorithm incorporates cross-document coreference information. The 100 texts in the test set were first clustered into 28 topics (clusters). We then apply cross-document coreference on each cluster. Compared to single document coreference, cross-document coreference obtained 0.5% higher F-Measure, using MaxEnt-Ranker, improving performance for 15 of these 28 clusters.

These clusters were then extended by selecting 84 additional related texts from a corpus of 15,000 unlabeled Chinese news articles (using a cross-entropy metric to select texts). 24 clusters gave further improvement, and an overall 0.2% further improvement on F-Measure was obtained.

6.5 Efficiency

Model	Training	Test
MaxEnt-Rank	7 hours	55 minutes
SVMRank	48 hours	2 hours
p-Norm Push Ranking	3.2 hours	10 minutes

Table 3. Efficiency Comparison

In Table 3 we summarize the running time of these three algorithms in our application.

7 Discussion

We have shown that the other components of an IE pipeline can provide information which can substantially improve the performance of an NE tagger, and that these improvements can be realized through a variety of re-ranking algorithms. MaxEnt re-ranking using binary sampling and p-Norm Push Ranking proved about equally effective.[7] p-Norm Push Ranking was particularly efficient for decoding (about 10 documents / minute), although no great effort was invested in tuning these procedures for speed.

We presented methods to handle cross-sentence inference using staged re-ranking and to incorporate additional evidence through document clustering.

An N-best / re-ranking strategy has proven effective for this task because with relatively small values of N we are already able to include highly-rated hypotheses for most sentences. Using the values of N we have used throughout (dependent on the margin of the baseline HMM, but never above 30), the upper bound of N-best performance (if we always picked the top-scoring hypothesis) is 97.4% recall, 96.2% precision, F=96.8%.

Collins (2002) also applied re-ranking to improve name tagging. Our work has addressed both name identification and classification, while his only evaluated name identification. Our re-ranker used features from other pipeline stages, while his were limited to local features involving lexical information and 'word-shape' in a 5-token window. Since these feature sets are essentially disjoint, it is quite possible that a combination of the two could yield even further improvements. His boosting algorithm is a modification of the method in (Freund et al., 1998), an adaptation of AdaBoost, whereas our p-Norm Push Ranking algorithm can emphasize the hypotheses near the top, matching our objective.

Roth and Yih (2004) combined information from named entities and semantic relation tagging, adopting a similar overall goal but using a quite different approach based on linear programming.

[6] The ACE04 value scoring metric can be found at: http://www.nist.gov/speech/tests/ace/ace04/doc/ace04-evalplan-v7.pdf

[7] The features were initially developed and tested using the MaxEnt re-ranker, so it is encouraging that they worked equally well with the p-Norm Push Ranker without further tuning.

They limited themselves to name classification, assuming the identification given. This may be a natural subtask for English, where capitalization is a strong indicator of a name, but is much less useful for Chinese, where there is no capitalization or word segmentation, and boundary errors on name identification are frequent. Expanding their approach to cover identification would have greatly increased the number of hypotheses and made their approach slower. In contrast, we adjust the number of hypotheses based on the margin in order to maintain efficiency while minimizing the chance of losing a high-quality hypothesis.

In addition we were able to capture selectional preferences (probabilities of semantic types as arguments of particular semantic relations as computed from the corpus), whereas Roth and Yih limited themselves to hard (boolean) type constraints.

Acknowledgment

This material is based upon work supported by the Defense Advanced Research Projects Agency under Contract No. HR0011-06-C-0023, and the National Science Foundation under Grant IIS-00325657 and a postdoctoral research fellowship. Any opinions, findings and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the U. S. Government.

References

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance Learning Name-finder. *Proc. ANLP1997*. pp. 194-201. Washington, D.C.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking. *Proc. ACL2005*. pp. 173-180. Ann Arbor, USA

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. *Proc. ACL2005*. pp. 263-270. Ann Arbor, USA

Michael Collins. 2002. Ranking Algorithms for Named-Entity Extraction: Boosting and the Voted Perceptron. *Proc. ACL 2002*. pp. 489-496

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. *Proc. ACL2002*. pp. 263-270. Philadelphia, USA

Michael Collins and Terry Koo. 2003. Discriminative Reranking for Natural Language Parsing. *Journal of Association for Computational Linguistics*. pp. 175-182.

Yoav Freund, Raj Iyer, Robert E. Schapire and Yoram Singer. 1998. An efficient boosting algorithm for combining preferences. *Machine Learning: Proceedings of the Fifteenth International Conference.* pp. 170-178

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference - 6: A brief history. *Proc. COLING1996*,. pp. 466-471. Copenhagen.

James Henderson and Ivan Titov. 2005. Data-Defined Kernels for Parse Reranking Derived from Probabilistic Models. *Proc. ACL2005.* pp. 181-188. Ann Arbor, USA.

Heng Ji and Ralph Grishman. 2005. Improving Name Tagging by Reference Resolution and Relation Detection. *Proc. ACL2005.* pp. 411-418. Ann Arbor, USA.

Heng Ji and Ralph Grishman. 2006. Analysis and Repair of Name Tagger Errors. *Proc. ACL2006 (POSTER).* Sydney, Australia.

Thorsten Joachims. 1998. Making large-scale support vector machine learning practical. *Advances in Kernel Methods: Support Vector Machine.* MIT Press.

Taku Kudo, Jun Suzuki and Hideki Isozaki. 2005. Boosting-based Parse Reranking Derived from Probabilistic Models. *Proc. ACL2005.* pp. 189-196. Ann Arbor, USA.

John Platt, Nello Cristianini, and John Shawe-Taylor. 2000. Large margin dags for multiclass classification. *Advances in Neural Information Processing Systems 12.* pp. 547-553

Dan Roth and Wen-tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. *Proc. CONLL2004.* pp. 1-8

Dan Roth and Wen-tau Yih. 2002. Probabilistic Reasoning for Entity & Relation Recognition. *Proc. COLING2002.* pp. 835-841

Cynthia Rudin. 2006. Ranking with a p-Norm Push. *Proc. Nineteenth Annual Conference on Computational Learning Theory (CoLT 2006)*, Pittsburgh, Pennsylvania.

Libin Shen and Aravind K. Joshi. 2003. An SVM Based Voting Algorithm with Application to Parse ReRanking. *Proc. HLT-NAACL 2003 workshop on Analysis of Geographic References.* pp. 9-16

Libin Shen and Aravind K. Joshi. 2004. Flexible Margin Selection for Reranking with Full Pairwise Samples. *Proc.IJCNLP2004.* pp. 446-455. Hainan Island, China.