# Joint Parsing and Named Entity Recognition

**Jenny Rose Finkel and Christopher D. Manning**
Computer Science Department
Stanford University
Stanford, CA 94305
{jrfinkel|manning}@cs.stanford.edu

## Abstract

For many language technology applications, such as question answering, the overall system runs several independent processors over the data (such as a named entity recognizer, a coreference system, and a parser). This easily results in inconsistent annotations, which are harmful to the performance of the aggregate system. We begin to address this problem with a joint model of parsing and named entity recognition, based on a discriminative feature-based constituency parser. Our model produces a consistent output, where the named entity spans do not conflict with the phrasal spans of the parse tree. The joint representation also allows the information from each type of annotation to improve performance on the other, and, in experiments with the OntoNotes corpus, we found improvements of up to 1.36% absolute F1 for parsing, and up to 9.0% F1 for named entity recognition.

## 1 Introduction

In order to build high quality systems for complex NLP tasks, such as question answering and textual entailment, it is essential to first have high quality systems for lower level tasks. A good (deep analysis) question answering system requires the data to first be annotated with several types of information: parse trees, named entities, word sense disambiguation, etc. However, having high performing, low-level systems is not enough; the assertions of the various levels of annotation must be *consistent* with one another. When a named entity span has crossing brackets with the spans in the parse tree it is usually impossible to effectively combine these pieces of information, and system performance suffers. But, unfortunately, it is still common practice to cobble together independent systems for the various types of annotation, and there is no guarantee that their outputs will be consistent.

This paper begins to address this problem by building a joint model of both parsing and named entity recognition. Vapnik has observed (Vapnik, 1998; Ng and Jordan, 2002) that "one should solve the problem directly and never solve a more general problem as an intermediate step," implying that building a joint model of two phenomena is more likely to harm performance on the individual tasks than to help it. Indeed, it has proven very difficult to build a joint model of parsing and semantic role labeling, either with PCFG trees (Sutton and McCallum, 2005) or with dependency trees. The CoNLL 2008 shared task (Surdeanu et al., 2008) was intended to be about joint dependency parsing and semantic role labeling, but the top performing systems decoupled the tasks and outperformed the systems which attempted to learn them jointly. Despite these earlier results, we found that combining parsing and named entity recognition modestly improved performance on both tasks. Our joint model produces an output which has consistent parse structure and named entity spans, and does a better job at both tasks than separate models with the same features.

We first present the joint, discriminative model that we use, which is a feature-based CRF-CFG parser operating over tree structures augmented with NER information. We then discuss in detail how we make use of the recently developed OntoNotes corpus both for training and testing the model, and then finally present the performance of the model and some discussion of what causes its superior performance, and how the model relates to prior work.

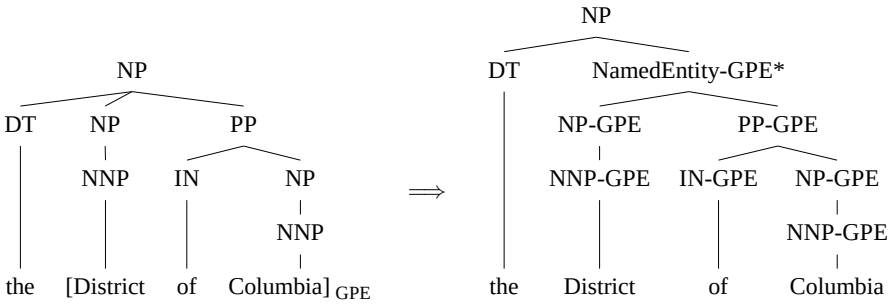


Figure 1: An example of a (sub)tree which is modified for input to our learning algorithm. Starting from the normalized tree discussed in section 4.1, a new *NamedEntity* node is added, so that the named entity corresponds to a single phrasal node. That node, and its descendents, have their labels augmented with the type of named entity. The * on the *NamedEntity* node indicates that it is the root of the named entity.

## 2 The Joint Model

When constructing a joint model of parsing and named entity recognition, it makes sense to think about how the two distinct levels of annotation may help one another. Ideally, a named entity should correspond to a phrase in the constituency tree. However, parse trees will occasionally lack some explicit structure, such as with right branching NPs. In these cases, a named entity may correspond to a contiguous set of children within a subtree of the entire parse. The one thing that should never happen is for a named entity span to have crossing brackets with any spans in the parse tree.

For named entities, the joint model should help with boundaries. The internal structure of the named entity, and the structural context in which it appears, can also help with determining the type of entity. Finding the best parse for a sentence can be helped by the named entity information in similar ways. Because named entities *should* correspond to phrases, information about them should lead to better bracketing. Also, knowing that a phrase is a named entity, and the type of entity, may help in getting the structural context, and internal structure, of that entity correct.

### 2.1 Joint Representation

After modifying the OntoNotes dataset to ensure consistency, which we will discuss in Section 4, we augment the parse tree with named entity information, for input to our learning algorithm. In the cases where a named entity corresponds to multiple contiguous children of a subtree, we add a new *NamedEntity* node, which is the new parent to those children. Now, all named entities correspond to a single phrasal node in the entire tree. We then augment the labels of the phrasal node and its descendents with the type of named entity. We also distinguish between the root node of an entity, and the descendent nodes. See Figure 1 for an illustration. This representation has several benefits, outlined below.

#### 2.1.1 Nested Entities

The OntoNotes data does not contain any nested entities. Consider the named entity portions of the rules seen in the training data. These will look, for instance, like *none* → *none person*, and *organization* → *organization organization*. Because we only allow named entity derivations which we have seen in the data, nested entities are impossible. However, there is clear benefit in a representation allowing nested entities. For example, it would be beneficial to recognize that the *United States Supreme Court* is a an *organization*, but that it also contains a nested *GPE*.[1] Fortunately, if we encounter data which has been annotated with nested entities, this representation will be able to handle them in a natural way. In the given example, we would have a derivation which includes *organization* → *GPE organization*. This information will be helpful for correctly labeling nested entities such as *New Jersey Supreme Court*, because the model will learn how nested entities tend to decompose.

#### 2.1.2 Feature Representation for Named Entities

Currently, named entity recognizers are usually constructed using sequence models, with linear chain

[1] As far as we know, GENIA (Kim et al., 2003) is the only corpus currently annotated with nested entities.

conditional random fields (CRFs) being the most common. While it is possible for CRFs to have links that are longer distance than just between adjacent words, most of the benefit is from local features, over the words and labels themselves, and from features over adjacent pairs of words and labels. Our joint representation allows us to port both types of features from such a named entity recognizer. The local features can be computed at the same time the features over parts of speech are computed. These are the leaves of the tree, when only the named entity for the current word is known.[2] The pairwise features, over adjacent labels, are computed at the same time as features over binary rules. Binarization of the tree is necessary for efficient computation, so the trees consist solely of unary and binary productions. Because of this, for all pairs of adjacent words within an entity, there will be a binary rule applied where one word will be under the left child and the other word will be under the right child. Therefore, we compute features over adjacent words/labels when computing the features for the binary rule which joins them.

### 2.2 Learning the Joint Model

We construct our joint model as an extension to the discriminatively trained, feature-rich, conditional random field-based, CRF-CFG parser of (Finkel et al., 2008). Their parser is similar to a chart-based PCFG parser, except that instead of putting probabilities over rules, it puts *clique potentials* over local subtrees. These unnormalized potentials know what span (and split) the rule is over, and arbitrary features can be defined over the local subtree, the span/split and the words of the sentence. The inside-outside algorithm is run over the clique potentials to produce the partial derivatives and normalizing constant which are necessary for optimizing the log likelihood.

### 2.3 Grammar Smoothing

Because of the addition of named entity annotations to grammar rules, if we use the grammar as read off the treebank, we will encounter problems with sparseness which severely degrade performance. This degradation occurs because of CFG rules which only occur in the training data augmented with named entity information, and because of rules which only occur without the named entity information. To combat this problem, we added extra rules, unseen in the training data.

#### 2.3.1 Augmenting the Grammar

For every rule encountered in the training data which has been augmented with named entity information, we add extra copies of that rule to the grammar. We add one copy with all of the named entity information stripped away, and another copy for each other entity type, where the named entity augmentation has been changed to the other entity type.

These additions help, but they are not sufficient. Most entities correspond to noun phrases, so we took all rules which had an NP as a child, and made copies of that rule where the NP was augmented with each possible entity type. These grammar additions sufficed to improve overall performance.

#### 2.3.2 Augmenting the Lexicon

The lexicon is augmented in a similar manner to the rules. For every part of speech tag seen with a named entity annotation, we also add that tag with no named entity information, and a version which has been augmented with each type of named entity.

It would be computationally infeasible to allow any word to have any part of speech tag. We therefore limit the allowed part of speech tags for common words based on the tags they have been observed with in the training data. We also augment each word with a distributional similarity tag, which we discuss in greater depth in Section 3, and allow tags seen with other words which belong to the same distributional similarity cluster. When deciding what tags are allowed for each word, we initially ignore named entity information. Once we determine what base tags are allowed for a word, we also allow that tag, augmented with any type of named entity, if the augmented tag is present in the lexicon.

## 3 Features

We defined features over both the parse rules and the named entities. Most of our features are over one or the other aspects of the structure, but not both.

Both the named entity and parsing features utilize the words of the sentence, as well as orthographic and distributional similarity information. For each word we computed a *word shape* which encoded

[2] Note that features can include information about other *words*, because the entire sentence is observed. The features *cannot* include information about the labels of those words.

information about capitalization, length, and inclusion of numbers and other non-alphabetic characters. For the distributional similarity information, we had to first train a distributional similarity model. We trained the model described in (Clark, 2000), with code downloaded from his website, on several hundred million words from the British national corpus, and the English Gigaword corpus. The model we trained had 200 clusters, and we used it to assign each word in the training and test data to one of the clusters.

For the named entity features, we used a fairly standard feature set, similar to those described in (Finkel et al., 2005). For parse features, we used the exact same features as described in (Finkel et al., 2008). When computing those features, we removed all of the named entity information from the rules, so that these features were just over the parse information and not at all over the named entity information.

Lastly, we have the joint features. We included as features each augmented rule and each augmented label. This allowed the model to learn that certain types of phrasal nodes, such as *NP*s are more likely to be named entities, and that certain entities were more likely to occur in certain contexts and have particular types of internal structure.

## 4 Data

For our experiments we used the LDC2008T04 OntoNotes Release 2.0 corpus (Hovy et al., 2006). The OntoNotes project leaders describe it as "a large, multilingual richly-annotated corpus constructed at 90% internanotator agreement." The corpus has been annotated with multiple levels of annotation, including constituency trees, predicate structure, word senses, coreference, and named entities. For this work, we focus on the parse trees and named entities. The corpus has English and Chinese portions, and we used only the English portion, which itself has been split into seven sections: ABC, CNN, MNB, NBC, PRI, VOA, and WSJ. These sections represent a mix of speech and newswire data.

### 4.1 Data Inconsistencies

While other work has utilized the OntoNotes corpus (Pradhan et al., 2007; Yu et al., 2008), this is the first work to our knowledge to simultaneously model the multiple levels of annotation available. Because this is a new corpus, still under development, it is not surprising that we found places where the data was inconsistently annotated, namely with crossing brackets between named entity and tree annotations.

In the places where we found inconsistent annotation it was rarely the case that the different levels of annotation were inherently inconsistent, but rather inconsistency results from somewhat arbitrary choices made by the annotators. For example, when the last word in a sentence ends with a period, such as *Corp.*, one period functions both to mark the abbreviation and the end of the sentence. The convention of the Penn Treebank is to separate the final period and treat it as the end of sentence marker, but when the final word is also part of an entity, that final period was frequently included in the named entity annotation, resulting in the sentence terminating period being part of the entity, and the entity not corresponding to a single phrase. See Figure 2 for an illustration from the data. In this case, we removed the terminating period from the entity, to produce a consistent annotation.

Overall, we found that 656 entities, out of 55,665 total, could not be aligned to a phrase, or multiple contiguous children of a node. We identified and corrected the following sources of inconsistencies:

**Periods and abbreviations.** This is the problem described above with the *Corp.* example. We corrected it by removing the sentence terminating final period from the entity annotation.

**Determiners and PPs.** Noun phrases composed of a nested noun phrase and a prepositional phrase were problematic when they also consisted of a determiner followed by an entity. We dealt with this by flattening the nested NP, as illustrated in Figure 3. As we discussed in Section 2.1, this tree will then be augmented with an additional node for the entity (see Figure 1).

**Adjectives and PPs.** This problem is similar to the previous problem, with the difference being that there are also adjectives preceding the entity. The solution is also similar to the solution to the previous problem. We moved the adjectives from the nested NP into the main NP.

These three modifications to the data solved most, but not all, of the inconsistencies. Another source of problems was conjunctions, such as *North and South Korea*, where *North and South* are a phrase, but *South Korea* is an entity. The rest of the errors seemed to be due to annotation errors and other

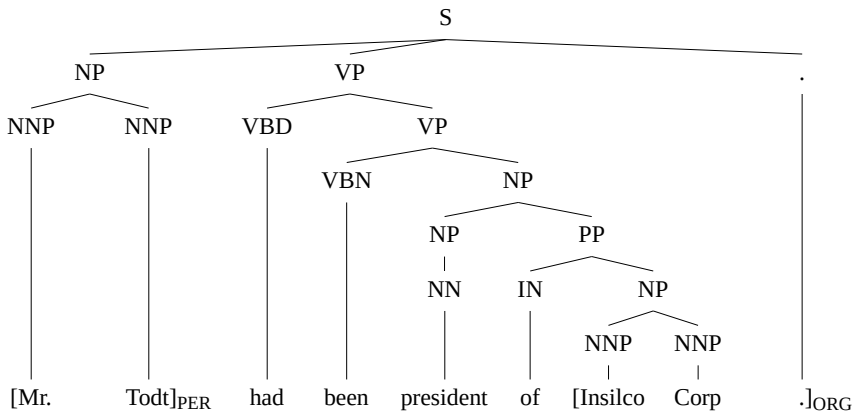


Figure 2: An example from the data of inconsistently labeled named entity and parse structure. The inclusion of the final period in the named entity results in the named entity structure having crossing brackets with the parse structure.

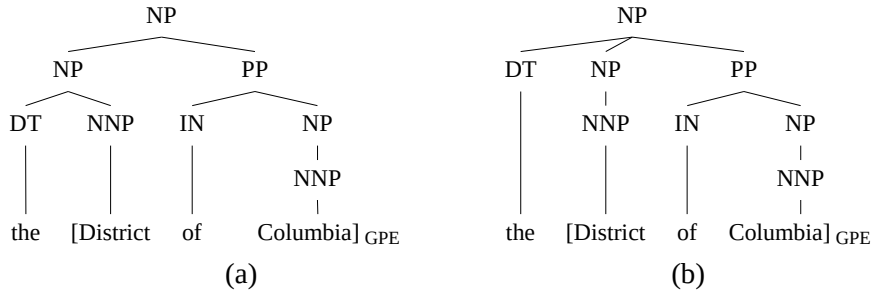


Figure 3: (a) Another example from the data of inconsistently labeled named entity and parse structure. In this instance, we flatten the nested NP, resulting in (b), so that the named entity corresponds to a contiguous set of children of the top-level NP.

random weirdnesses. We ended up unable to make 0.4% of the entities consistent with the parses, so we omitted those entities from the training and test data.

One more change we made to the data was with respect to possessive NPs. When we encountered noun phrases which ended with *(POS 's)* or *(POS ')*, we modified the internal structure of the NP. Originally, these NPs were flat, but we introduced a new nested NP which contained the entire contents of the original NP except for the POS. The original NP label was then changed to PossNP. This change is motivated by the status of *'s* as a phrasal affix or clitic: It is the NP preceding *'s* that is structurally equivalent to other NPs, not the larger unit that includes *'s*. This change has the additional benefit in this context that more named entities will correspond to a single phrase in the parse tree, rather than a contiguous set of phrases.

### 4.2 Named Entity Types

The data has been annotated with eighteen types of entities. Many of these entity types do not occur very often, and coupled with the relatively small amount of data, make it difficult to learn accurate entity models. Examples are *work of art*, *product*, and *law*. Early experiments showed that it was difficult for even our baseline named entity recognizer, based on a state-of-the-art CRF, to learn these types of entities.[3] As a result, we decided to merge all but the three most dominant entity types into into one general entity type called *misc*. The result was four distinct entity types: *person*, *organization*, *GPE* (geo-political entity, such as a city or a country), and *misc*.

[3]The difficulties were compounded by somewhat inconsistent and occasionally questionable annotations. For example, the word *today* was usually labeled as a *date*, but about 10% of the time it was not labeled as anything. We also found several strange *work of art*s, including *Stanley Cup* and the *U.S.S. Cole*.

| | Training | | Testing | |
|---|---|---|---|---|
| | Range | # Sent. | Range | # Sent. |
| ABC | 0–55 | 1195 | 56–69 | 199 |
| CNN | 0–375 | 5092 | 376–437 | 1521 |
| MNB | 0–17 | 509 | 18–25 | 245 |
| NBC | 0–29 | 552 | 30–39 | 149 |
| PRI | 0–89 | 1707 | 90–112 | 394 |
| VOA | 0–198 | 1512 | 199–264 | 383 |

Table 1: Training and test set sizes for the six datasets in sentences. The file ranges refer to the numbers within the names of the original OntoNotes files.

## 5 Experiments

We ran our model on six of the OntoNotes datasets described in Section 4,[4] using sentences of length 40 and under (approximately 200,000 annotated English words, considerably smaller than the Penn Treebank (Marcus et al., 1993)). For each dataset, we aimed for roughly a 75% train / 25% test split. See Table 1 for the the files used to train and test, along with the number of sentences in each.

For comparison, we also trained the parser without the named entity information (and omitted the *NamedEntity* nodes), and a linear chain CRF using just the named entity information. Both the baseline parser and CRF were trained using the exact same features as the joint model, and all were optimized using stochastic gradient descent. The full results can be found in Table 2. Parse trees were scored using *evalB* (the extra *NamedEntity* nodes were ignored when computing evalB for the joint model), and named entities were scored using entity F-measure (as in the CoNLL 2003 *conlleval*).[5]

While the main benefit of our joint model is the ability to get a consistent output over both types of annotations, we also found that modeling the parse and named entities jointly resulted in improved performance on both. When looking at these numbers, it is important to keep in mind that the sizes of the training and test sets are significantly smaller than the Penn Treebank. The largest of the six datasets, CNN, has about one seventh the amount of training data as the Penn Treebank, and the smallest, MNB, has around 500 sentences from which to train. Parse performance was improved by the joint model for five of the six datasets, by up to 1.36%. Looking at the parsing improvements on a per-label basis, the largest gains came from improved identication of NML consituents, from an F-score of 45.9% to 57.0% (on all the data combined, for a total of 420 NML constituents). This label was added in the new treebank annotation conventions, so as to identify internal left-branching structure inside previously flat NPs. To our surprise, performance on NPs only increased by 1%, though over 12,949 constituents, for the largest improvement in absolute terms. The second largest gain was on PPs, where we improved by 1.7% over 3,775 constituents. We tested the significance of our results (on all the data combined) using Dan Bikel's randomized parsing evaluation comparator[6] and found that both the precision and recall gains were significant at $p \leq 0.01$.

Much greater improvements in performance were seen on named entity recognition, where most of the domains saw improvements in the range of 3–4%, with performance on the *VOA* data improving by nearly 9%, which is a 45% reduction in error. There was no clear trend in terms of precision versus recall, or the different entity types. The first place to look for improvements is with the boundaries for named entities. Once again looking at all of the data combined, in the baseline model there were 203 entities where part of the entity was found, but one or both boundaries were incorrectly identified. The joint model corrected 72 of those entities, while incorrectly identifying the boundaries of 37 entities which had previously been correctly identified. In the baseline NER model, there were 243 entities for which the boundaries were correctly identified, but the type of entity was incorrect. The joint model corrected 80 of them, while changing the labels of 39 entities which had previously been correctly identified. Additionally, 190 entities were found which the baseline model had missed entirely, and 68 enti-

[4] These datasets all consistently use the new conventions for treebank annotation, while the seventh WSJ portion is currently still annotated in the original 1990s style, and so we left the WSJ portion aside.

[5] Sometimes the parser would be unable to parse a sentence (less than 2% of sentences), due to restrictions in part of speech tags. Because the underlying grammar (ignoring the additional named entity information) was the same for both the joint and baseline parsers, it is the case that whenever a sentence is unparseable by either the baseline or joint parser it is in fact unparsable by both of them, and would affect the parse scores of both models equally. However, the CRF is able to named entity tag any sentence, so these unparsable sentences had an effect on the named entity score. To combat this, we fell back on the baseline CRF model to get named entity tags for unparsable sentences.

[6] Available at http://www.cis.upenn.edu/ dbikel/software.html

| | | Parse Labeled Bracketing | | | Named Entities | | | Training |
|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ | Time |
| ABC | Just Parse | 70.18% | 70.12% | **70.15%** | | – | | 25m |
| | Just NER | | – | | 76.84% | 72.32% | 74.51% | |
| | Joint Model | 69.76% | 70.23% | 69.99% | 77.70% | 72.32% | **74.91%** | 45m |
| CNN | Just Parse | 76.92% | 77.14% | 77.03% | | – | | 16.5h |
| | Just NER | | – | | 75.56% | 76.00% | 75.78% | |
| | Joint Model | 77.43% | 77.99% | **77.71%** | 78.73% | 78.67% | **78.70%** | 31.7h |
| MNB | Just Parse | 63.97% | 67.07% | 65.49% | | – | | 12m |
| | Just NER | | – | | 72.30% | 54.59% | 62.21% | |
| | Joint Model | 63.82$ | 67.46% | **65.59%** | 71.35% | 62.24% | **66.49%** | 19m |
| NBC | Just Parse | 59.72% | 63.67% | 61.63% | | – | | 10m |
| | Just NER | | – | | 67.53% | 60.65% | 63.90% | |
| | Joint Model | 60.69% | 65.34% | **62.93%** | 71.43% | 64.81% | **67.96%** | 17m |
| PRI | Just Parse | 76.22% | 76.49% | 76.35% | | – | | 2.4h |
| | Just NER | | – | | 82.07% | 84.86% | 83.44% | |
| | Joint Model | 76.88% | 77.95% | **77.41%** | 86.13% | 86.56% | **86.34%** | 4.2h |
| VOA | Just Parse | 76.56% | 75.74% | 76.15% | | – | | 2.3h |
| | Just NER | | – | | 82.79% | 75.96% | 79.23% | |
| | Joint Model | 77.58% | 77.45% | **77.51%** | 88.37% | 87.98% | **88.18%** | 4.4h |

Table 2: Full parse and NER results for the six datasets. Parse trees were evaluated using evalB, and named entities were scored using macro-averaged F-measure (conlleval).

ties were lost. We tested the statistical significance of the gains (of all the data combined) using the same sentence-level, stratified shuffling technique as Bikel's parse comparator and found that both precision and recall gains were significant at $p < 10^{-4}$.

An example from the data where the joint model helped improve both parse structure and named entity recognition is shown in Figure 4. The output from the individual models is shown in part (a), with the output from the named entity recognizer shown in brackets on the words at leaves of the parse. The output from the joint model is shown in part (b), with the named entity information encoded within the parse. In this example, the named entity *Egyptian Islamic Jihad* helped the parser to get its surrounding context correct, because it is improbable to attach a PP headed by *with* to an *organization*. At the same time, the surrounding context helped the joint model correctly identify *Egyptian Islamic Jihad* as an *organization* and not a *person*. The baseline parser also incorrectly added an extra level of structure to the person name *Osama Bin Laden*, while the joint model found the correct structure.

## 6 Related Work

A pioneering antecedent for our work is (Miller et al., 2000), who trained a Collins-style generative parser (Collins, 1997) over a syntactic structure augmented with the *template entity* and *template relations* annotations for the MUC-7 shared task. Their sentence augmentations were similar to ours, but they did not make use of features due to the generative nature of their model. This approach was not followed up on in other work, presumably because around this time nearly all the activity in named entity and relation extraction moved to the use of discriminative sequence models, which allowed the flexible specification of feature templates that are very useful for these tasks. The present model is able to bring together both these lines of work, by integrating the strengths of both approaches.

There have been other attempts in NLP to jointly model multiple levels of structure, with varying degrees of success. Most work on joint parsing and semantic role labeling (SRL) has been disappointing, despite obvious connections between the two tasks. Sutton and McCallum (2005) attempted to jointly model PCFG parsing and SRL for the CoNLL 2005 shared task, but were unable to improve performance on either task. The CoNLL 2008 shared task (Surdeanu et al., 2008) was joint dependency parsing and SRL, but the top performing systems decoupled the tasks, rather than building joint models. Zhang and Clark (2008) successfully built a joint

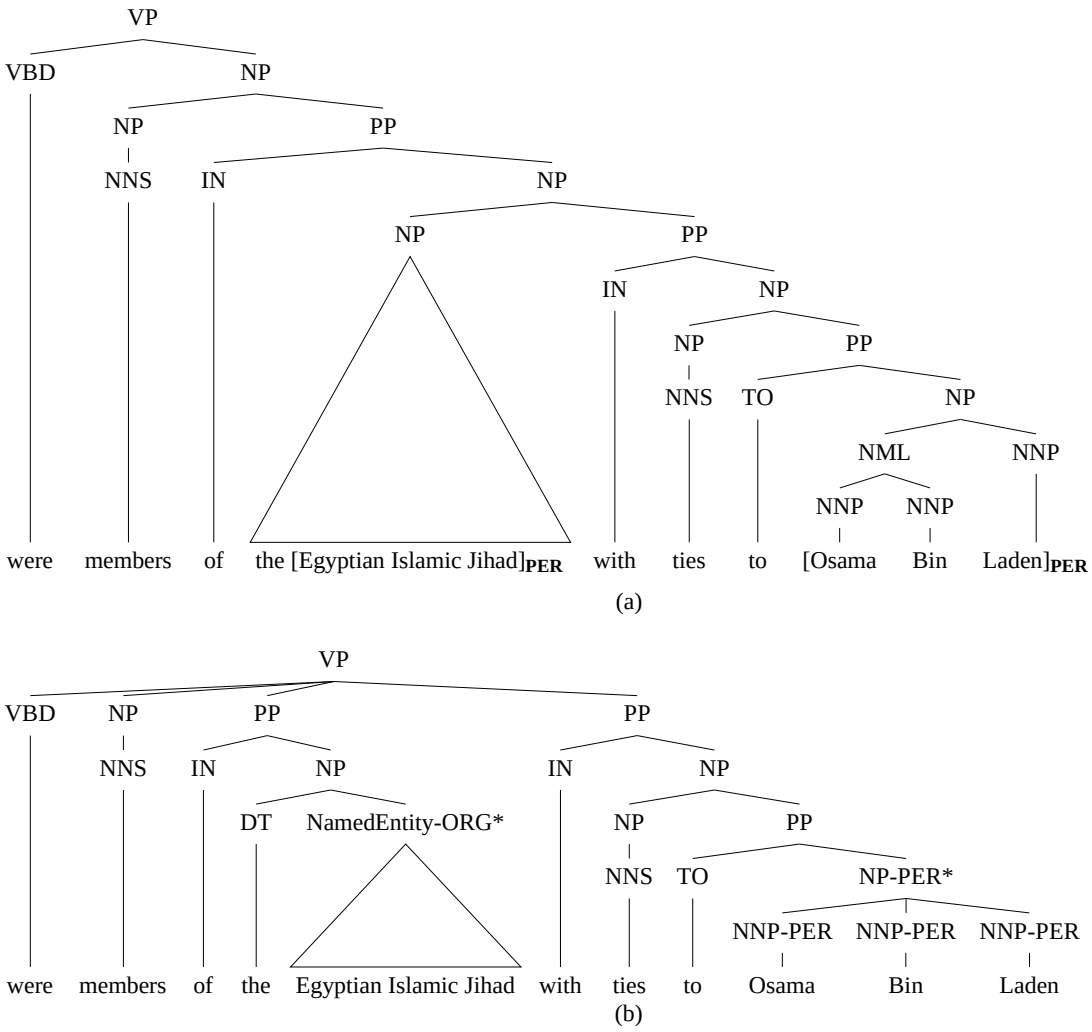


Figure 4: An example for which the joint model helped with both parse structure and named entity recognition. The individual models (a) incorrectly attach the PP, label *Egyptian Islamic Jihad* as a *person*, and incorrectly add extra internal structure to *Osama Bin Laden*. The joint model (b) gets both the structure and the named entity correct.

model of Chinese word segmentation and parts of speech using a single perceptron.

An alternative approach to joint modeling is to take a pipelined approach. Previous work on linguistic annotation pipelines (Finkel et al., 2006; Hollingshead and Roark, 2007) has enforced consistency from one stage to the next. However, these models are only used at test time; training of the components is still independent. These models also have the potential to suffer from search errors and are not guaranteed to find the optimal output.

## 7 Conclusion

We presented a discriminatively trained joint model of parsing and named entity recognition, which improved performance on both tasks. Our model is based on a discriminative constituency parser, with the data, grammar, and features carefully constructed for the joint task. In the future, we would like to add other levels of annotation available in the OntoNotes corpus to our model, including word sense disambiguation and semantic role labeling.

## Acknowledgements

The first author is supported by a Stanford Graduate Fellowship. This paper is based on work funded in part by the Defense Advanced Research Projects Agency through IBM. The content does not necessarily reflect the views of the U.S. Government, and no official endorsement should be inferred. We also wish to thank the creators of OntoNotes, without which this project would not have been possible.

## References


Alexander Clark. 2000. Inducing syntactic categories by context distribution clustering. In *Proc. of Conference on Computational Natural Language Learning*, pages 91–94, Lisbon, Portugal.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL 1997*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005*.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *EMNLP 2006*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based conditional random field parsing. In *ACL/HLT-2008*.

Kristy Hollingshead and Brian Roark. 2007. Pipeline iteration. In *ACL 2007*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *HLT-NAACL 2006*.

Jin-Dong Kim, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. 2003. Genia corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl. 1):i180–i182.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *In 6th Applied Natural Language Processing Conference*, pages 226–233.

Andrew Ng and Michael Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NIPS)*.

Sameer S. Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in ontonotes. *International Conference on Semantic Computing*, 0:446–453.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL)*, Manchester, UK.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Conference on Natural Language Learning (CoNLL)*.

V. N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

Liang-Chih Yu, Chung-Hsien Wu, and Eduard Hovy. 2008. OntoNotes: Corpus cleanup of mistaken agreement using word sense disambiguation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1057–1064.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *ACL 2008*.