Applying Sentence Simplification to the CoNLL-2008 Shared Task

David Vickrey and **Daphne Koller**
Stanford University
Stanford, CA 94305-9010
{dvickrey,koller}@cs.stanford.edu

Abstract

Our submission to the CoNLL-2008 shared task (Surdeanu et al., 2008) focused on applying a novel method for semantic role labeling to the shared task. Our system first simplifies each sentence to be labeled using a set of *hand-constructed* rules; the weights of the system are trained on semantic role labeling data to generate simplifications which are as useful as possible for semantic role labeling. Our system is only a semantic role labeling system, and thus did not receive a score for Syntactic Dependencies (or, by extension, a score for the complete problem). Unlike most systems in the shared task, our system took constituency parses as input. On the subtask of semantic dependencies, our system obtained an F1 score of 76.17, the highest in the open task. In this paper we give a high-level overview of the sentence simplification system, and discuss and analyze the modifications to this system required for the CoNLL-2008 shared task.

1 Sentence Simplification

The main technical interest of our method is a sentence simplification system. This system is described in depth in (Vickrey and Koller, 2008); for lack of space, we omit many details, including a discussion of related work, from this paper.

Current semantic role labeling systems rely primarily on syntactic features in order to identify and classify roles. Features derived from a syntactic parse of the sentence have proven particularly useful (Gildea and Jurafsky, 2002). For example, the syntactic subject of "eat" is nearly always the ARG0. An example sentence with extracted path features is shown in Figure 1.

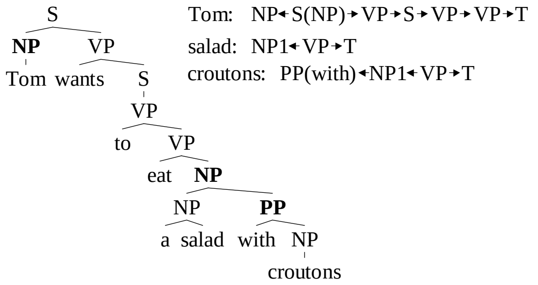

Figure 1: Constituency parse with path features for verb "eat".

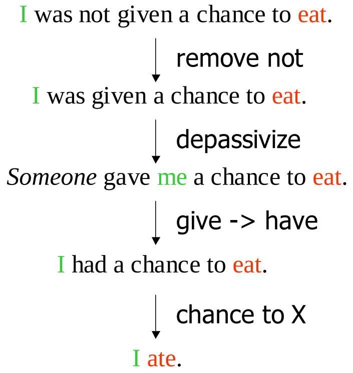
I was not given a chance to eat.
remove not
I was given a chance to eat.
depassivize
Someone gave me a chance to eat.
give -> have
I had a chance to eat.
chance to X
I ate.

Figure 2: Example simplification

A major problem with this approach is that the path from a phrase to the verb can be quite complicated. In the sentence "He expected to receive a prize for winning," the path from "win" to its ARG0, "he", involves the verbs "expect" and "receive" and the preposition "for." The corresponding path through the parse tree likely occurs a small number of times (or not at all) in the training corpus. If the test set contained exactly the same sentence but with "expected" replaced by "did not expect" we would extract a different parse path feature; therefore, as far as the classifier is concerned, the syntax of the two sentences is totally unrelated.

The idea of our method is to learn a mapping from full, complicated sentences to simplified sen-

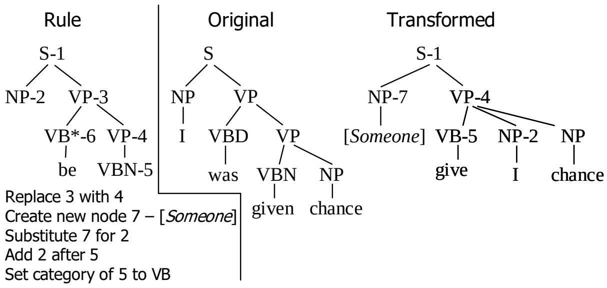

Figure 3: Rule for depassivizing a sentence

tences. Figure 2 shows an example of a series of simplifications applied to the sentence "I was not given a chance to eat." Our method combines hand-written syntactic simplification rules with machine learning, which determines which rules to prefer. We then use the output of the simplification system as input to an SRL system that is trained to label simplified sentences.

There are several reasons to simplify sentences before doing semantic role labeling. First, labeling simplified sentences is much easier than labeling raw sentences. Second, by mapping all sentences to a common, canonical form, we can generalize more effectively across sentences with differing syntax. Third, our model is effective at sharing information across verbs, since most of our simplification rules apply equally well regardless of the target verb. These latter two benefits are particularly important for verbs with few labeled training instances; using training examples efficiently can lead to considerable gains in performance.

Note that unlike most participants in the CoNLL-2008 Shared Task (Surdeanu et al., 2008), our model took as input constituency parses as generated by the Charniak parser (specifically, we used the parses provided with the CoNLL-2005 shared task distribution). This was the only labeled data used that was not available in the closed task.

1.1 Transformation Rules

At the center of our sentence simplification system is a hand-written set of *transformation rules*. A transformation rule takes as input a parse tree and produces as output a different, changed parse tree. Since our goal is to produce a simplified version of the sentence, the rules are designed to bring all sentences toward the same common format.

A rule (see left side of Figure 3) consists of two parts. The first is a "tree regular expression", a tree fragment with optional constraints at each node. The rule assigns numbers to each node which are referred to in the second part of the rule. Formally, a rule node X *matches* a parse-tree node A if: (1) All constraints of node X (e.g., constituent category, head word, etc.) are satisfied by node A. (2) For each child node Y of X, there is a child B of A that matches Y; two children of X cannot be matched to the same child B. There are no other requirements. A can have other children besides those matched, and leaves of the rule pattern can match to internal nodes of the parse (corresponding to entire phrases in the original sentence). For example, the same rule is used to simplify both "I had a chance to eat," and "I had a chance to eat a sandwich," (into "I ate," and "I ate a sandwich,").

Rule Category	#	Original	Simplified
Sentence normalization	24	Thursday, I slept.	I slept Thursday.
Sentence extraction	4	I said he slept.	He slept.
Passive	5	I was hit by a car.	A car hit me.
Misc Collapsing/Rewriting	20	John, a lawyer, ...	John is a lawyer.
Conjunctions	8	I ate and slept.	I ate.
Verb Collapsing/Rewriting	14	I must eat.	I eat.
Verb Raising/Control (basic)	17	I want to eat.	I eat.
Verb RC (ADJP/ADVP)	6	I am likely to eat.	I eat.
Verb RC (Noun)	7	I have a chance to eat.	I eat.
Modified nouns	8	The food I ate ...	Float(The food) I ate.
Floating nodes	5	Float(The food) I ate.	I ate the food.
Inverted sentences	7	Nor will I eat.	I will eat.
Questions	7	Will I eat?	I will eat.
Possessive	7	John's chance to eat...	John has a chance to eat.
Verb acting as PP/NP	7	Including tax, the total...	The total includes tax.
"Make" rewrites	8	Salt makes food tasty.	Food is tasty.

Table 1: Rule categories with sample simplifications. Target verbs are underlined.

The second part of the rule is a series of simple steps that are applied to the matched nodes. For example, one type of simple step applied to the pair of nodes (X,Y) removes X from its current parent and adds it as the final child of Y. Figure 3 shows the depassivizing rule and the result of applying it to the sentence "I was given a chance." The transformation steps are applied sequentially from top to bottom. Any nodes not matched are unaffected by the transformation; they remain where they are relative to their parents. For example, "chance" is not matched by the rule, and thus remains as a child of the VP headed by "give."

1.2 Rule Set

Altogether, we currently have 154 (mostly unlexicalized) rules. Table 1 shows a summary of our rule-set, grouped by type. Note that each row lists only one possible sentence and simplification rule from that category; many of the categories handle a variety of syntax patterns. Our rule set was developed by analyzing performance and coverage on the PropBank WSJ training set; neither the development set nor (of course) the test set were used during rule creation. Please refer to (Vickrey and Koller, 2008) for more details about the rule set.

In the context of the CoNLL-2008 Shared Task, the rule set might be viewed as consisting of outside information. Since we only submitted a system to the open task, this was not an issue.

1.3 Generating Simple Sentences

We now describe how to produce all possible simplified sentences for a given input sentence. We maintain a set of derived parses S which is initialized to contain only the original, untransformed parse. One iteration of the algorithm consists of applying every possible matching transformation rule to every parse in S, and adding all resulting parses to S. With carefully designed rules, repeated iterations are guaranteed to converge; that is, we eventually arrive at a set $\hat{S}$ such that if we apply an iteration of rule application to $\hat{S}$, no new parses are added. Note that we simplify the whole sentence without respect to a particular verb.

We then find all parses in $\hat{S}$ that have "eat" as the main verb. We call such a parse a *valid simple sentence*; this is exactly the canonicalized version of the sentence which our simplification rules are designed to produce.

1.4 Labeling Simple Sentences

For a particular sentence/target verb pair s, v, the output from the previous section is a set $S^{sv} = \{t_i^{sv}\}_i$ of valid simple sentences. From the training set, we now extract a set of *role patterns* $G^v = \{g_j^v\}_j$ for each verb v. For example, a common role pattern for "give" is that of "I gave him a sandwich". We represent this pattern as $g_1^{give} = \{ARG0 = Subject\ NP,\ ARG1 = Postverb\ NP2,\ ARG2 = Postverb\ NP1\}$.

For each simple sentence $t_i^{sv} \in S^{sv}$, we apply all extracted role patterns g_j^v to t_i^{sv}, obtaining a set of possible role labelings. We call a simple sentence/role labeling pair a *simple labeling* and denote the set of candidate simple labelings $C^{sv} = \{c_k^{sv}\}_k$.

1.5 Probabilistic Model

Given a (possibly large) set of candidate simple labelings C^{sv}, we need to select a correct one. We assign a score to each candidate based on its features: which rules were used to obtain the simple sentence, which role pattern was used, and features about the assignment of constituents to roles. The set of extracted features for the sentence "I was given a sandwich by John" with simplification "John gave me a sandwich" is shown in Figure 4.

Rule = Depassivize
Pattern = {ARG0 = Subj NP, ARG1 = PV NP2, ARG2 = PV NP1}
Role = ARG0, Head Word = John
Role = ARG1, Head Word = sandwich
Role = ARG2, Head Word = I
Role = ARG0, Category = NP
Role = ARG1, Category = NP
Role = ARG2, Category = NP
Role = ARG0, Position = Subject NP
Role = ARG1, Position = Postverb NP2
Role = ARG2, Position = Postverb NP1

Figure 4: Features for "John gave me a sandwich."

We now define a log-linear model which assigns a probability to each candidate simple labeling based on its score. Specifically, the probability of a simple labeling c_k^{sv} with respect to a weight vector $\mathbf{w}$ is $P(c_k^{sv}) = \frac{e^{\mathbf{w}^T \mathbf{f}_k^{sv}}}{\sum_{k'} e^{\mathbf{w}^T \mathbf{f}_{k'}^{sv}}}$.

Unfortunately, we do not have labeled examples of correct simplifications. To get around this, we treat the correct simplification as a hidden variable. Thus, we say that the probability of a particular semantic role labeling is $\sum_{c_k^{sv} \in K^{sv}} P(c_k^{sv})$. This leads to our final objective,

$$\sum_{s,v} \left(\log \frac{\sum_{c_k^{sv} \in K^{sv}} e^{\mathbf{w}^T \mathbf{f}_k^{sv}}}{\sum_{c_{k'}^{sv} \in C^{sv}} e^{\mathbf{w}^T \mathbf{f}_{k'}^{sv}}} \right) - \frac{\mathbf{w}^T \mathbf{w}}{2\sigma^2}.$$

We train our model by optimizing the objective using standard methods, specifically BFGS. Due to the summation over the hidden variable representing the choice of simplification (not observed in the training data), our objective is not convex. Thus, we are not guaranteed to find a global optimum; in practice we have gotten good results using the initialization of setting all weights to 0.

2 Baseline Model

In addition to our simplification system, we also built a high-performing logistic regression model for semantic role labeling, which we refer to as **Baseline**. This model uses a slightly modified version of the features used in (Pradhan et al., 2005). This model was also trained on the PropBank training set, using Charniak constituency parses.

Both our simplification model and **Baseline** produce labeled constituencies. Since we were required to produce semantic dependency relations, we simply labeled the head word of each constituent with the role chosen by the model.

3 Labeling Nouns

The 2008 shared task requires systems to label the arguments of both nouns and verbs. However, our sentence simplification system was built to handle only verbs. While in principle the model can naturally be extended to label nouns by the addition of further syntactic simplification rules, we were not able to complete this extension in time for the contest deadline. Instead, we trained our **Baseline** model to label the arguments of nouns as well as verbs. The features of this model are the same as those used to label verbs, and were not extended to handle special features of nouns.

4 Identifying Predicates

Another important subtask was to identify the predicates to be labeled. In the labeled training corpus, nouns with no labeled arguments are generally skipped (i.e., not labeled as predicates at all). Thus, we made a strong simplifying assumption: if a predicate (either noun or verb) is labeled by our system as having no arguments, we should not label it as being a predicate. On the development set, out of a total of 6390 labeled predicates, only 23 had no labeled arguments; thus, this assumption appears to be quite reasonable.

Our system architecture was as follows. First, we modified the training (and test) set by marking as a potential predicate every word that was either: a) a verb that wasn't "do", "be", or "have" or b) a noun found in the nombank index. Then, we trained our system on *all* potential predicates (not just predicates that were actually labeled). Finally, after applying our classifier to the test data, we removed any predicate with no labeled arguments.

5 Sense-Tagging Predicates

We tried three simple heuristics for sense-labeling the predicates. All of them were applied at the *end* of our pipeline, and thus did not interact with the argument labeling decisions.

The simplest heuristic labeled every predicate as sense 1. A slightly more intelligent heuristic labeled every predicate with its most common sense in the training set. Finally, we extended this heuristic to label each verb with its most common sense for the subcategorization (i.e., set of roles) actually produced by the labeling system. Thus, if one sense was intransitive while the other was transitive, we would be able to distinguish between them (assuming that our labeling system produced the correct set of arguments). For this third heuristic, we ignored all but the core arguments (ARG0 - ARG5). The final heuristic was the most effective, as discussed in the results section.

6 Results

The first stage of **Baseline**, which tries to filter out constituents which are obviously not arguments, took about three hours and approximately 4Gb of memory to train[1]. The second stage, which performs the final classification of arguments, took about four hours and 3Gb of memory to train.

The sentence simplification system, which we will refer to as **Simplification**, works in two steps. First, it generates the set of all possible simplifications for each sentence. This step took a relatively small amount of memory, under 1Gb, but took around 24 hours to complete. The set of simplifications is stored in a compact form; the total storage required for all simplifications of all sentences was roughly 4 times the (compressed) size of the Charniak input parses. The second step, which trains the model using the possible simplifications, took around 12 hours and 3Gb of memory.

We only submitted results for the semantic dependencies portion of the competition. The system we used was the **Combined** system described in (Vickrey and Koller, 2008), which combines the simplification procedure with the **Baseline** model. The **Combined** model was augmented with the modifications described above. Our system achieved an official F1 score on the SRL subtask of 76.17, the highest in the open task. Our results are not strictly comparable to those in the closed task, due to the use of the Charniak parser trained on Penn Treebank constituency parses. However, a comparison still provides insight into the relative strength of our system; our score would place us tied for fourth in the closed challenge for semantic dependencies.

We will now discuss the relative contributions of various components of our system. All results in this section are for TestWSJ + TestBrown.

Our **Combined** model provides the same benefit over **Baseline** as described in (Vickrey and

[1] All runs were done on a dual core 2.66Mhz Xeon machine with 4Gb of RAM

Koller, 2008) for labeling the arguments of verbs[2]. When applied to just verb predicates, the **Combined** model provides a statistically significant improvement of 1.2 points of F1 score over **Baseline**. However, since the CoNLL-2008 shared task adds both labeling of noun dependencies and predicate identification and sense tagging, the gain due to better labeling arguments of verbs is reduced. The **Baseline** model achieves an F1 score of 75.31 on the semantic dependencies task, .86 F1 points lower than the **Combined** system.

Note that while most of this gain is directly due to better verb argument labeling, better verb argument labeling also indirectly slightly improves predicate identification and sense-tagging since we use the predicted arguments for both of of these subtasks. We do in fact see a small increase for labeling and sense-tagging predicates, from 80.72 F1 for the **Baseline** to 80.81 F1 for **Combined**.

As mentioned, we use **Baseline** to label the arguments of nouns. Noun argument labeling appears to be more difficult than verb argument labeling, or at least requires some modification of the features. **Baseline** obtains an F1 score of 75.64 for verbs, but only 68.19 F1 for nouns.

On the subtask of predicate identification, **Combined** achieved an F1 of 90.65. It performed better on verbs than nouns. For predicates with part of speech VB*[3] it scored 95.43 F1; for predicates with part of speech NN*, it scored 85.97 F1. Verbs without arguments are often labeled in the gold data, so the verb score could perhaps be improved by retaining verb predicates without arguments.

As described above, we tried three heuristics for sense-labeling predicates. Our final system used the third heuristic, which chose the most common sense for the set of labeled arguments produced by the system. **Combined** obtained an F1 score of 80.81 on the combined predicate identification/classification task, with a score of 82.58 for verbs and 79.28 for nouns. The decrease in performance by adding classification is *much* larger for verbs than nouns; verb sense classification is apparently significantly more difficult than noun sense classification (at least for verbal nouns).

Table 2 compares the results of the **Combined** system using each of the three heuristics. Going from the simplest heuristic to the third heuristic gained 1.52 points of F1 score on the subtask of predicate identification/classification, and an improvement of .48 F1 score for the overall semantic dependency score. Another interesting thing to note is that all of improvement for noun predicates came from choosing the most common sense instead of always choosing sense 1. On the other hand, using subcategorization information is quite important for sense-tagging verbs.

	Overall	Predicate ID/Class		
Heuristic	Score	All	Verbs	Nouns
Always 1	75.69	79.29	81.26	77.58
Most common	76.02	80.33	81.73	79.21
Best for subcat	**76.17**	**80.81**	**82.58**	**79.28**

Table 2: Relative performance of sense-labeling heuristics

7 Discussion and Future Work

The CoNLL-08 task introduces two new subtasks for labeling semantic dependencies: predicate identification and predicate classification. Our experimental results show that both are non-trivial and suggest that there is room for additional improvement on these subtasks.

We are particularly interested in two extensions to our simplification model related to the 2008 shared task. The first is extending our simplification model to handle the arguments of nouns. As discussed above, there is a large amount of room for improvement for argument labeling of nouns. The second is incorporating uncertainty from the parser into our model. Specifically, we would like to extract a complete parse forest from the Charniak parser and use it as input to our model. This would allow our simplification model to jointly reason about the correct parse, possible simplifications of those parses, and semantic role labelings of the resulting simplified sentences.

References

Gildea, D. and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*.

Pradhan, S., K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*, 60(1-3):11–39.

Surdeanu, M., R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*.

Vickrey, D. and D. Koller. 2008. Sentence simplification for semantic role labeling. *Proceedings of ACL*.

[2] Note that the scoring metrics are different between the CoNLL-2005 and CoNLL-2008 shared tasks. The CoNLL-2005 required the constituent boundaries to be labeled correctly, while the CoNLL-2008 only requires identifying the head word of each argument.

[3] This category includes some nouns, e.g. gerunds.