

# Generative Models for Semantic Role Labeling

Cynthia A. Thompson  
cindi@cs.utah.edu

Siddharth Patwardhan  
sidd@cs.utah.edu  
School of Computing  
University of Utah  
Salt Lake City, UT 84112

Carolyn Arnold  
csarnold@cs.utah.edu

## Abstract

This paper describes the four entries from the University of Utah in the semantic role labeling task of SENSEVAL-3. All the entries took a statistical machine learning approach, using the subset of the FrameNet corpus provided by SENSEVAL-3 as training data. Our approach was to develop a model of natural language generation from semantics, and train the model using maximum likelihood and smoothing. Our models performed satisfactorily in the competition, and can flexibly handle varying permutations of provided versus inferred information.

## 1 Introduction

The goal in the SENSEVAL-3 semantic role labeling task is to identify roles and optionally constituent boundaries for each role, given a natural language sentence, target, and frame. The Utah approach to this task is to apply machine learning techniques to create a model capable of semantically analyzing unseen sentences. We have developed a set of generative models (Jordan, 1999) that have the advantages of flexibility, power, and ease of applicability for semi-supervised learning scenarios. We can supplement any of the generative models with a constituent classifier that determines, given a sentence and parse, which parse constituents are most likely to correspond to a role. We apply the combination to the “hard,” or restricted version of the role labeling task, in which the system is provided only with the sentence, target, and frame, and must determine which sentence constituents to label with roles.

We discuss our overall model, the constituent classifier we use in the hard task, and the classifier’s use at role-labeling time. We entered four sets of answers, as discussed in Section 5. The first two correspond to the “easy” task, in which the *role-bearing constituents* – those parts of the sentence corresponding to a role – are provided to the system with the target and frame. The second two are vari-

ants for the “hard” task. Finally, we discuss Future Work and conclude the paper.

## 2 Role Labeler

Our general approach is to use a generative model defining a joint probability distribution over targets, frames, roles, and constituents. The advantage of such a model is its generality: it can determine the probability of any subset of the variables given values for the others. Three of our entries used the generative model illustrated in Figure 1, and the fourth used a model grouping all roles together, as described further below. The first model functions

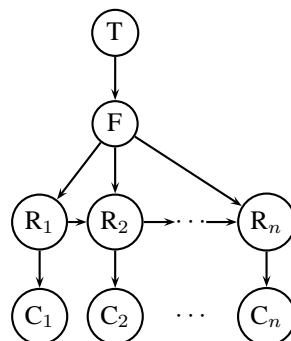


Figure 1: First Order Model.

as follows. First, a target,  $T$ , is chosen, which then generates a frame,  $F$ . The frame generates a (linearized) role sequence,  $R_1$  through  $R_n$  which in turn generates each constituent of the sentence,  $C_1$  through  $C_n$ . Note that, conditioned on a particular frame, the model is just a first-order Hidden Markov Model.

The second generative model treats all roles as a group. It is no longer based on a Hidden Markov model, but all roles are generated, in order, simultaneously. Therefore, the role sequence in Figure 1 is replaced by a single node containing all  $n$  roles. This can be compared to a case-based approach that memorizes all seen role sequences and calculates their likelihood. It is also similar to Gildea & Jurafsky’s (2002) frame element groups, though we distinguish between different role orderings, whereas

they do not. However, we still model constituent generation sequentially.

The FrameNet corpus contains annotations for all of the model components described above. We represent each constituent by its phrasal category together with its head word. As in Gildea & Jurafsky’s (2002) approach, we determine head words from the sentence’s syntactic parse, using a simple heuristic<sup>1</sup> when syntactic alignment with a parse is not available.

We estimate most of the model parameters using a straightforward maximum likelihood estimate based on fully labeled training data. We smooth emission probabilities with phrase type labels, due to the sparseness of head words. To label a test example, consisting of a target, frame, and constituent sequence, with a role label, we use the Viterbi algorithm. For further details, see Thompson (2003).

### 3 Constituent Classification for Role Labeling

To address the “hard” task we build a constituent classifier, whose goal is to detect the role-bearing constituents of a sentence. We use a Naive Bayes classifier from the Weka Machine Learning toolkit (Witten and Frank, 2000) to classify every sentence constituent as role-bearing or not. In our cross-validation studies, Naive Bayes was both accurate and efficient. To generate the training examples for the classifier, we generate a parse tree for every sentence in the SENSEVAL-3 training data, using the Collins (1996) statistical parser. We call each node in this tree a constituent. Once it is trained, the classifier can sift through a new constituent list and decide which are likely to be role-bearing. The selected constituents are passed on to the Role Labeler for labeling with semantic roles, as described in Section 4.

We train the classifier on examples extracted from the SENSEVAL-3 training data. Each example is a list of attributes corresponding to a constituent in a sentence’s parse tree, along with its classification as role-bearing or not. We extract the attributes by traversing each sentence’s parse tree from the root node down to nodes all of whose children are pre-terminals.<sup>2</sup> We create a training example for every visited node.

We decided to use the following attributes from the parse trees and FrameNet examples:

**Target Position:** The position of the target word as

<sup>1</sup>The heuristic chooses the preposition for PP’s and the last word of the phrase for all other phrases.

<sup>2</sup>We later fixed this to traverse the tree to the pre-terminals themselves, as discussed further in Section 5.

being *BEFORE*, *AFTER*, or *CONTAINS* (contained in) the constituent.

**Distance from target:** The number of words between the start of the constituent and target word.

**Depth:** The depth of the constituent in the parse tree.

**Height:** The number of levels in the parse tree below the constituent.

**Word Count:** The number of words in the constituent.

**Path to Target:** Gildea and Jurafsky (2002) show that the path from a constituent node to the node corresponding to the target word is a good indicator that a constituent corresponds to a role. We use the 35 most frequently occurring paths in the training corpus as attribute values, as these cover about 68% of the paths in the training corpus. The remaining paths are specified as “OTHER”.

**Length of Path to Target:** The number of nodes between the constituent and the target in the path.

**Constituent Phrase Type**

**Target POS:** The target word’s part-of-speech – noun, verb, or adjective.

**Frame ID**

By generating examples in the manner described above, we create a data set that is heavily biased towards negative examples – 90.8% of the constituents are not role bearing. Therefore, the classifier can obtain high accuracy by labeling everything as negative. This is undesirable since then no constituents would be passed to the Role Labeler. However, passing all constituents to the labeler would cause it to try to label all of them and thus achieve lower accuracy. This results in the classic precision-recall tradeoff. We chose to try to bias the classifier towards high recall by using a cost matrix that penalizes missed positive examples more than missed negatives. The resulting classifier’s cross-validation precision was 0.19 and its recall was 0.91. If we do not use the cost matrix, the precision is 0.30 and the recall is 0.82. We are still short of our goal of perfect recall and reasonable precision, but this provides a good filtering mechanism for the next step of role labeling.

### 4 Combining Constituent Classification with Role Labeling

The constituent classifier correctly picks out most of the role bearing constituents. However, as we have seen, it still omits some constituents and, as it was designed to, includes several irrelevant constituents per sentence. For this paper, because we plan to improve the constituent classifier further, we did not use it to bias the Role Labeler at training time, but

only used it to filter constituents at test time for the hard task.

When using the classifier with the Role Labeler at testing time, there are two possibilities. First, all constituents deemed relevant by the classifier could be presented to the labeler. However, because we aimed for high recall but possibly low precision, this would allow many irrelevant constituents as input. This both lowers accuracy and increases the computational complexity of labeling. The second possibility is thus to choose some reasonable subset of the positively identified constituents to present to the labeler. The options we considered were a top-down search, a bottom-up search, and a greedy search; we chose a top-down search for simplicity. In this case, the algorithm searches from the root down in the parse tree until it finds a positively labeled constituent. While this assumes that no subtree of a role-bearing constituent is also role-bearing, we discovered that some role-bearing constituents do overlap with each other in the parse trees. However, in the Senseval training corpus, only 1.2% of the sentences contain a (single) overlapping constituent. In future work we plan to investigate alternative approaches for constituent choice.

After filtering via our top down technique, we present the resulting constituent sequence to the role labeler. Since the role labeler is trained on sequences containing only true role-bearing constituents but tested on sequences with potentially missing and potentially irrelevant constituents, this stage provides an opportunity for errors to creep into the process. However, because of the Markovian assumption, the presence of an irrelevant constituent has only local effects on the overall choice of a role sequence.

## 5 Evaluation

The SENSEVAL-3 committee chose 40 of the most frequent 100 frames from FrameNet II for the competition. In experiments with validation sets, our algorithm performed better using only the SENSEVAL-3 training data, as opposed to also using sentences from the remaining frames, so all our models were trained only on that data. We calculated performance using SENSEVAL-3’s scoring software.

We submitted two set of answers for each task. We summarize each system’s performance in Table 1. For the easy task, we used both the grouped (*FEG Easy*) and first order (*FirstOrder Easy*) models. The grouped model performed better on experiments with validation sets, perhaps due to the fact that many frames have a small number of possible

System	Precision	Recall	Overlap
<i>FEG Easy</i>	85.8%	84.9%	85.7%
<i>FirstOrder Easy</i>	72.8%	72.1%	72.5%
<i>CostSens Hard</i>	38.7%	33.5%	29.5%
<i>Hard</i>	35.5%	45.3%	25.5%

Table 1: System Scores.

System	Precision	Recall	Overlap
<i>CostSens Hard</i>	47.2%	42.2%	41.5%
<i>Hard</i>	60.2%	24.7%	57.1%

Table 2: Newer System Scores.

role permutations corresponding to a given number of constituents. In less artificial conditions this version would be less flexible in incorporating both relevant and irrelevant constituents.

For the hard task, we used only the first order model, due both to its greater flexibility and to the low precision of our classifier: if all positively classified constituents were passed to the group model, the sequence length would be greater than any seen at training time, when only correct constituents are given to the labeler. We used both the cost sensitive classifier (*CostSens Hard*) and the regular constituent classifier to filter constituents (*Hard*). There is a precision/recall tradeoff in using the different classifiers. We were surprised how poorly our labeler was performing on validation sets as we prepared our results. We found out that our classifier was omitting about **70%** of the role-bearing constituents from consideration, because they only matched a parse constituent at a pre-terminal node. We fixed this bug after submission, learned a new constituent classifier, and used the same role labeler as before. The improved results are shown in Table 2. Note that our recall has an upper limit of 85.8% due to mismatches between roles and parse tree constituents.

## 6 Future Work

We have identified three problems for future research. First, our constituent classifier should be improved to produce fewer false positives and to include a higher percentage of true positives. To do this, we first plan to enhance the feature set. We will also explore improved approaches to combining the results of the classifier with the role labeler. For example, in preliminary studies, a bottom-up search for positive constituents in the parse tree seems to yield better results than our current top-down approach.

Second, since false positives cannot be entirely

avoided, the labeler needs to better handle constituents that should not be labeled with a role. To solve this problem, we will adapt the idea of *null generated* words from machine translation (Brown et al., 1993). Instead of having a word in the target language that corresponds to no word in the source language, we have a constituent that corresponds to no state in the role sequence.

Finally, we will address roles that do not label a constituent, called *null-instantiated* roles. An example is the sentence “The man drove to the station,” in which the VEHICLE role does not have a constituent, but is implicitly there, since the man obviously drove *something* to the station. This problem is more difficult, since it involves obtaining information not actually in the sentence. One possibility is to consider inserting null-instantiated roles at every step. We will consider only roles seen as null-instantiated at training time. This method will restrict the search space, which would otherwise be extremely large.

## 7 Conclusions

In conclusion, our generative model performs robustly on the easy version of the SENSEVAL-3 role labeling task. The combination of our constituent classifier with the role labeling has more room for improvement, but performed reasonably well considering the difficulties of the task and the sparse feature set that we incorporated into our generative model. Alternative sentence chunking models for semantic analysis, and the extension of our generative models, should lead to future improvements. The key advantage of our approach is the treatment of a sentence’s roles as a sequence. This allows the model to consider relationships between roles as it semantically analyzes a sentence.

## 8 Acknowledgements

This work was supported in part by the Advanced Research and Development Activity’s Advanced Question Answering for Intelligence Program. The basic Role Labeler was developed in collaboration with Chris Manning and Roger Levy.

## References

- P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.
- M. J. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Assoc.*

*for Computational Linguistics*, pages 184–191, Santa Cruz, CA.

- Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28:245–288.

- M. Jordan, editor. 1999. *Learning in Graphical Models*. MIT Press, Cambridge, MA.

- C. A. Thompson, R. Levy, and C. Manning. 2003. A generative model for semantic role labeling. In *Proceedings of the Fourteenth European Conference on Machine Learning*, pages 397–408, Croatia.

- I. Witten and E. Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco.