Discriminative vs. Generative Approaches in Semantic Role Labeling

Deniz Yuret
Koç University
dyuret@ku.edu.tr

Mehmet Ali Yatbaz
Koç University
myatbaz@ku.edu.tr

Ahmet Engin Ural
Koç University
aural@ku.edu.tr

Abstract

This paper describes the two algorithms we developed for the CoNLL 2008 Shared Task "Joint learning of syntactic and semantic dependencies". Both algorithms start parsing the sentence using the same syntactic parser. The first algorithm uses machine learning methods to identify the semantic dependencies in four stages: identification and labeling of predicates, identification and labeling of arguments. The second algorithm uses a generative probabilistic model, choosing the semantic dependencies that maximize the probability with respect to the model. A hybrid algorithm combining the best stages of the two algorithms attains 86.62% labeled syntactic attachment accuracy, 73.24% labeled semantic dependency F1 and 79.93% labeled macro F1 score for the combined WSJ and Brown test sets[1].

1 Introduction

In this paper we describe the system we developed for the CoNLL 2008 Shared Task (Surdeanu et al., 2008). Section 2 describes our approach for identifying syntactic dependencies. For semantic role labeling (SRL), we pursued two independent approaches. Section 3 describes our first approach, where we treated predicate identification and labeling, and argument identification and labeling as four separate machine learning problems. The final program consists of four stages, each stage taking the answers from the previous stage as given and performing its own identification or labeling task based on a model generated from the training set. Section 4 describes our second approach where we used a generative model based on the joint distribution of the predicate, the arguments, their labels and the syntactic dependencies connecting them. Section 5 summarizes our results and suggests possible improvements.

[1]These numbers are slightly higher than the official results due to a small bug in our submission.

2 Syntactic dependencies

We used a non-projective dependency parser based on spanning tree algorithms. The parameters were determined based on the experimental results of the English task in (McDonald et al., 2005), i.e. we used projective parsing and a first order feature set during training. Due to the new representation of hyphenated words in both training and testing data of our shared task and the absence of the gold part of speech (GPOS) column in the test data, the format of the CoNLL08 shared task is slightly different from the format of the CoNLL05 shared task, which is supported by the McDonald's parser. We reformatted the data accordingly. The resulting labeled attachment score on the test set is 87.39% for WSJ and 80.46% for Brown.

3 The 4-stage discriminative approach

Our first approach to SRL consists of four distinct stages: (1) predicate identification, (2) predicate labeling, (3) argument identification, and (4) argument labeling.

A discriminative machine learning algorithm is trained for each stage using the gold input and output values from the training set. The following sections describe the machine learning algorithm, the nature of its input/output, and the feature selection process for each stage. The performance of each stage is compared to a most frequent class baseline and analyzed separately for the two test sets and for nouns and verbs. In addition we look at the performance given the input from the gold data vs. the input from the previous stage.

3.1 Predicate identification

The task of this stage is to determine whether a given word is a nominal or a verb predicate using the dependency-parsed input. As potential predicates we only consider words that appear as a predicate in the training data or have a corresponding PropBank or NomBank XML file. The method constructs feature vectors for each occurrence of a target word in the training and test data. It assigns class labels to the target words in the training data depending on whether a target word is a predicate or not, and finally classifies the test data. We experimented with combinations of the following features for each word in a $2k + 1$ word window around the target: (1) POS(W): the part of speech of the word, (2) DEP(W, HEAD(W)): the syntactic dependency of the word, (3) LEMMA(W): the lemma of the word, (4) POS(HEAD(W)): the part of speech of the syntactic head.

We empirically selected the combination that gives the highest accuracy in terms of the precision and recall scores on the development data. The method achieved its highest score when we used features 1-3 for the target word and features 1-2 for the neighbors in a [-3 +3] word window. TiMBL (Daelemans et al., 2004) was used as the learning algorithm.

Table 1 (4-stage, All1) shows the results of our learning method on the WSJ and Brown test data. The noun and verb results are given separately (Verb1, Noun1). To distinguish the mistakes coming from parsing we also give the results of our method after the gold parse (4-stage-gold). Our results are significantly above the most frequent class baseline which gives 72.3% on WSJ and 65.3% on Brown.

3.2 Predicate labeling

The task of the second stage is deciding the correct frame for a word given that the word is a predicate. The input of the stage is 11-column data, where the columns contain part of speech, lemma and syntactic dependency for each word. The first stage's decision for the frame is indicated by a string in the predicate column. The output of the stage is simply the replacement of that string with the chosen frame of the word. The chosen frame of the word may be word.X, where X is a valid number in PropBank or NomBank.

The statistics of the training data show that by picking the most frequent frame, the system can pick the correct frame in a large percent of the cases. Thus we decided to use the most frequent frame baseline for this stage. If the word is never seen in the training, first frame of the word is picked as default.

In the test phase, the results are as the following; in the Brown data, assuming that the stage 1 is gold, the score is 80.8%, noting that 11% of the predicates are not seen in the training phase. In WSJ, the score based on gold input is 88.3%, and only 5% of the predicates are not seen in the training phase. Table 1 gives the full results for Stage 2 (4-stage, Verb2, Noun2, All2).

3.3 Argument identification

The input data at this stage contains the syntactic dependencies, predicates and their frames. We look at the whole sentence for each predicate and decide whether each word should be an argument of that predicate or not. We mark the words we choose as arguments indicating which predicate they belong to and leave the labeling of the argument type to the next stage. Thus, for each predicate-word pair we have a yes/no decision to make.

As input to the learning algorithm we experimented with representations of the syntactic dependency chain between the predicate and the argument at various levels of granularity. We identified the syntactic dependency chain between the predicate and each potential argument using breadth-first-search on the dependency tree. We tried to represent the chain using various subsets of the following elements: the argument lemma and part-of-speech, the predicate frame and part-of-speech, the parts-of-speech and syntactic dependencies of the intermediate words linking the argument to the predicate.

The syntactic dependencies leading from the argument to the predicate can be in the head-modifier or the modifier-head direction. We marked the direction associated with each dependency relation in the chain description. We also experimented with using fine-grained and coarse-grained parts of speech. The coarse-grained part of speech consists of the first two characters of the Penn Treebank part of speech given in the training set.

We used a simple learning algorithm: choose the answer that is correct for the majority of the instances with the same chain description from the training set. Not having enough detail in the chain description leaves crucial information out that would help with the decision process, whereas having too much detail results in bad classifications due to sparse data. In the end, neither the argument lemma, nor the predicate frame improved the performance. The best results were achieved with a chain description including the coarse parts of speech and syntactic dependencies of each word leading from the argument to the predicate. The results are summarized in Table 1 (4-stage, Verb3, Noun3, All3).

3.4 Argument labeling

The task of this stage is choosing the correct argument tag for a modifier given that it is modifying a particular predicate. Input data format has additional columns indicating which words are arguments for which predicates. There are 54 possible values for a labeled argument. As a baseline we take the most frequent argument label in the training data (All1) which gives 37.8% on the WSJ test set and 33.8% on the Brown test set.

The features to determine the correct label of an argument are either lexical or syntactic. In a few cases, they are combined. The following list gives the set we have used. Link is the type of the syntactic dependency. Direction is left or right, depending the location of the head and the modifier in the sentence. LastLink is the type of the dependency at the end of the dependency chain and firstLink is type of the dependency at the beginning of the dependency chain.

Feature1 : modifierStem + headStem

Feature2 : modifierStem + coarsePosModifier + headStem + coarsePosHead + direction

Feature3 : coarsePosModifier + headPos + firstLink + lastLink + direction

Feature4: modifierStem + coarsePosModifier

The training phase includes building simple histograms based on four features. Feature1 and Feature2 are sparser than the other two features and are better features as they include lexical information. Last two features are less sparse, covering most of the development data, i.e. their histograms give non-zero values in the development phase. In order to match all the instances in the development and use the semantic information, a cascade of the features is implemented similar to the one done by Gildea and Jurafsky(2002), although no weighting and a kind of back-off smoothing is used. First, a match is searched in the histogram of the first feature, if not found it is searched in the following histogram. After a match, the most frequent argument with that match is returned. Table 1 gives the performance (4-stage, Verb4, Noun4, All4).

4 The generative approach

One problem with the four-stage approach is that the later stages provide no feedback to the earlier ones. Thus, a frame chosen because of its high prior probability will not get corrected when we fail to find appropriate arguments for it. A generative model, on the other hand, does not suffer from this problem. The probability of the whole assignment, including predicates, arguments, and their labels, is evaluated together and the highest probability combination is chosen.

4.1 The generative model

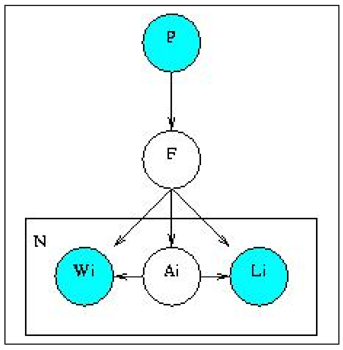

Figure 1: The graphical model depicting the conditional independence assumptions.

Our generative model specifies the distribution of the following random variables: P is the lemma (stem+pos) of a candidate predicate. F is the frame chosen for the predicate (could be null). A_i is the argument label of word i with respect to a given predicate (could be null). W_i is the lemma (stem+pos) of word i. L_i is the syntactic depen-

WSJ	Verb1	Verb2	Verb3	Verb4	Noun1	Noun2	Noun3	Noun4	All1	All2	All3	All4
4-stage	97.1	85.5	85.7	71.7	84.6	78.4	61.1	49.4	90.6	81.8	76.6	63.5
generative	96.1	88.4	83.4	74.0	82.8	79.5	69.8	63.2	89.0	83.6	77.4	69.2
4-stage-gold	**97.4**	88.3	**95.2**	82.7	**85.2**	92.7	70.5	81.9	**91.1**	90.5	86.0	82.4
generative-gold	96.3	**92.6**	91.1	**88.0**	83.4	**95.5**	**80.7**	**86.9**	89.4	**94.0**	**86.7**	**87.5**
hybrid	97.1	89.3	85.7	74.7	84.6	80.9	70.9	64.0	90.6	84.9	79.5	70.2

Brown	Verb1	Verb2	Verb3	Verb4	Noun1	Noun2	Noun3	Noun4	All1	All2	All3	All4
4-stage	93.0	74.5	78.9	59.0	74.4	58.6	52.3	38.8	86.0	68.6	72.8	54.3
generative	91.4	71.7	76.1	60.0	70.8	59.3	54.0	45.3	83.1	66.6	69.6	55.7
4-stage-gold	**93.0**	**80.8**	**93.7**	73.2	**75.7**	80.3	70.1	70.5	**86.5**	80.8	**88.2**	72.4
generative-gold	91.6	80.6	85.8	**78.05**	71.2	**85.9**	**70.5**	**75.1**	83.5	**82.6**	81.8	**77.1**
hybrid	93.0	73.3	78.9	60.4	74.4	62.9	57.6	47.5	86.0	69.3	73.4	57.0

Table 1: The F1 scores for different datasets, models, stages, and predicate parts of speech. The "Verb" in the column heading indicates verbal predicates, "Noun" indicates nominal predicates, "All" indicates all predicates. The numbers 1-4 in column headings indicate the 4 stages: (1) predicate identification, (2) predicate labeling, (3) argument identification, (4) argument labeling. The gold results assume perfect output from the previous stages. The highest number in each column is marked with boldface.

dency chain leading from word i to the given predicate (similar to Section 3.3).

We consider each word in the sentence as a candidate predicate and use the joint distribution of the above variables to find the maximum probability F and A_i labels given P, W_i, and L_i. The graphical model in Figure 1 specifies the conditional independence assumptions we make. Equivalently, we take the following to be proportional to the joint probability of a particular assignment:

$$\Pr(F|P) \prod_i \Pr(A_i|F) \Pr(W_i|FA_i) \Pr(L_i|FA_i)$$

4.2 Parameter estimation

To estimate the parameters of the generative model we used the following methodology:

For $\Pr(F|P)$ we use the maximum likelihood estimate from the training data. As a consequence, frames that were never observed in the training data have zero probability. One exception is lemmas which have not been observed in the training data, for which each frame is considered equally likely.

For $\Pr(A_i|F)$ we also use the maximum likelihood estimate and normalize it using sentence length. For a given argument label we find the expected number of words in a sentence with that label for frame F. We divide this expected number with the length of the given sentence to find $\Pr(A_i|F)$ for a single word. Any leftover probability is given to the null label. If the sentence length is shorter than the expected number of arguments, all probabilities are scaled down proportionally.

For the remaining two terms $\Pr(L_i|F, A_i)$ and $\Pr(W_i|F, A_i)$ using the maximum likelihood estimate is not effective because of data sparseness. The arguments in the million word training data contain about 16,000 unique words and 25,000 unique dependency chains. To handle the sparseness problem we smoothed these two estimates using the part-of-speech argument distribution, i.e. $\Pr(L_i|\text{POS}, A_i)$ and $\Pr(W_i|\text{POS}, A_i)$, where POS represents the coarse part of speech of the predicate.

5 Results and Analysis

Table 1 gives the F1 scores for the two models (4-stage and generative), presented separately for noun and verb predicates and the four stages of predicate identification/labeling, argument identification/labeling. In order to isolate the performance of each stage we also give their scores with gold input. The rest of this section analyzes these results and suggests possible improvements.

A hybrid algorithm: A comparison of the two algorithms show that the 4-stage approach is superior in predicate and verbal-argument identification and the generative algorithm is superior in the labeling of predicates and arguments and nominal-argument identification. This suggests a hybrid algorithm where we restrict the generative model to take the answers for the better stages from the 4-stage algorithm (Noun1, Verb1, Verb3) as given.

Tables 1 and 2 present the results for the hybrid algorithm compared to the 4-stage and generative models.

Data/algorithm	Unlabeled	Labeled
WSJ 4-stage	81.15	69.44
WSJ generative	81.01	73.66
WSJ hybrid	82.94	74.74
Brown 4-stage	76.91	58.76
Brown generative	73.76	59.05
Brown hybrid	77.22	60.80

Table 2: Semantic scores for the 4-stage, generative, and hybrid algorithms

Parsing performance: In order to see the effect of syntactic parsing performance, we ran the hybrid algorithm starting with the gold parse. The labeled semantic score went up to 78.84 for WSJ and 67.20 for Brown, showing that better parsing can add about 4-6% to the overall performance.

Syntactic vs lexical features: Our algorithms use two broad classes of features: information from the dependency parse provides syntactic evidence, and the word pairs themselves provide semantic evidence for a possible relation. To identify their relative contributions, we experimented with two modifications of the generative algorithm: *gen-l* does not use the $\Pr(W_i|FA_i)$ term and *gen-w* does not use the $\Pr(L_i|FA_i)$ term. *gen-l*, using only syntactic information and the predicate, gets a labeled semantic score of 70.97 for WSJ and 58.83 for Brown, a relatively small decrease. In contrast *gen-w*, using only lexical information gets 43.06 for WSJ and 33.17 for Brown causing almost a 40% decrease in performance.

On the other hand, we find that the lexical features are essential for certain tasks. In labeling the arguments of nominal predicates, finding an exact match for the lexical pair guarantees a 90% accuracy. If there is no exact match, the 4-stage algorithm falls back on a syntactic match, which only gives a 75% accuracy.

Future work: The hybrid algorithm shows the strengths and weaknesses of our two approaches. The generative algorithm allows feedback from the later stages to the earlier stages and the 4-stage machine learning approach allows the use of better features. One way to improve the system could be by adding feedback to the 4-stage algorithm (later stages can veto input coming from previous ones), or adding more features to the generative model (e.g. information about neighbor words when predicting F). More importantly, there is no feedback between the syntactic parser and the semantic role labeling in our systems. Treating both problems under the same framework may lead to better results.

Another property of both models is the independence of the argument label assignments from each other. Even though we try to control the number of arguments of a particular type by adjusting the parameters, there are cases when we end up with no assignments for a mandatory argument or multiple assignments where only one is allowed. A more strict enforcement of valence constraints needs to be studied.

The use of smoothing in the generative model was critical, it added about 20% to our final F1 score. This raises the question of finding more effective smoothing techniques. In particular, the jump from specific frames to coarse parts of speech is probably not optimal. There may be intermediate groups of noun and verb predicates which share similar semantic or syntactic argument distributions. Identifying and using such groups will be considered in future work.

References

Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch. 2004. TiMBL: Tilburg memory-Based Learner. Tilburg University.

Gildea, D. and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245 288.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online Large-Margin Training of Dependency Parsers. *Ann Arbor*, 100.

Surdeanu, Mihai, 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-2008).*