Semantic Role Labeling with Structured Perceptron

Maosen Zhang
School of EECS, Peking University
zhangmaosen@pku.edu.cn

Abstract

We develop a semantic role labeling system with structured perceptron. (Collins, 2002) We use Viterbi algorithm in decoding for performance. The core part of our program is written by myself and only some trivial functions are from externel libraries. We achieve 70.83 F1 score on development set.

1 Data Preprocess

First of all, bracketed representation of roles is converted into IOB2 representation. For example, the beginning word of role A0 is tagged "A0-B"; the following wordw of role A0 are tagged "A0-I"; words not in a role are tagged "O".

Moreover, Although the POS tags are provided in the dataset, we still need other information. We use Stanford Parser and Stanford CoreNLP packet to generate Named Entity Recognition (NER) labels and parsing trees for the sentences in the dataset.

2 Structured Perceptron

We use structured perceptron as the sequence labeling model for the task.(Collins, 2002)

$$score_y = \sum_i \lambda_{f_i(x,y)} f_i(x,y)$$

2.1 Features

The features we use in the perceptron are following. In order to improve efficiency, we divide those features into four groups: word features, verb features, relative features, tag features.

Word features are features concerning the current word:

- Word unigram, bigram, trigram: w_i, $w_{i-1}w_i$, w_iw_{i+1}, $w_{i-1}w_iw_{i+1}$, $w_{i-2}w_{i-1}w_i$, $w_iw_{i+1}w_{i+2}$;
- POS tag unigram, bigram, trigram;
- NER tag unigram, bigram, trigram;

Verb features are features concerning the given verb:

- Verb context: word and POS tag unigram, bigram, trigram around the verb;
- Sentence length;

Relative features are features concerning both the current word and the given verb:

- The postion of current word relative to the verb (before, is, after);(Hacioglu et al., 2004)
- The distance between current word and verb (minus means before);
- The path between the current word and the verb in the parsing tree(Xue and Palmer, 2004);

Tag features (actually the transition score between the previous tag and the current tag):

- The semantic role label of previous word.

2.1.1 Collecting Features for instances

In order to improve efficiency, we collect different features in different stages. We initialize the dataset with their word features and verb features before training and store them, so that we can save the time for collecting them during training time. While the relative features and tag features are collected during training and decoding time. Note that the tag features are collected dynamically depending the viterbi process during decoding. We design different functions for getting those features.

Because the word, verb and relative features are static during viterbi process, which are independent of previous decode label, we call them "static features" and we design a function for collecting them together.

2.1.2 Feature Representation

Since the feature vectors are sparse, we use hash table (dict in Python) to store weights for given feature template and label. For example, weights['W=apple']['A0-B'] indicates the weight for feature "current word is apple and tag is A0-B". We store features in a list and retrieve weights for them and compute the score.

2.2 Viterbi Decode

Given a sentence with its features and weights, we use viterbi algorithm to decode the tag sequence.

$$\pi(j,u,v) = max_{<t_1,...,t_{j-2}>} score(t_1,...,t_{j-2},u,v) \tag{1}$$

We use dynamic programming to compute the lattice π:

$$\pi(j,u,v) = max_t(\pi(j-1,t,u)+score(j,t,u,v)) \tag{2}$$

2.3 Training

For each instance of training dataset, we first use viterbi algorithm to decode the tag sequence with current weights. Then we compare the decoded sequence with the golden sequence and update the weights according to the difference between the two sequences. More specificly, let the golden tag sequence be $y_1, y_2, ..., y_n$ and the decoded predicted tag sequence be $z_1, z_2, ..., z_n$. If $y_k \neq z_k$, we will increase the weights for the features of g_k and the transition score of y_{k-1} and y_k, and decrease the weights for the features of p_k and the transition score of z_{k-1} and z_k. (This is extreamly important according to experiment!)

Let Z be decoded sequence:

$$Z = argmax_{t[1:n] \in GEN(w[1:n])} \lambda \cdot f(w[1:n], t[1:n])$$

If $Z_k \neq Y_k$, update:

$$\lambda^* = \lambda + f(X_k, Y_k) - f(X_k, Z_k)$$

2.4 Averaged Perceptron

According to Collins (Collins, 2002), we compute the averaged model parameters of models after training several iterations, and use these parameters to inference the input data. It significantly improved the performance.

Precision	Recall	F1	
Overall	73.67	68.19	**70.83**
A0	66.78	61.15	63.84
A1	74.49	74.06	74.28
A2	59.20	47.25	52.55
A3	54.55	40.00	46.15
A4	100.00	22.22	36.36
AM	80.17	70.85	75.22
Perfect props	41.57	-	-

Table 1: Font guide.

3 Experiment

On development dataset, We got results as Tabel 1 shows.

References

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.