Dependency Parsing with Empty Category*

王柯评 Keping Wang†

1 Overall Method

I use pipelines for this dependency parsing with empty category (EC) task:

(1) Parse tst.nec.

(2) Detect EC in parsed tst.nec, and save as unparsed tst.ec.

(3) Parse tst.ec.

Two (types of) tasks are involved: dependency parsing, and EC detection. For both tasks, I employ classification-based methods, and the core classifiers are SVM implemented using libsvm library. During training process, sentences are decoded into list of Input-Target pairs, which are then passed to the classifier for training. A classifier classifies Input into a specific type of Target. During prediction process, sentences are first decoded into Inputs, then the classifier takes Inputs and output the Targets, and finally sentences are decoded into the desired forms using these Targets.

2 Transition-based Dependency Parsing

Arc-Standard algorithm is used for dependency parsing (emulating Kübler et al. (2009)). For the classifier, Input is State, and Target is Transition.

The features I use include:

*The course project for the EMNLP course at PKU.

†StudentID：1200015412，Mail：kepingwang@yahoo.com.

(1) The first two words in the stack (last two in queue), and the first three words in the buffer.

(2) Postags of the first three words in the stack, and the first four words in the buffer.

(3) Postags/Labels of the leftmost/rightmost child of the first word in stack/buffer for the current tree.

(4) Combination of the postags of the first word in stack and in buffer.

The parameters for this task are:

(1) SVM parameter: C - cost of constraint violation.

(2) SVM parameter: γ parameter for the RBF (Guassian) kernel.

(3) rareCutoff: all words with frequency equal to or lower than this cutoff are seen as "RARE". This parameter is set to 9, not finely tuned because of the limited time I have.

A potential improvement that I haven't had the time to implement is to use beam search. Now without beam search, an optimal Transition could be forever thrown away be a single mis-classification.

Some mis-classifications may result in Transitions that are incompatible with the sentence decoding. For example: A terminal State is one with an empty Buffer. A compatible series of Transitions will finish building the dependency tree when a terminal state is encountered. In this case a series of all Shift Transitions is incompatible with the sentence decoding. For now I hard code some (limited) rules to prevent this sort of incompatibilty from happening, but beam search can potentially be combined with some hard rules to improve performance.

3 Empty Category (EC) Detection

As in the empty category detection paper of Xue and Yang (2013), I use tuple (h, t) (together with the sentence and dependency tree information) as Input to the classifier,

and the Target is whether there is an EC at (h,t), that is, before word t, and its head being h. (The task is simplified to only one type of EC.)

Features included are:

(1) Words of h, t, and p, where p is the word before the EC.

(2) Postags of h, t, and p.

(3) Postag combinations of (h,t) and (t,p).

(4) The number of Verbs/Punctuations between h and t.

(5) The distance between h and t.

(6) The distance between t and the leftmost/rightmost child of h.

(7) The label/postag of the leftmost/rightmost child of h.

One thing about the EC detection task is the data skewness. Since the Input form is (h, t) tuple, and the number of ECs in each sentence are very limited, most tuples will have no EC. This leads to extremely long training time, and high precision and very low recall. To solve this issue, I randomly subset the tuples with no EC in each sentence so that the data is more balanced. As a result, compared to the dependency parsing task, the EC detection task has an additional parameter subsetRatio: what proportion of non-EC tuples are preserved.

4 Parameter Tuning

The parameters to be tuned are C, γ in each of the three stages in the pipeline system, and subsetRatio in stage (2).

Kernel SVM takes at least $O(n^2)$ (n number of observations) to train, and each sentence will be decoded into many Inputs, so train the whole training set (7656 sentences) takes forever. To tune the model in a reasonable period of time, I use a subset of 400 sentences as my training set. This, however, doesn't result in the optimal parameters

since 1) the data distribution might differ and 2) the feature space is different (fewer non-RARE words included in 400 sentences set).

I used 2000 sentences to train the final model used for test data prediciton. The estimated performance is: 85% accuracy for dependency parsing; 50% precision and 60% recall for EC detection.

References

Kübler, Sandra, Ryan McDonald, and Joakim Nivre (2009), "Dependency parsing." *Synthesis Lectures on Human Language Technologies*, 1, 1–127.

Xue, Nianwen and Yaqin Yang (2013), "Dependency-based empty category detection via phrase structure trees." In *HLT-NAACL*, 1051–1060.