Final Project Report

A Shortest Path Dependency Kernel for Relation Extraction

Zehua Mai   N15096707

1. Description

1.1 Task

The Task this project dealing with is a subproblem in relation extraction.

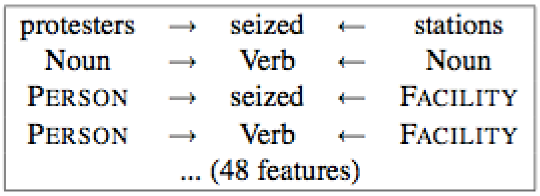
In this task, the predicate (PRED) and the support (SUPPORT) are known for each sentence. The goal is to correctly classify arguments (ARG0, ARG1, ARG2, ARG3) of the specified predicate.

1.2 Approach

Firstly, we parse the dependency tree for a sentence. And then, find the shortest path between two entities in the dependency tree of the sentence.
Since the path is completely lexicalized and consequently leads to overfitting. We associate each word on the path with its word classes (e.g. POS tag) to increase the generality.
The set of features can then be defined as a Cartesian product over these word classes.

$$\begin{bmatrix} \text{protesters} \\ \text{NNS} \\ \text{Noun} \\ \textsc{Person} \end{bmatrix} \times [\rightarrow] \times \begin{bmatrix} \text{seized} \\ \text{VBD} \\ \text{Verb} \end{bmatrix} \times [\leftarrow] \times \begin{bmatrix} \text{stations} \\ \text{NNS} \\ \text{Noun} \\ \textsc{Facility} \end{bmatrix}$$

Feature generation from dependency path.



protesters	→	seized	←	stations
Noun	→	Verb	←	Noun
PERSON	→	seized	←	FACILITY
PERSON	→	Verb	←	FACILITY
		... (48 features)		

Sample Features.

Directly compute the inner product of the feature vectors infeasible, due to the high dimensionality of the feature space. So we apply kernel function with SVM to deal with this situation.

$$K(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} 0, & m \neq n \\ \prod_{i=1}^{n} c(x_i, y_i), & m = n \end{cases}$$

kernel function

$X = X_1X_2...X_m$ and$Y = Y_1Y_2...Y_m$ are two relation instance, where X_i denotes the set of word classes corresponding to position i
$C(X_i , Y_i)$ is the number of common word classes between X_i and Y_i .

1.3 system limits

The data I'm dealing with is huuuuuuuge, and the processing time is really loooooong, which is out of my expection.

The training data is from Penn Treebank Wall Street Journal corpus. There are 10623 sentences in the training file. For each sentence, the program will need 3-4 seconds to extract all the relation instance(i.e. the shortest path between the predicate and all the head words in this sentence) from it. Do a simple math and you'll find it takes 8 hours to merely extract all the sample instances. I don't have that much time so I just extract the instances from the first 2458 sentences and harvest 24003 sample instances.

After that, I have to precompute the kernels before I train them with libsvm. The time complexity for this problem turns out to be quadratic, because I have to compute the kernel between any two pair of instances. I launched a process to compute it at 1 p.m. and now(5:30 p.m) it's still running....

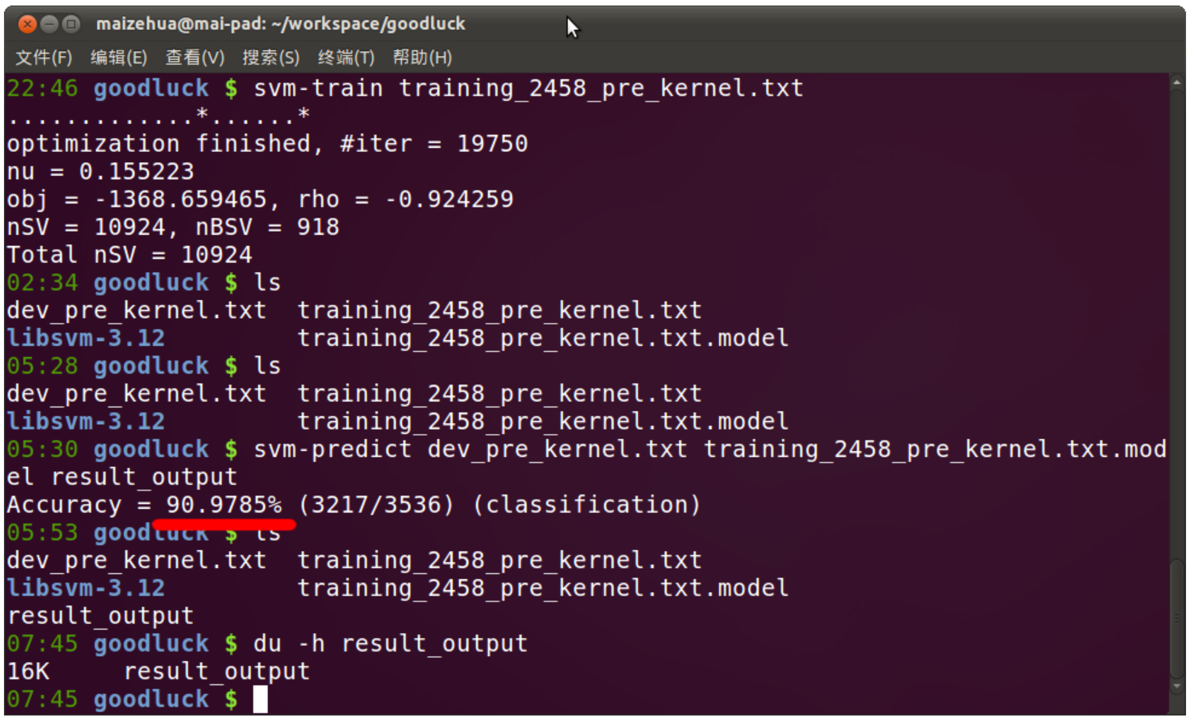
Finally, I got the two precompute kernel file. The size of testing file is 286 MB and the size of training file is 972 MB. They are huge! Maybe there's something wrong?
The libsvm training and predicting process also takes long time(several hours).

1.4 used resources

Stanford parser is used to parse the dependency tree.
libsvm is used to train and predict the file whose kernel is precomputed.
Guava is used for general java programming.

2. Result

2.1 screenshot


maizehua@mai-pad: ~/workspace/goodluck
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
22:46 goodluck $ svm-train training_2458_pre_kernel.txt
.............*......*
optimization finished, #iter = 19750
nu = 0.155223
obj = -1368.659465, rho = -0.924259
nSV = 10924, nBSV = 918
Total nSV = 10924
02:34 goodluck $ ls
dev_pre_kernel.txt  training_2458_pre_kernel.txt
libsvm-3.12         training_2458_pre_kernel.txt.model
05:28 goodluck $ ls
dev_pre_kernel.txt  training_2458_pre_kernel.txt
libsvm-3.12         training_2458_pre_kernel.txt.model
05:30 goodluck $ svm-predict dev_pre_kernel.txt training_2458_pre_kernel.txt.mod
el result_output
Accuracy = 90.9785% (3217/3536) (classification)
05:53 goodluck $ ls
dev_pre_kernel.txt  training_2458_pre_kernel.txt
libsvm-3.12         training_2458_pre_kernel.txt.model
result_output
07:45 goodluck $ du -h result_output
16K     result_output
07:45 goodluck $ 
```

The result seems cheerful, but I don't think this is the true result. Because it is based on relation instances(shortest paths), not on sentences. Please forgive me for having no time implementing a more reasonable scorer.

2.2 links to file

Testing data:	http://dl.dropbox.com/u/42828695/dev_pre_kernel.txt
Trained model:	http://dl.dropbox.com/u/42828695/training_2458_pre_kernel.txt.model
output:	http://dl.dropbox.com/u/42828695/result_output

Since the training data is huge, I have no more space to upload it.

3. Future Work

From a linguistics perspective, more word classes could be used. Such as NE tags, [Noun, active verb, passive verb]. Also, a extension is to extract the relationships between both entities along with their roles.

In the sight of machine learning, a better kernel function could be designed. Also, we can try out different type and arguments of libsvm to find the best.

From a software engineering point of view, this program could be reworked to support multithreading, which I believe will significantly reduce the processing time.

4. Reference

Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In Proceedings of the Human Language TechnologyConference and Conference on Empirical Methods in Natural LanguageProcessing Vancouver, B.C., pp. 724--731, October 2005.