GSLT NLP-1 course Spring 2005

Report from:

Training a Transformation-Based Tagger with Torbjörn Lager's μ-TBL System

Preben Wik
preben@speech.kth.se

The assignment consisted of getting to know, and learning how to use the µ-TBL system, which is a generalized form of transformation-based learning implementation, created by Torbjörn Lager. Specifically, the task consisted of training a simple transformation-based tagger for English, using a subset of the Wall Street Journal corpus, annotated with the Penn Treebank tagset as training data. A relatively small training set was used (ranging from 7.500-60.000 words), since the purpose of the exercise was just to demonstrate the technique.

Part A

Installation and testing.

The windows distribution was easy to install, but unfortunately the manual was outdated. It belongs to an older distribution (version) of the program, i.e. changes have been made to the distribution without updating the manual. Following the manual I started making a test script. Here is a summary of the errors in the script:

```
% My first script
set training_data='data/10kW_suc'.
        #No such file in the distribution
set test_data='data/2kW_suc'.
        #No such file in the distribution
set algorithm='algorithms/brill'.
set templates='templates/brill_templates'.
        #No such file in the distribution
[...]
```

Oh well, this is only noted in case someone wants to update the manual to reflect the current distribution.

Our task in this exercise was to improve upon the performance of the default test script that came with the distribution (test.script). A compressed version of the test script (comments etc. removed) looks like this:

```
% default test.script
set training_data='data/wsj_7500'.
set test_data='data/wsj_test'.
set algorithm='algorithms/brill'.
set templates='templates/test_templates'.
set score_threshold=6.
set accuracy_threshold=0.5.
set verbosity=2.
learn_rule_seq.
test_rule_seq.
save('rules/test.pl').
write_html_error_data.
```

Four ways to improve the tagger were suggested in the assignment, and they were all tested by me.

1. Modify the set of templates.
The fourth line in the test.script: set templates='templates/test_templates'. Points to a file with the template rules used. It looked like this:
tag:A>B <- tag:C@[-1].
tag:A>B <- tag:C@[1].
tag:A>B <- tag:C@[-1,-2].
tag:A>B <- tag:C@[-1] & tag:D@[1].
tag:A>B <- wd:C@[0].
tag:A>B <- wd:C@[1].
tag:A>B <- wd:C@[0] & wd:D@[-1].
tag:A>B <- wd:C@[0] & tag:D@[-1].

Looking at the original templates that Brill used (Brill 1995- as described in Jurafsky&Martin), I tried to modify the templates, and came up with a set that looked like this:
tag:_>_ <- wd:_@[-1].
tag:_>_ <- wd:_@[1].
tag:_>_ <- wd:_@[-2].
tag:_>_ <- wd:_@[2].
tag:_>_ <- wd:_@[-1,-2].
tag:_>_ <- wd:_@[1,2].
tag:_>_ <- wd:_@[-1,-2,-3].
tag:_>_ <- wd:_@[1,2,3].
tag:_>_ <- wd:_@[-1] & wd:_@[1].
tag:_>_ <- wd:_@[-1] & wd:_@[2].
tag:_>_ <- wd:_@[1] & wd:_@[-2].

Testing the new set of templates, leaving everything else from the test script unchanged, the score accuracy went up a little, (96.3% instead of 96.1%) but since the amount of training data used in this initial test was so small, the true effect of this modification is at this point uncertain. I therefore trained the initial template set, as well as the modified template set with a larger set of training data, to review the effect of this.

2. Increase the size of the training data.
The first line in the script points to the training data:
set training_data='data/wsj_7500'.
Four files of training data are available with the distribution. The default with 7500 words, as well as 15.000, 30.000 and 60.000 words.
Leaving everything else from the test script unchanged, the score accuracy went up on every increase of the training data, indicating what also seems intuitively reasonable, that accuracy goes up if we have more data to train the tagger on.
data/wsj_7500 - 8 rules added – Recall/Precision/F-score = 96.1% (original)
data/wsj_15000 - 12 rules added – Recall/Precision/F-score = 96.5%
data/wsj_30000 - 38 rules added – Recall/Precision/F-score = 96.9%
data/wsj_60000 - 43 rules added – Recall/Precision/F-score = 97.0%

If the training data was increased and the modified set of templates were used, the result was different, but not remarkably so:
data/wsj_7500 - 10 rules added – Recall/Precision/F-score = 96.3%
data/wsj_15000 - 18 rules added – Recall/Precision/F-score = 96.4%
data/wsj_30000 - 26 rules added – Recall/Precision/F-score = 96.8%
data/wsj_60000 - 67 rules added – Recall/Precision/F-score = 97.0%

3. Lower the score threshold.
I did two experiments in changing the score threshold. The default was set to 6, and I tried with the values 1 and 2, leaving everything else from the test script unchanged:
set score_threshold=6. - 8 rules added – Recall/Precision/F-score = 96.1%
set score_threshold=1. - 196 rules added – Recall/Precision/F-score = 96.0%
set score_threshold=2. - 41 rules added – Recall/Precision/F-score = 96.3%

It is easy to see that more is not necessarily better... Bringing the score threshold down to 1 made the algorithm add 196 rules as opposed to 8 with the default score threshold setting of 6. In addition to that the recall score actually went down, although marginally, from 96,1% to 96,0%. The middle way of a score threshold setting of 2 was the one that scored the best.

4. Change the accuracy threshold.
Finally I tried to experiment with the accuracy threshold changing it from the default value 0.5 to 0.7 and to 1.0
set accuracy_threshold=0.5. - 8 rules added – Recall/Precision/F-score = 96.1% (original)
set accuracy_threshold=0.7. - 8 rules added – Recall/Precision/F-score = 96.1%
set accuracy_threshold=1.0. - 6 rules added – Recall/Precision/F-score = 96.2%

Changing the accuracy threshold didn't seem to do much, so I tried the same change of settings (0.5, 0.7 and 1.0) with a different size of training data – 30.000 words, to see if the effect would be bigger: using data/wsj_30000
set accuracy_threshold=0.5. - 38 rules added – Recall/Precision/F-score = 96.9%
set accuracy_threshold=0.7. - 25 rules added – Recall/Precision/F-score = 96.8%
set accuracy_threshold=1.0. - 14 rules added – Recall/Precision/F-score = 96.4%

Summing up it seems that the biggest improvements are being made from increasing the training corpus. The other parameters do have an effect, but not nearly as big as I had expected. Trying to make a 'myBestScript', my guess is to have as big a training corpus as possible, keep my modified template, lower the score threshold somewhat, and keep the accuracy threshold moderate. I have made one like this:
set training_data='data/wsj_60000'.
set templates='templates/myTestTemplates2'.
set score_threshold=3.
set accuracy_threshold=0.5.
and got: 146 rules and Recall/Precision/F-score = 97.1%
Finally increasing score threshold to 4 and accuracy threshold 0.7 - I got 126 rules and 97.0 % precision. I'll stick with the 97.1 for part B

Part B

Given the best rule sequence so far, I tried to make a few manual changes to the rules using the html error analysis tool, write_html_error_data as a guide.
It computes error data and saves it in a HTML-based format in the file error_data.html.

Trying to find general rules that will remove several errors in one stroke, is not an easy task. Initially I found among other errors - 13 occurrences tagged as IN that should be RB. Five of these were:

89688:	925 million on sales of	about $ 7 billion in fiscal
90703:	Allied 's high-yield bonds was	**about** $ 50 million .
91693:	analysts say , United generates	**about** $ 1.1 billion in cash
92620:	loans , an increase from	**about** $ 3.4 billion of such
93009:	to buy Syncor 's for	**about** $ 81.8 million , but

The rule ‘ change IN to RB if the word is ‘about’ and the following word is a $-sign’ seemed to make sense.
I added a hand-written rule like:
tag:'IN'>'RB' <- wd:'about'@[0] & wd:'$'@[1] o

The new error_data.html contained - among other errors - 14 occurrences tagged as RB that should be IN: Among these two of the occurances were the reverse of what I had corrected with my new rule.

85815:	cash investment income to cover	**about** $ 1.4 billion of interest
88718:	William Collins PLC , for	**about** $ 1.3 billion to a

Grammar is definitely not my strongest side, and I am not able to see any difference between the two. is it a case of different persons coding differently, or what?

The special case of last paragraph turned out not to be so special after all.
Similarily: I found 15 occurrences tagged as VBD that should be VBN: among them:

86402:	Brazil has so far	**failed** to make a $ 1.6
86922:	Pantera 's also	**failed** to pay interest due Sept.

I added the rule:
tag:'VBD'>'VBN' <- wd:'failed'@[0] & wd:'to'@[1] o
and checked the ‘new and improved’ version with the new error_data.html. It contained:
24 occurrences tagged as VBN that should be VBD: among them

90238:	a bankruptcy-law filing since it	**failed** to make interest payments in
91602:	debt ; Resorts International Inc.	**failed** to meet interest payments on

A similar conflict between JJR and RBR. I added:
tag:'JJR'>'RBR' <- wd:'more'@[0] & wd:'than'@[1] o
with the conflicting new result of occurrences tagged as RBR that should be JJR:

Conclusions:

The manual corrections I've made to the rules has not made any great improvements to mankind. In my final test-run I came up to 97,3% correctly tagged, which is IMHO a fairly good improvement over the 96,1% from the test script.

A few questions comes to mind:

The conflicting new result I got from changing the rule set could be a result of conflicting human coding, but I see that (being ignorant to the finer aspects of grammar-tagging myself) it could also be a consistent human coding, and these kinds of difficulties could still occur. So it seems to be a general trend - that was probably an intended part of the exercise – which is also mentioned as one of the difficulties with the brill-tagger.

The strength that the Brill-tagger has, of being able to - in clear human-readable terms - point to the errors of the classifier being used, should be a very interesting supplement to stochastically-based classifiers, such as an HMM speech-recognition system for example. In such systems the correlations between the weights, are so complicated that it is almost impenetrable to the naked eye. It would be interesting to look further into.

All in all I found the exercise interesting, and the Brill algorithm, as well as the µ-TBL System good to get to know.