Natural Language Processing

Lab 5: Part-of-Speech Tagging

---

1 Introduction

In this assignment, we are going to train and evaluate a statistical part-of-speech tagger for English. We will make use of HunPoS, an open-source implementation of the standard HMM tagging model, which is installed on our Linux system.

2 Data

The data sets we are going to use come from the English Web Treebank.[1] The training set `ewt-train-wt.txt` contains sentences like the following, separated by blank lines:

```
But           CONJ
in            ADP
my            PRON
view          NOUN
it            PRON
is            VERB
highly        ADV
significant   ADJ
.             PUNCT
```

Each line contains a word and its part-of-speech, separated by a tab. The tag set consists of 17 coarse-grained categories taken from the Unviersal Dependencies project and listed in the following table.

Open class words		**Closed class words**		**Other**	
ADJ	adjective	ADP	preposition/postposition	PUNCT	punctuation
ADV	adverb	AUX	auxiliary verb	SYM	symbol
INTJ	interjection	CONJ	coordinating conjunction	X	unspecified
NOUN	noun	DET	determiner		
PROPN	proper noun	NUM	numeral		
VERB	verb	PART	particle		
		PRON	pronoun		
		SCONJ	subordinating conjunction		

The development set, which will be used to evaluate the tagger, comes in two versions:

- `ewt-dev-w.txt` contains only words and will be used as input to the tagger;
- `ewt-dev-wt.txt` contains both words and tags and will be used for comparison when evaluating the output of the tagger.

All the files needed for the assignment can be found in `/local/kurs/nlp/tagging/`.

3 Train a tagger

To train a tagger using `ewt-train-wt.txt`, you need to run a command like:

```
hunpos-train mytagger < ewt-train-wt.txt
```

The trained tagger will be stored in the file `mytagger` (or whatever name you choose for your tagger).

4 Run a tagger

Once you have trained a tagger, you can run it on the development set as follows:

```
hunpos-tag mytagger < ewt-dev-w.txt > ewt-dev-out.txt
```

The tagged output will be stored in the file `ewt-dev-out.txt`.

[1] The English Web Treebank (https://catalog.ldc.upenn.edu/LDC2012T13) contains syntactically annotated texts taken from sources such as weblogs, reviews, question-answers, newsgroups, and email. We use a version distributed by the Universal Dependencies project (http://universaldependencies.org).

5 Evaluate a tagger

To evaluate the output of the tagger, you run the evaluation script `score.py`:

```
python score.py tag ewt-dev-wt.txt ewt-dev-out.txt
```

In addition to reporting the overall *accuracy* (the percentage of tokens that were tagged correctly), the script prints a table showing the precision and recall for different part-of-speech tags. To give an example, say the precision for the NOUN tag is X%. This means that X% of all words that have been tagged as a noun are in fact nouns. Now say the recall is Y%. This means that Y% of all real nouns in the text have been correctly tagged as nouns by our tagger.

6 Use a shellscript

Shellscripts are used to run several command lines after each other in one single script. We introduce their usage here, because you will run the three commands that were introduced in the previous three sections, namely *train*, *run* and *score* a tagger several times in order to find an optimal setting for your tagger in the next exercise. Shellscripts allow you to better keep track of different parameter settings you used and the results these settings gave. In `/local/kurs/nlp/tagging/` we give you a shellscript named baseline_script.sh. Have a look at the script and try to understand what it does. Then run it using the following command: `sh baseline_script.sh`

7 Tune a tagger

The HunPoS implementation of HMM tagging has a few parameters that can be tuned for better tagging. In particular, the `-t` flag can be used to set the number of tags that are taken as context in the contextual probabilities. The default is `-t 2`, which means that we train a trigram model. See what happens if you instead use `-t 1` or `-t 3` when training the tagger. This means you copy the shellscript into a new file (e.g., using `cp baseline_script.sh t1_script.sh`), rename the experiment in the beginning of the shellscript (e.g., 't1' instead of 'baseline')[2] and add the parameter `-t 1` to the training command in the script:

```
hunpos-train $EXP_tagger -t 1 < ewt-train-wt.txt
```

For other parameters, consult the HunPoS user manual.[3] For each new parameter setting you try, remember to copy the shellscript and rename the experiment at the top of the script – otherwise old results will be overwritten.

[2] It is most helpful if you give the script file and the experiment within the script the same name and if that name correlated with the parameter setting.

[3] https://code.google.com/p/hunpos/wiki/UserManualI