Natural Language Processing - Assignment 4

Radhika Nikam
rn1334@nyu.edu

March 7, 2019

1 Problem Statement

The problem statement for the first programming assignment is to implement a Part-of-Speech (POS) Tagger using the Viterbi algorithm. We are provided with a copy of the *Wall Street Journal* corpus for this purpose, which is divided into three parts:

1. *WSJ_02-21.pos* - words and tags for training
2. *WSJ_24.words* - words for validation
3. *WSJ_24.pos* - words and tags for validation
4. *WSJ_23.words* - words for testing

A scoring program is also provided to calculate the scores on the validation/development corpus.

In a nutshell, the Viterbi algorithm implemented for this assignment assumes a Hidden Markov Model (HMM) while learning the emission and transition probabilities from the training corpus (while handling unknown words). The Viterbi decoder then serves to find the most likely path from start to end of POS tags for sentences in the development and test corpus.

2 Implementation

2.1 Method

A bigram HMM model was implemented with improvements to handle unknown words and suffixes. If words occurred more than once in the corpus, they were added to the vocabulary, else they were given special tokens along with occurrences of punctuations, capitals and digits.

Since open classes (nouns, verbs, adjectives, adverbs) tend to have more unknown words owing to flexible suffix-ization (see what I did there?), their suffixes can be handled and given special categories.[1,2,3] This is implemented in the submitted code, and leads to an improvement of close to 2% on the validation set. The suffixes used can be found in the file */src/constants.py* included with the submission.

These categories are used to resolve rare and unknown words, and preprocess the data set before it can be trained on. Sentence ends/starts are tagged 'START', and the blankspaces changed to 'NO_WORD' Subsequently, emission and transition probabilities for the corpus are calculated, and alpha smoothing is used to generate their probability distributions.

For decoding on the development corpus, the Viterbi algorithm [4] is implemented, as given for a bigram model.[5]

2.2 Running the code

The zipped file will contain the *Wall Street Journal* corpus in the folder **/WSJ_POS_CORPUS_FOR_STUDENTS**. Auxilary data generated during training is in **/data**, and **/src** contains the following files:

- *main.py* - creates the POS tagger
- *hmm.py* - defines util functions used during training and validation
- *viterbi.py* - the implementation of the Viterbi algorithm, heavily borrowed from it's wiki page.
- *constants.py* - directory, folders, global variables
- *scoring.py* - evaluation program for the trained models

The **.pos** files for validation and test are generated in the home folder itself.

For a system with Python3.6 (preferably with Anaconda) installed, you should be able to run the codes locally.

: To run the POS-Tagger, execute:

```
$ python main.py -d=../WSJ_POS_CORPUS_FOR_STUDENTS
```

where the argument **-d** takes only the corpus directory.

2.3 Evaluation

: To evaluate the generated .pos file:

```
$ python scoring.py
```

which will generate the scores for the validation .pos files.

3 Results

The results of the initial assessment (with uniform probability for unknown words) came to **92.33%**. Further trials (rounded to 2 significant digits) are shown:

`Trial 1` - Dealt with unknown/rare words using morphological features, smoothing factor = 1
accuracy: 93.36%

`Trial 2` - modified Trial 2 by using smoothing factor 0.5
accuracy: 94.22%

`Trial 3` - Used smoothing factor 0.25
accuracy: 94.69%

`Trial 4` - Smoothing factor 0.1
accuracy: 94.99%

`Trial 5` - Smoothing factor 0.01
accuracy: 95.27%

`Trial 6` - Smoothing factor 0.001
accuracy: 95.28%

`Trial 7` - Smoothing factor 0.0001
accuracy: 95.28%

`Trial 8` - Smoothing factor 0.005
accuracy: 95.30%

The best result on the validation set was obtained for $\alpha = 0.005$, with an accuracy of 95.30%. Since α is a redistribution parameter and the WSJ corpus is not small, lesser values of α seem to be preferred as it only marginally redistributes the probabities while taking care of unknown words.

The generated .pos file for the test set is included in the submission as well.

References

[1] Nordquist, Richard.(2019, February 5) *A List of 26 Common Suffixes in English.* Retrieved from https://www.thoughtco.com/common-suffixes-in-english-1692725

[2] *Suffixes from English Grammar Today*. Retrieved from https://dictionary.cambridge.org/us/grammar/british-grammar/word-formation/suffixes

[3] Delahunty, Gerald P., and Garvey, James J. (2010) *The English Language: From Sound to Sense. Perspectives on Writing*. Fort Collins, Colorado: The WAC Clearinghouse and Parlor Press. Available at https://innovationtest2.colostate.edu/books/perspectives/sound/

[4] Wikipedia contributors. (2019, February 15). *Viterbi algorithm*. In Wikipedia, The Free Encyclopedia. Retrieved 18:58, March 7, 2019, from https://en.wikipedia.org/w/index.php?title=Viterbi_algorithm&oldid=883459332

[5] Jurafsky, Daniel, and James H. Martin. (2009).*Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. 2nd edition. Prentice-Hall.