Assignment 3: POS Tagging

Sharon Hains
University of Alberta
Edmonton, Alberta, Canada
hains@ualberta.ca

Sharif Bakouny
University of Alberta
Edmonton, Alberta, Canada
albakoun@ualberta.ca

1 Part 1: Part of Speech Tagging

1.1 Stanford POS Tagger

After much trial and error, we were able to get Domain1 and Domain2 models created using the Stanford POS Tagger. The properties set for this tagger were based off of the 'english-bidirectional-distsim.tagger.props' file provided in the initial download of the Stanford POS Tagger. Although we were encouraged to adjust our default parameters to get an increased accuracy, we found that it was not necessary as the accuracy returned is much higher than any of our other types of models.

We did note that the usage of the Stanford POS took significantly longer for evaluation.

For the Stanford POS Tagger, its accuracies were:
Domain1Test trained on Domain1Train – 87.91%
Domain1Test trained on Domain2Train – 85.12%
Domain2Test trained on Domain1Train – 82.75%
Domain2Test trained on Domain2Train – 87.03%

1.2 HMMTagger

For the Hidden Markov Model (HMM), its accuracies were:
Domain1Test trained on Domain1Train – 26.71%
Domain1Test trained on Domain2Train – 18.07%
Domain2Test trained on Domain1Train – 25.57%
Domain2Test trained on Domain2Train – 35.72%

As we can see from the results above, the accuracies follow suit that Domain1 or Domain2 have a higher accuracy when being tested on their own corresponding models. We may note that the accuracies are not particularly high, as normally the accuracies of POS taggers used today are expected to be around 96-97% as we learned in class.

The lower inaccuracy may also be how the HMM tagger handles unknown words. As noted by Collins (2013), because the probability of unseen words is 0, the subsequent arg max for all possible paths calculated by the Viterbi algorithm will also be 0. So, because our training corpus is not that large, the likelihood is increased that our HMM tagger will encounter an unseen word in the test text, which will drive our accuracy downwards.

1.3 Brill Tagger

As we understand it, the Brill Tagger takes a set of templates and creates rules based on these templates based on the words around it. This would introduce the option of evaluation with bigrams, trigrams, etc (Godayal, 2018). Prior to our current code, we did test with trigrams and bigrams, but we found that the usage of what we understand to be the bigram Template gave us the best accuracy.

There is not a lot of clear documentation online as to how to properly set up the templates, or what was mentioned in the assignment about using a backoff tagger. Because of this, if there was a more suitable way to implement the Brill tagger, that is why we did not implement it.

We used the HMM Tagger for the initial tagger used for the Brill tagger. We also set a maximum of 10 rules to be used.

For the Brill Tagger, its accuracies were:
Domain1Test trained on Domain1Train – 51.54%
Domain1Test trained on Domain2Train – 47.21%
Domain2Test trained on Domain1Train – 52.77%
Domain2Test trained on Domain2Train – 60.16%

Again, as expected the test text had the highest accuracy when tested against its corresponding training file. Interestingly, we see that even though the Brill Tagger uses the HMM tagger, the Brill accuracies are significantly higher than the HMM tagger accuracies, where the Brill accuracy is almost double for all test cases.

The higher accuracy may be because of how the Brill method deals with unseen words. As in, this method would calculate the most likely path of a series of tags, for example NN -> VB.

2 Error Analysis

Below are the 5 most frequent errors we saw in the form of (Correct Tag, Incorrect Tag). These tags go from most frequent to least.

2.1 Stanford POS Tagger

Domain1Test trained on Domain1Train – ('VBD', 'VBN'), ('JJ', 'NN'), ('VBP', 'VB'), ('VB', 'NN'), ('MD', 'NN')
Domain1Test trained on Domain2Train – ('VBN', 'VBD'), ('VBP', 'VB'), ('NN', 'JJ'), ('NN', 'VB'), ('RB', 'JJ')
Domain2Test trained on Domain1Train – ('VBD', 'VBN'), ('NN', 'VB'), ('RB', 'IN'), ('JJ', 'NN'), ('RB', 'JJ')
Domain2Test trained on Domain2Train – ('VBN', 'VBD'), ('NN', 'VB'), ('RB', 'JJ'), ('RB', 'IN'), ('NN', 'VBD')

2.2 HMM Tagger

Domain1Test trained on Domain1Train – ('NN', 'JJ'), ('IN', 'JJ'), ('DT', 'JJ'), ('NNP', 'JJ'), ('NNS', 'JJ')
Domain1Test trained on Domain2Train – ('NN', 'NNP'), ('IN', 'NNP'), ('DT', 'NNP'), ('JJ', 'NNP'), ('NNS', 'NNP')
Domain2Test trained on Domain1Train – ('NN', 'JJ'), ('IN', 'JJ'), ('.', 'JJ'), ('VBD', 'JJ'), ('DT', 'JJ')
Domain2Test trained on Domain2Train – ('NN', 'NNP'), ('IN', 'NNP'), ('.', 'NNP'), ('VBD', 'NNP'), ('DT', 'NNP')

How does the HMM tagger approaches unknown words? As noted in Part 1 by Collins (2013), because the probability of unseen words is 0, the subsequent arg max for all possible paths calculated by the Viterbi algorithm will also be 0. So, because our training corpus is not that large, the likelihood is increased that our HMM tagger will encounter an unseen word in the test text and will be incorrectly tagged.

2.3 Brill Tagger

Domain1Test trained on Domain1Train – ('JJ', 'NN'), ('IN', 'NN'), ('NNS', 'NN'), ('NNP', 'NN'), ('RB', 'NN')
Domain1Test trained on Domain2Train – ('JJ', 'NN'), ('IN', 'NN'), ('NNS', 'NN'), ('NNP', 'NN'), ('RB', 'NN')
Domain2Test trained on Domain1Train – ('IN', 'NN'), ('VBD', 'NN'), ('PRP', 'NN'), ('RB', 'NN'), ('JJ', 'NN')
Domain2Test trained on Domain2Train – ('IN', 'NN'), ('VBD', 'NN'), ('RB', 'NN'), ('JJ', 'NN'), ('PRP', 'NN')

2.4 Summary

For the Stanford POS tagger, there was not one particular tag that stood out as the most frequently mistagged. There seemed to be an even combination between VBN, NN,VB, and VBD. Stanford POS Tagger uses context features to handle unknown words, such as previous and following words and other features.

For the HMM Tagger, the most frequently mistagged tags were JJ and NNP. We note that as per the class notes, one of the issues with the Brill training method is that the rules generated from the Templates can interact with each with each other, therefore creating the wrong tag. Brill Tagger handle unknown words by backing off to the initial tagger (HMM) in our case. As most unknown words in English tend to be nouns, most unknown words were tagged as NN.

These errors can be attributed to multiple reasons, one being that our training corpora are not that large, so types of tags that should be attributed to certain words are not being seen at point of training.

Another reason, in particular for the Stanford POS Tagger, is that it uses the Maximum Entropy Model to create our tagger. However, as stated in the class notes, this method of tagging does not include bidirectionality, so the meaning of one side of words can be lost because only the other side of the words are being used.

3 Learner English

As mentioned in Part 2, in the error analysis sections, are the 5 most frequent errors we saw in the form of (Correct Tag, Incorrect Tag). These tags go from most frequent to least.

For each type of model, the ELLTest.txt was tested on the Domain1 and Domain2 trained taggers.

3.1 Stanford POS Tagger

Accuracy:
Domain 1: 80.59%
Domain 2: 80.35%

Error Analysis:
Domain 1: ('VBP', 'VB'), ('VBD', 'VBN'), ('RB', 'JJ'), ('VB', 'NN'), ('IN', 'TO')
Domain 2: ('VBP', 'VB'), ('VBN', 'VBD'), ('RB', 'JJ'), ('PRP$', 'PRP'), ('IN', 'TO'), ('VB', 'NN')

3.2 HMM Tagger

Accuracy:
Domain 1: 34.09%
Domain 2: 29.74%

Error Analysis:
Domain 1: ('NN', 'JJ'), ('IN', 'JJ'), ('PRP', 'JJ'), ('DT', 'JJ'), ('.', 'JJ')
Domain 2: ('NN', 'NNP'), ('IN', 'NNP'), ('PRP', 'NNP'), ('DT', 'NNP'), ('.', 'NNP')

3.3 Brill Tagger

Accuracy:
Domain 1: 54.91%
Domain 2: 52.98%

Error Analysis:
Domain 1: ('IN', 'NN'), ('JJ', 'NN'): 344, ('PRP', 'NN'), ('RB', 'NN'), ('NNS', 'NN')
Domain 2: ('IN', 'NN'), ('JJ', 'NN'), ('PRP', 'NN'), ('RB', 'NN'), ('NNS', 'NN')

3.4 Taggers Trained on ELL

3.4.1 Stanford POS Tagger

Accuracy: 70.18%
Error Analysis: ('NN', 'PRP'), ('IN', 'PRP'), ('.', 'PRP'), ('DT', 'PRP'), ('JJ', 'PRP')

3.4.2 HMM Tagger

Accuracy: 70.18%
Error Analysis: ('NN', 'PRP'), ('IN', 'PRP'), ('.', 'PRP'), ('DT', 'PRP'), ('JJ', 'PRP')

3.4.3 Brill Tagger

Accuracy: 78.40%
Error Analysis: ('IN', 'NN'), ('JJ', 'NN'), ('PRP', 'NN'), ('RB', 'NN'), ('NNS', 'NN')

3.4.4 Summary

For the Domain 1 and Domain 2 taggers, we can see that the accuracy percentages are very similar to Part 1, where Stanford POS tagger had the highest accuracy, followed by Brill tagger and then HMM tagger.

For the error analysis, the errors presented for each model were very similar to Part 2, which makes sense because the same Domain files are being used against the ELL test file.

For the taggers trained on the ELL training file, we can see there is a significant jump in accuracy in comparison to the Domain 1 and 2 training files. This is in line with our expected results because the probabilities calculated for the ELL training file would be more in line with the ELL test file, than the Domain files.

Interestingly, we have noted that the Stanford POS tagger and the Brill tagger have remained very similar between Domain 1/2 and the ELL training file in terms of which tags were used incorrectly. This would indicate that the way the taggers are trained per model type then may influence which kind of errors show up.

Only the HMM tagger had quite different results between the Domain 1/2 trainer and ELL trainer, as the ELL trainer had used the tag PRP the most incorrect.

References: *@misc{mohit gupta_omg aspire to inspire before i expire_aspire to inspire before i expire_2019, title={NLP: Brill Tagger}, url={https://www.geeksforgeeks.org/nlp-brill-tagger/}, journal={GeeksforGeeks}, author={Mohit Gupta_OMG Aspire to Inspire before I expire and Aspire to Inspire before I expire}, year={2019}, month={Oct}}*

@misc{bushi_2015, title={Python Code to train a Hidden Markov Model, using NLTK}, url={https://gist.github.com/blumonkey/007955ec2f67119e0909}, journal={Gist}, publisher={GitHub}, author={Bushi, Samuel S}, year={2015}, month={Aug}}

@misc{5. data structures - python 3.8.0 documentation_2019, title={5. Data Structures¶}, url={https://docs.python.org/3/tutorial/datastructures.html}, journal={5. Data Structures - Python 3.8.0 documentation}, publisher={Python Software Foundation}, year={2019}, month={Oct}}

@misc{bushi_2015, title={Python Code to train a Hidden Markov Model, using NLTK}, url={https://gist.github.com/blumonkey/007955ec2f67119e0909}, journal={Gist}, publisher={GitHub}, author={Bushi, Samuel S}, year={2015}, month={Aug}}

@misc{nltk.tag package - nltk 3.4.5 documentation_2019, title={nltk.tag package¶}, url={https://www.nltk.org/api/nltk.tag.html#nltk.tag.brill.BrillTagger.tag}, journal={nltk.tag package - NLTK 3.4.5 documentation}, publisher={NLTK}, year={2019}, month={Aug}}

@misc{the stanford natural language processing group, title={The Stanford NLP Group}, url={https://nlp.stanford.edu/software/pos-tagger-faq.html}, journal={The Stanford Natural Language Processing Group}, publisher={Stanford}}

@misc{freecodecamp.org_2018, title={An introduction to part-of-speech tagging and the Hidden Markov Model}, url={https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/}, journal={freeCodeCamp.org}, publisher={freeCodeCamp.org}, author={Divya Godayal}, year={2018}, month={Mar}}

@misc{5. categorizing and tagging words_2019, title={5. Categorizing and Tagging Words}, url={https://www.nltk.org/book/ch05.html}, journal={5. Categorizing and Tagging Words}, publisher={NLTK}, year={2019}, month={Sep}}

@article{lamb_danso_2014, title={Developing an Automatic Part-of-Speech Tagger for Scottish Gaelic}, DOI={10.3115/v1/w14-4601}, journal={Proceedings of the First Celtic Language Technology Workshop}, author={Lamb, William and Danso, Samuel}, year={2014}}

@misc{collins, title={Chapter 2: Tagging Problems, and Hidden Markov Models}, url={http://www.cs.columbia.edu/~mcollins/hmms-spring2013.pdf}, journal={Chapter 2: Tagging Problems, and Hidden Markov Models}, publisher={Columbia University}, author={Collins, Michael}}