Assignment 3 Report

Marc-Andre Haley • Chu Duc Thang

<u>Tuning efforts</u>

HMM

Estimation function	Mean Accuracy
MLE	0.23
Laplace	0.73
ELE	0.76
Bell	**0.83**
Lidstone-Gamma = 0.1	0.79
Lidstone-Gamma = 0.5	0.76
Lidstone-Gamma = 1	0.73
Lidstone-Gamma = 0.01	0.79
Lidstone-Gamma = 0.02	0.79
Lidstone-Gamma = 0.04	0.79
Lidstone-Gamma = 0.06	0.79
Lidstone-Gamma = 0.08	0.79

We tuned the HMM tagger by using different estimators. For every estimator, we trained the model using that estimator and performed 5-fold cross validation using the training data. This gave us an estimation to accuracy for every estimator and reduces the randomness if we run the estimator only once. Then, we chose the estimator with the highest accuracy to train our HMM model and test its accuracy on the test data. We got the estimators from the NLTK website and the estimator function that performed the best was the Bell estimation (0.83). Moreover, MLE estimator has the worst performance (0.23). Laplace and ELE estimators have similar performance, around 0.73 and 0.76 respectively. Interestingly, Lidstone estimators with different Gamma values have almost same performance (around 0.79)

Brill

Template / max # rules	5	20	100	250
1 *see code	0.835	0.843	0.848	0.848
2: fntbl37()	0.836	0.846	0.857	**0.859**

3: nltkdemo18()	0.836	0.845	0.855	0.855
4: brill24()	0.836	0.844	0.856	0.857
5 *see code	0.836	0.845	0.855	0.855

The Brill tagger had more parameters that we could tune, namely templates, number of max-rules. The table above shows the mean accuracy scores after 5-fold cross validation using the training data on different combinations of parameters: templates and maximum number of transformations. Templates determine the rules that will be generated, and max_rules determines the maximum amount of transformations the tagger produces. There are 5 templates used for hyperparameter tuning. The first and last template are user-defined, while we also used fntbl37 (37 templates from fntbl distribution), nltkdemo18 (18 templates from original nltk demo), and brill24 (24 templates from seminal TBL Brill paper). As a general trend we can see that increasing the maximum number of rules slightly increases accuracy. Out of the 5 templates we tested, fntbl37() had the best mean accuracy (0.855). These results were obtained while using the nltk UnigramTagger (with simple Regex tagger as backoff) as the brill tagger trainer baseline. We also tested the same parameter combinations using only the simple Regex tagger as our baseline and obtained accuracy results about 10% lower. In the end our optimized brill tagger used the fntbl37() template, max_rules=250, and the UnigramTagger baseline.

Tagger performance

Accuracy of taggers with optimal parameters

	HMM	Brill
Accuracy	0.872	0.881
OOD Accuracy	0.813	0.860

After we select the best set of hyperparameter for HMM and Brill model, we retrain the model with this set on the whole training data and test on both out-of-domain and in-domain testset.

Tagger errors

In order to analyze the tagger errors, we plugged the test data and output data into a text comparison software. We did the same with the ood data.

Brill: The first common mistake the brill tagger made is mistaking VBN and VBD. This is because in most cases both versions of the word are written the same way and are distinguished by the words preceding the word itself. We think that because we used a unigram tagger as our baseline, the tagger couldn’t tell the difference between those verbs. The other mistake that appeared a lot was mistaking JJ for NN and vice versa. Most of the time this was on words that can be used as both and their tag depends on the context of the sentence. For example ‘light’ can be used as a noun when talking about an object that emits light, or as an adjective when describing something as light, i.e. not heavy. An error that was common in the non-ood results was classifying verbs as nouns. Most of the time, this was again for words that can have multiple tags depending on the context. For example ‘cook’ can be used as a verb or a noun.

HMM: The HMM tagger output seems to have less patterns amongst its errors than the brill tagger, not many errors are common accross the output. However one error that was common in the ood data, was classifying nouns as proper nouns. It happened with many nouns that started a sentence and as a result were capitalized. This makes sense because proper nouns are capitalized and the tagger could have learned this. However, it also happened to many nouns that were not capitalized, which can happen if our model has a bias towards predicting proper nouns by default. In the output, many of the words misclassified as NNP follow each other, so one explanation is that, if the tagger thinks one of the nouns is a proper noun, this increases the probability of the next word to be a proper noun. This makes sense because proper nouns do often follow another proper noun, e.g. first and last name, titles, company names etc. Similarly to the output from the brill tagger many verbs were classified as nouns in the non-ood results.

HMM vs Brill errors

We think the nature of most of the errors in the Brill tagger come from the tagger not using the context around the word. On the other hand, since HMM is a statistical model, it clearly uses the context around the word in order to predict the tag. This is made obvious by the fact that the same word, ‘Her’, was tagged differently throughout the text by the HMM tagger. However, the fact that HMM uses prior words doesn’t necessarily make it better and can create other kinds of errors like the proper noun error we discussed above. The differences between errors from both models can be explained by the difference in how both models work and learn tags: The HMM tagger is a generative tagger and the brill tagger is a transformational one. Both models performed similarly well, with the brill tagger having slightly higher accuracy. One thing we noticed is that the difference in performance between the ood and non-ood data was greater for the HMM model. This is to be expected because the HMM model learns the likelihood of tags matching certain words depending on the words that precede it. Therefore when we have data that is not in the same domain that was used to train the tagger, these likelihoods may not stay the same because context around words is likely to change. Since the brill tagger does not rely on context around words, testing with data out of domain does not have much of an effect.

After looking at the errors made by both taggers, it’s clear they have their strengths and weaknesses. Maybe combining both approaches into one tagger could make a more accurate tagger? For example, the HMM part of the tagger could calculate probabilities for certain tags and if none of them surpass a certain probability threshold the brill tagger could tag it according to its learned rules?