Statistical NLP 2017 - Assignment 3

Part-Of-Speech Tagging

Daniel Rivera Ruiz

2017-10-18

Part-Of-Speech (POS) Tagging refers to the process of assigning a syntactic category to each word in an input text. The task of POS tagging is a disambiguation one, since most words can be mapped to several tags depending on the context, like surrounding words or punctuation. In this assignment, a POS Tagger was developed based on three main components: a local trigram scorer modeled as a Hidden Markov Model (HMM), a Viterbi decoder, and a classifier for unknown words. This document briefly explains the design process of the POS Tagger and the results obtained.

1 Baseline POS Tagger

The baseline tagger as described in [1], is a minimalist implementation of a POS Tagger for the standard Wall Street Journal data from the Penn Treebank. This basic tagger assigns to each known word in the test set its most frequent observed tag during training, and assigns all unknown words the most frequent observed tag (over all) during training. The baseline tagger performs at 91.3% accuracy in general and 42.7% for unknown words for an in-domain dev set. These values will be used as reference for the rest of the results obtained throughout the assignment.

2 Hidden Markov Model (HMM) Scorer

The first improvement implemented over the baseline tagger was the replacement of the simple dummy tagger for a local tag trigram scorer based in a HMM. While the dummy tagger only considered the posterior log probability for each instance $logP(t_i|w_i)$ and ignored completely the context surrounding it, the HMM tagger introduces a trigram probability over the tags along with a prior probability, therefore assigning to each tag the score

$$s(t_i) = log(P(t_i|t_{i-1}, t_{i-2})P(w_i|t_i))$$

and maximizing over the sum of all tags. To implement this HMM scorer, it is important to smooth the trigram probability $P(t_i|t_{i-1}, t_{i-2})$ due to the sparsity of the data. This means that some tag trigrams that were not observed during training may appear in the dev or test sets, and without smoothing our model would assign zero probability to such cases.

To avoid this, the *deleted interpolation* algorithm as described in [2] is implemented. This algorithm defines the trigram probability as the following the linear interpolation:

$$P(t_i|t_{i-1}, t_{i-2}) = \lambda_3\hat{P}(t_i|t_{i-1}, t_{i-2}) + \lambda_2\hat{P}(t_i|t_{i-1}) + \lambda_1\hat{P}(t_i)$$

For this model to be probabilistically consistent, the λ_i must add up to 1, and they are selected as to maximize the likelihood of the whole corpus while successively deleting each trigram from the training set.

The HMM scorer improves the performance of the baseline model in about two points for the overall accuracy, scoring at 93.2%. The unknown words accuracy, however, remains unchanged at 42.7%. This is actually to be expected since the model to predict tags for unknown words has not yet been modified.

3 Viterbi Decoder

To further improve the performance of the model, a second-order Viterbi decoder was implemented. Instead of just traversing the states of the HMM model and selecting the highest scoring transition at each step (greedy decoder), the secod-order Viterbi decoder keeps track of the quantity

$$\pi(k, u, v) = \max_{w \in K}(\pi(k-1, w, u) * q(v||w, u) * e(x_k|v)) \tag{1}$$

which is a recursively defined score that represents the highest probability for any sequence of tags ending in the bigram (u, v), and for which the base case is $\pi(0, S, S) = 1$. For the notation in equation 1, which is taken from [3], the q probability is equivalent to the transitions (tag trigram probabilities) of the HMM model, and e corresponds to the emissions (prior probabilities) of the HMM model.

Aside from computing the π scores, the Viterbi algorithm must also keep track of the *pointers* to the states that generated such values, in order to traverse backwards the transitions of the trellis and retrieve the sequence of states that generated the highest scoring path.

After implementing the Viterbi decoder, the overall accuracy in the dev set gained another two points, for a final score of 95.3%. The unknown words accuracy boosted to 54.1%, which is still quite poor, but considering that the unknown words model remained unchanged, it can be considered an important improvement resulting from the Viterbi algorithm alone. The reason behind this improvement is that, even though unknown words are still being mapped to a probability distribution over the tags where the most frequent tag during training has a higher probability mass, this condition is no longer sufficient for the algorithm to select this tag. The Viterbi decoder, as explained above, will keep track of the scoring sequence up to two previous states, and therefore will not necessarily choose the highest-scoring tag for the unknown word in successive steps.

4 Unknown Words Classifier

The final task of the assignment was to come up with a better model to considerably improve the performance of the tagger for previously unseen words. So far, the tagger just assigned the same probability distribution to all unknown words i.e., the distribution of the tags over the training set. This was a very poor assumption to make, since the part-of-speech for a given word can be predicted with some level of confidence by looking at its morphology. The endings or suffixes of words can be particularly telling when it comes to part-of-speech categories. As it is exemplified in [4] words ending in *-able* are much more likely to be adjectives than nouns or any other category.

Based on this intuition and making use of the results from assignment 2, a MaxEnt classifier for unknown words was set up with the following parameters:

- All words that appeared 5 times or less during training were passed as examples to the MaxEnt classifier.
- The label set for the classifier coincides with the tag set of the words from the previous step, and each word is labeled with the tag most frequently observed during training of the POS Tagger.
- Bigram, trigram and 4-gram over the words characters were used as the features to train the classifier.

After training the classifier, the POS tagger was updated as follows:

- During training, all the infrequent words (the ones that were used to train the classifier) are substituted with their predicted labels.
- At test time, whenever an unknown word is encountered, it goes through the classifier and gets replaced by its label.

Implementing the improved classifier for unknown words, the overall accuracy of the POS tagger did not increase as substantially as with the other steps of the process, gaining only a half point from 95.3% to 95.8%. The accuracy for unknown words tagging did experiment a significant improvement, reaching a maximum value of 86.09% for a classifier with a smoothing parameter $\sigma = 0.3$.

The fact that the overall accuracy of the tagger was barely affected even though the accuracy for unknown words improved considerably, reveals that the tagger is still making some mistakes even for previously seen words, which can be explained by the fact that most common words in the English language are usually ambiguous, which means that they can be mapped to 2 or more part-of-speech tags depending on the context.

5 Results

To evaluate the performance of the POS tagger, two dev data sets were available: one within the domain of the training data and one out of domain. Table 1 shows the results obtained in both sets for the different steps of the design process.

Table 1: POS Taggers Comparison.

Scorer	Decoder	Classifier	In Domain		Out of Domain	
			Overall	Unknown	Overall	Unknown
Baseline			91.32%	42.74%	87.07%	53.31%
HMM Trigram	Greedy	Most Frequent	93.21%	42.74%	89.51%	53.31%
HMM Trigram	Viterbi	Most Frequent	95.30%	54.07%	90.51%	42.67%
HMM Trigram	Greedy	MaxEnt	93.98%	85.68%	90.60%	73.81%
HMM Trigram	Viterbi	MaxEnt	95.85%	86.09%	92.54%	74.07%

Additionally, the complete HMM POS tagger was evaluated using different values for the smoothing factor σ of the unknown words classifier. As it had been stated previously, the best accuracy for unknown words was achieved with a value of $\sigma = 0.3$, which can be observed in figure 1.

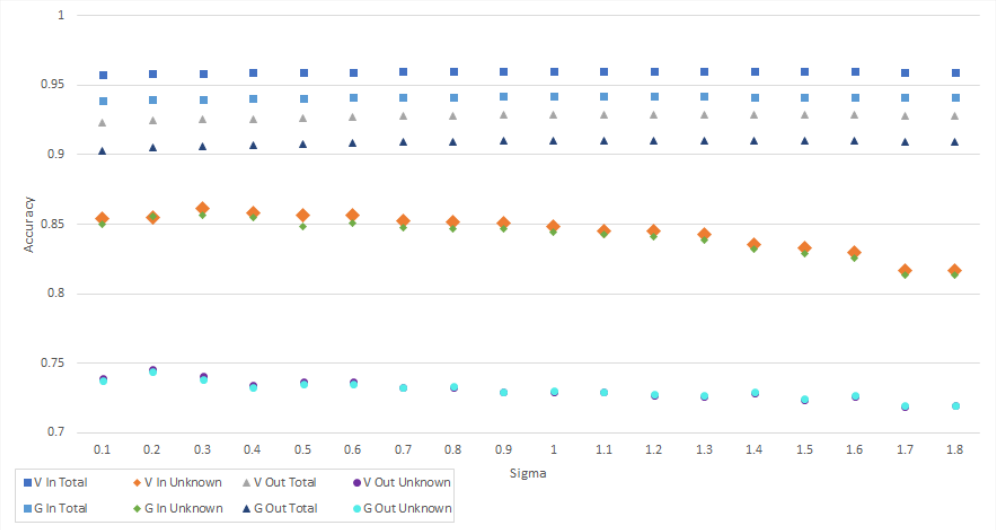

Figure 1: Accuracy for the HMM POS Tagger as σ varies.

Finally, a few observations on the model and how the results obtained could potentially be improved:

- The training process of the tagger was performed on sentences extracted from the WSJ alone. To improve its performance in other domains, sentences from a wider range of sources could be included.
- The accuracy of the tagger can also be further improved by introducing information about word capitalization into the model. As it is explained in [4], the probability distribution of tags for a given word can vary if the word is capitalized or not. This variation can be exploited to render the model more comprehensive.[1]
- Better results might also be obtained with more robust models, like higher-order HMMs or conditional random fields. These models, however, will also be more expensive to train, both time-wise and computational-wise.

[1] This condition was originally within the scope of the assignment but was not implemented due to time issues.

6 Conclusions

After having completed the assignment and analyzed the results obtained, the following conclusions can be drawn:

- Automated POS tagging is a challenging NLP problem that relies heavily in a thorough task of data preprocessing in order to label enough examples for model training.
- Ambiguous words can be particularly challenging for the tagger. This is a relevant issue that should be considered when designing the tagger because in a normal context these words will account for more than 50% of all the tokens.
- The fact that there will be unseen words during the training process must also be taken into account. Dealing with this situation will usually benefit from creativity and experience to come up with reliable models. Regardless of the solution implemented, it is of the utmost importance to maintain all components probabilistacally sound, otherwise the performance of the model will suffer drastically.

References

[1] A. Parikh, "Statistical NLP Assignment 3," 2017. Available: "http://www.cs.nyu.edu/~aparikh/assignment3-2017-revised.pdf".

[2] D. Jurafsky and J. Martin, *Speech and Language Processing*. Prentice-Hall, 2 ed., 2009.

[3] M. Collins, "Tagging with Hidden Markov Models," 2011. Available: "http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/hmms.pdf".

[4] T. Brants, "TnT: A Statistical Part-Of-Speech Tagger," 2000. Available: "http://www.coli.uni-saarland.de/~thorsten/publications/Brants-ANLP00.pdf".