Part-of-Speech Tagging isiXhosa with a Hidden Markov Model

1st Simon du Toit
Applied Mathematics Department
Stellenbosch University

***Abstract*—In this project an HMM (hidden Markov model) is used to perform part-of-speech tagging on the isiXhosa language. The HMM incorporates absolute discount smoothing to deal with unseen words and tag transitions. The Viterbi algorithm is used to tag a target corpus using the HMM parameters. The final tuned model achieves an accuracy of 79.1% on the test set.**

I. Introduction

I first describe how I process the text data and discuss the design of my HMM model and Viterbi algorithm. I then explain the implementation of these designs in detail. Finally, I present and discuss the results of tuning the model and applying it to the test set.

II. Design

In this section I describe how the data is processed and the theoretical design of the HMM (hidden Markov model), the chosen smoothing technique and the Viterbi algorithm I use for part-of-speech tagging.

A. Data Processing

All data is provided by the Resource Catalogue of the South African Centre for Digital Language Resources[1]. The data is structured as a table with a column for words and a column for the corresponding part-of-speech tags. The words are ordered into sentences, which are separated by empty rows. The tables are stored in two Excel files, a train set and a test set.

I read the training data into a table and replace the empty rows with ”Sentence Boundary” (SB) tags. In this implementation the SB tag simultaneously serves as the start-of-sentence and end-of-sentence tag. I also insert one row with an SB tag before the first row. The test data is processed similarly. I remove the first 20% of sentences from the training data to use as a validation set.

B. Hidden Markov Model

The part-of-speech tagger is implemented using an HMM. An HMM defines the probabilities of transitions between a sequence of states and their observed emissions or outputs[2]. Each transition probability is defined by $p(z_i|z_j)$ for states z_i and z_j, while each emission probability is expressed as $p(x|z_j)$ for an observation x.

In this problem the model's states are part-of-speech tags and its observed emissions are words. The transition probabilities are stored in a lookup table indexed by (tag, tag) pairs, while the emission probabilities are stored in a lookup table indexed by (word, tag) pairs. The probabilities are constructed by counting the occurrences of each (tag, tag) transition and each (word, tag) emission in the training set and normalizing. The SB tags are included in the transition probabilities but not the emission probabilities.

For the model to be able to deal with unseen words and tag transitions, a smoothing method must be used to assign non-zero probabilities to these events. I use absolute discount smoothing for the transition and emission probabilities. This involves removing a small mass $0 < d < 1$ from the count of each event seen in the training set, and dividing the total removed mass among the unseen events. Here d_{tr} and d_{em}, the respective discounts for the transition and emission probabilities, are hyperparameters.

C. Viterbi Tagging

The HMM can perform part-of-speech tagging by utilizing the Viterbi algorithm. The Viterbi algorithm is an instance of dynamic programming that is used to efficiently calculate the state sequence that is most likely to produce a given sequence of observations[2]. A target sentence is tagged by calculating the most likely sequence of tags to produce it, given the parameters of the HMM.

III. Implementation

In this section I explain how the designed components of the tagging model are implemented. All code is written in `Python` and the data is loaded using the `pandas` library.

A. Hidden Markov Model

The transition and emission probability tables are stored as dictionaries. Smoothing of the emission probabilities is performed as follows:

1) Store the counts of each observed (z_i, z_j) tag transition in the table. Also count the occurrence of each tag z_j in the training set.
2) Subtract d_{tr} from every table entry. Accumulate the subtracted d_{tr} for each z_j in $D_{tr}(z_j)$.
3) Divide each table entry and each $D_{tr}(z_j)$ by the count of the relevant z_j to normalize them.
4) For every unobserved (z_i, z_j) transition, add it to the table with probability $D_{tr}(z_j)$.
5) Count the number of unobserved transitions coming from each z_j, and divide the probability of each unobserved transition by the relevant count.

A similar process is used for smoothing the emission probabilities, but with (x, z_j) pairings and discount d_{em}. However, the model may encounter words which do not appear at all in the training vocabulary. Thus, smoothing of the emission probabilities is deferred until the model is applied to a target dataset. The trained model only stores the unnormalized emission counts. When applied to a target dataset, the combined vocabulary of the training set and target set is then used to smooth the emission probabilities before tagging is performed.

B. Viterbi Tagging

The Viterbi algorithm is implemented as a function which receives a sentence and computes the most likely part-of-speech tag sequence to produce that sentence, given the HMM parameters. The (smoothed) transition probabilities are stored in a transition matrix and the algorithm is partially vectorized to enable efficient computation. All probabilities are converted to log probabilities to prevent numerical underflow.

The model is applied to a target set by feeding each sentence into the Viterbi function and collecting the predicted tags in a list.

IV. Results and Insights

The hyperparameters d_{tr} and d_{em} are tuned manually on the validation set. The best configuration found is $d_{tr} = 0.5$ and $d_{em} = 0.1$, which results in a validation accuracy of 74.12%. The final model is applied to the test set and achieves an accuracy of 79.1%.

The model fairs better with a low emission discount d_{em}, indicating that unseen words in the validation and test sets do not have much influence on performance. This motivates the choice to use absolute discount smoothing, as it generally alters the training probabilities less than the simpler alternative of Laplace smoothing. This is however not true for the transition discount d_{tr}. The test accuracy is higher than the validation accuracy, which indicates that the model does not suffer from overfitting and generalizes well to the test set.

V. Conclusion

An HMM model with absolute discount smoothing is shown to be an effective tool for tagging isiXhosa text. Absolute discount smoothing is used to make the model robust to unseen tag transitions and observations. The discount hyperpamaters are tuned on a validation set and the final model achieves an accuracy of 79.1% on the test set.

References

[1] Martin Puttkammer, Martin Schlemmer, and Ruan Bekker. *CHLT isiXhosa Annotated Text Corpora*. North-West University. 2018. URL: https://repo.sadilar.org/handle/20.500.12185/309.

[2] Hao Tang. *Hidden Markov Models (Part 1)*. University of Edinburgh. 2021. URL: https://homepages.inf.ed.ac.uk/htang2/mini-asr/hmm/part1.html.