CS5800 Algorithms , Summer 2022, Northeastern University

Parts of Speech Tagging Using Dynamic Programming

Final Project Presentation

Instructor : Prof. Jonathan Mwaura

Team

Atharva Sapre

I had previously enrolled for NLP. The holistic nature of the course really helped me nurture my interest in this domain. Being currently enrolled for Algorithms, I was looking for a way to implement a greedy/dynamic approach to solve a real-world ML problem. After a fair amount of research, I found that the problem of POS tagging could be solved using a probabilistic and dynamic programming approach with the help of a Hidden Markov Model.

Anushka Patil

Speech Recognition and Machine Translation have always intrigued me and have been a few of the topics related to NLP that I have intended to work on. While reading about a few papers related to these topics I came across Parts of Speech Tagging which gives us an understanding of the structure of terms within corpus which can be used to make assumptions about the semantics. I thought of using one of the teachings from this course to solve a problem in my area of interest.

Sriram Hariharan

To me, the field of NLP has always been fascinating, since I started inclining towards ML and AI. This excitement and interest in learning about the process behind the screen pushed me to take up this project which aims at processing data, cleaning it, and applying algorithms to help the machine understand the text by deducing the POS represented by each word. Hence we dug deeper to understand a few solutions to find POS in text data and found that Viterbi uses a DP approach.

Pareekshit Reddy G

I came across parts of speech tagging while reading through articles on the Data science blog. It piqued my interest, and I decided to turn it into a project using my knowledge of algorithms. Annotating billion word document manually is impractical, so automatic tagging is used. Implementing this project with DP would improve my understanding of the subject and my confidence.

Table of Contents

Introduction

- A POS tag is a tag that indicates the part of speech for a word.
- Part-of-Speech tagging in itself may not be the solution to any particular NLP problem. It is a pre-requisite to simplify a lot of different problems.
- POS tagging has uses in the following NLP tasks:
  - Text to Speech Conversion : We need to know which word is being used in order to pronounce the text correctly.
  - Word Sense Disambiguation : Word-sense disambiguation is identifying which sense of a word (that is, which meaning) is used in a sentence, when the word has multiple meanings.
  - Semantic Analysis : POS tagging is a vital step in parsing and semantic analysis - enabling efficient parsing algorithms.

Problem Statement

- The most fundamental models in the field of Natural Language Processing (NLP) are based on Bag of Words. These models, however, fall short of capturing the syntactic relationships between words.
- Building lemmatizers, which are tools for breaking down words to their basic forms, need POS Tagging as well.
- POS tagging is the process of marking up a word in a corpus to a corresponding part of a speech tag, based on its context and definition.
- This task is not straightforward, as a particular word may have a different part of speech based on the context in which the word is used.
- Our aim is to use the probabilistic POS tagging model - Hidden Markov Model to compute the joint probability of a set of hidden states(POS tags of words in corpus) given a set of observed states.
- Then we use the Viterbi algorithm, which is a dynamic programming algorithm for obtaining the maximum a posteriori probability estimate of the most likely sequence of hidden states.

Dataset

- This project makes use of 2 datasets collected from Wall Street Journal (WSJ).
- One data set (WSJ-2_21.pos) will be used for training. The training set will be used to create the emission, transmission and tag counts.
- The other (WSJ-24.pos) for testing.
- The tagged training data has been preprocessed to form a vocabulary (hmm_vocab.txt).

Methodology

- First we create a Baseline model (bruteforce) for POS Tagging.
- We compute a few dictionaries that will help you to generate the tables which will help in predicting the parts of speech tag of each word. They are as follows:
  - Transition counts - This dictionary computes the number of times each tag happened next to another tag.
  - Emission counts - This dictionary will compute the probability of a word given its tag.
  - Tag counts - The dictionary will store tag as the key and number of times each tag appeared as the value.
- The Baseline Model works as follows :
- Choose the most prevalent POS for that word in the training set as the part of speech for that word.
- Check to see if the actual POS of the word is the same each time you anticipate using the word's most frequent POS. In that case, the prediction was accurate.
- By dividing the number of correctly predicted words by the total number of words for which we predicted the POS tag, we determine the accuracy.

Methodology

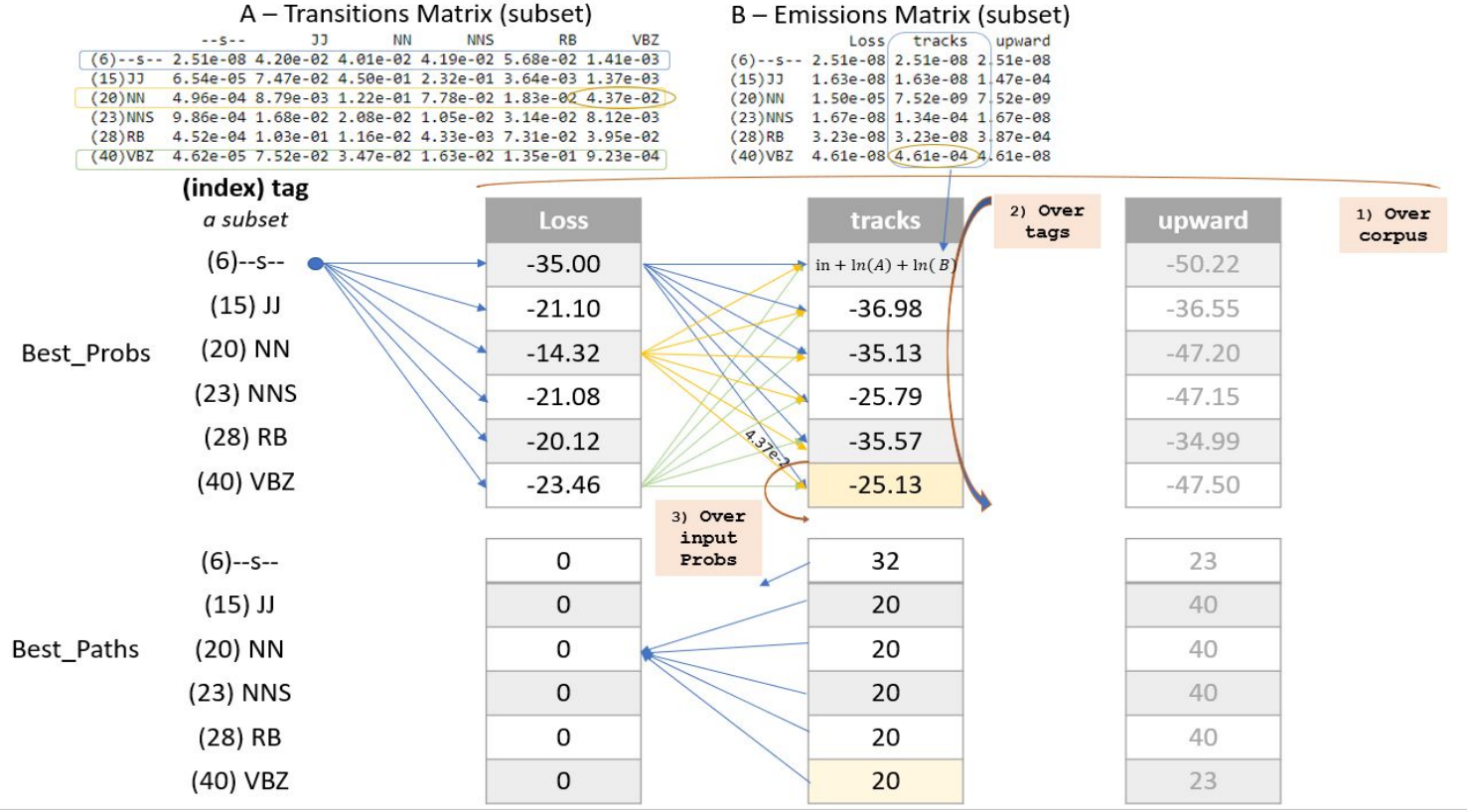
1) The next model we implemented was the Hidden Markov Model (HMM) using the viterbi algorithm.
2) The Markov Model contains a number of states and the probability of transition between those states. For this project the states are the parts-of-speech.
3) A Markov Model utilizes a transition matrix and an emission matrix.
4) We also generate 2 more matrices best_paths and best_probs using the above 2 matrices.
5) Best_probs and Best_paths matrices stores the probability of each path happening and the best paths up to that point.
6) HMM methodology:
    a) Generation of Emission matrix and transition matrix.
    b) Matrix initialization for best_paths and best_probs.
    c) Run the viterbi forward algorithm to fill the matrices.
    d) Run the viterbi backward algorithm to find the most probable sequence of POS tags.

Methodology

- After HMM, we implement the Viterbi algorithm which makes use of dynamic programming. We use transition and emission matrices from HMM to implement Viterbi.
- This is decomposed into 3 steps :
  - Initialization - In this part we initialize the best_paths and best_probabilities matrices that you we be populating in Viterbi feed_forward.
  - Viterbi Feed forward - Calculate the probability of each path happening and the best paths up to that point, at each step. For each word, compute a probability for each possible tag. Unlike the previous algorithm predict_pos (the 'warm-up' exercise), this will include the path up to that (word,tag) combination.
  - Viterbi Feed backward - The Viterbi backward algorithm gets the predictions of the POS tags for each word in the corpus using the best_paths and the best_probs matrices. Returns a list of predicted POS tags for each word in the corpus.
- By dividing the number of correctly predicted words by the total number of words for which we predicted the POS tag, we determine the accuracy.

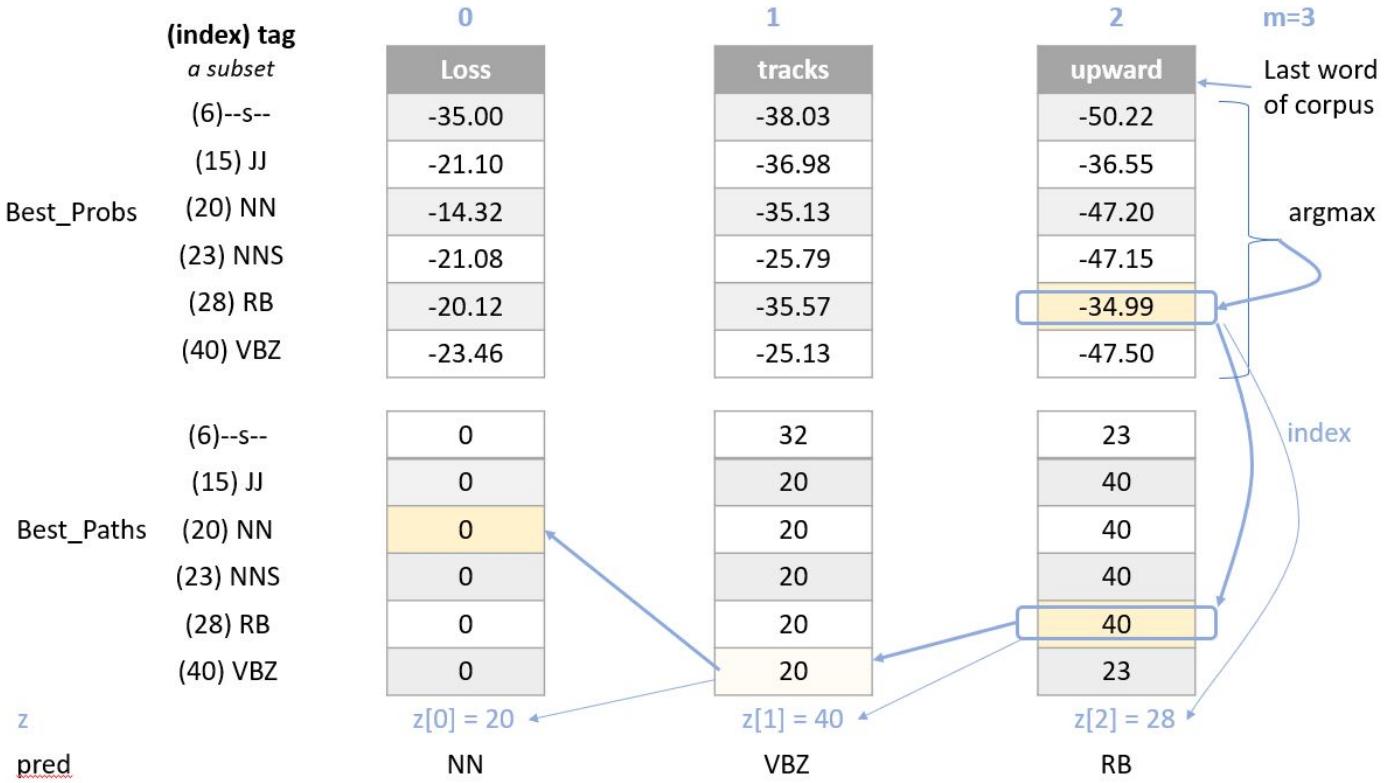
Methodology

Generation of Best paths and Best probabilities matrix (Viterbi Forward):



Methodology

Tracing the optimal path (Most probable sequence of POS tags)



Comparison

- The brute force (baseline) model for POS tagging generated an accuracy metric of 88.89%.
- The HMM Model with Viterbi algorithm using Dynamic Programming generated a prediction accuracy of 95.32% for the same dataset.
- By computing probabilities for each conceivable sequence of tags, the brute force approach to addressing this question entails selecting the sequence with the highest likelihood.
- However, All combinations of tags are possible. Hence, for a sentence with N words, and T tags the search space of possible state sequences X is $O(N^T)$.
- We will utilise the Viterbi method, a Dynamic Programming algorithm, to identify the most probable succession of hidden states that leads to a succession of observations, which requires $O(N^2*T)$.

Conclusion

- A large problem can be solved using the dynamic programming technique by being divided into a number of smaller, simpler subproblems, each of which is solved only once, and then the solutions are stored in a memory-based data structure.
- The Viterbi approach is definitely more appropriate for POS tagging. This is because the applications of POS tagging involve working with complex and huge datasets, hence it's important to think about the space and time complexity of the program to ensure that we get timely results.
- The complexity of Brute force method is $O(N^T)$, where as the complexity of HMM with Viterbi using DP is $O(N^2*T)$. Generally there are 46 or more tags involved. Hence, $O(N^T)$ is definitely significantly greater than $O(N^2*T)$.
- Hence, for POS tagging on larger datasets, and looking at the prediction accuracy generated by both the Brute force and HMM with Viterbi methods, we can conclude that HMM using Viterbi with Dynamic Programming is definitely more efficient that the Brute Force method for POS tagging.

Future Work

- We can use the Baum-Welch algorithm that utilizes EM to give full conditional likelihood to the hidden parameters resulting in much better accuracy at the cost of time.
- We can work on researching more about the different ways of achieving POS tagging. One of which is Transformation-based Learning (TBL).
- TBL is a rule-based algorithm for automatic tagging of POS to the given text. It allows us to have linguistic knowledge in a readable form, transforms one state to another state by using transformation rules.
- Complexity in tagging is reduced because in TBL we make use of of machine learned and human-generated rules. It is considered to be much faster than HMM tagger.
- Since, this is a sequence based modelling technique we can look into using RNNs and transformers to speed up the process of POS tagging and achieve higher predictive accuracy.

Thank You!