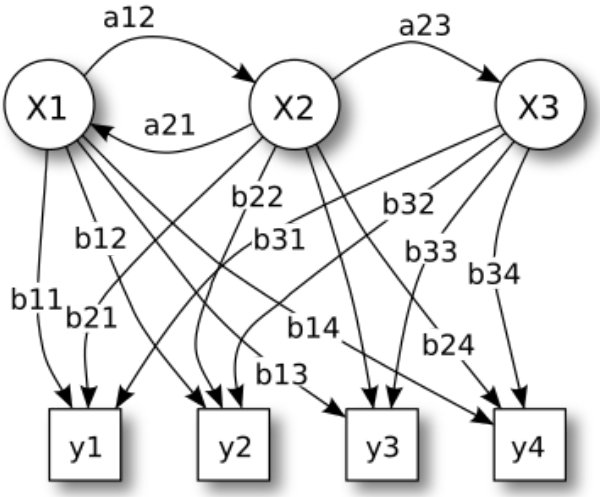
SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 OF THE UGC ACT, 1956)
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

ML PROJECT REPORT

PART OF SPEECH TAGGING USING HIDDEN MARKOV MODEL AND COMPARATIVE ANALYSIS



Chidambaranathan S

125018016, B. Tech. Computer Science and Business System

Table of Contents

Abstract

This project is about Part-of-Speech (POS) tagging using a Hidden Markov Model (HMM). The project used the BROWN_TE [4]I dataset, a variant of the Brown Corpus, which is in XML format. The data was parsed using the xml.etree.ElementTree library to extract emission and transition probabilities.

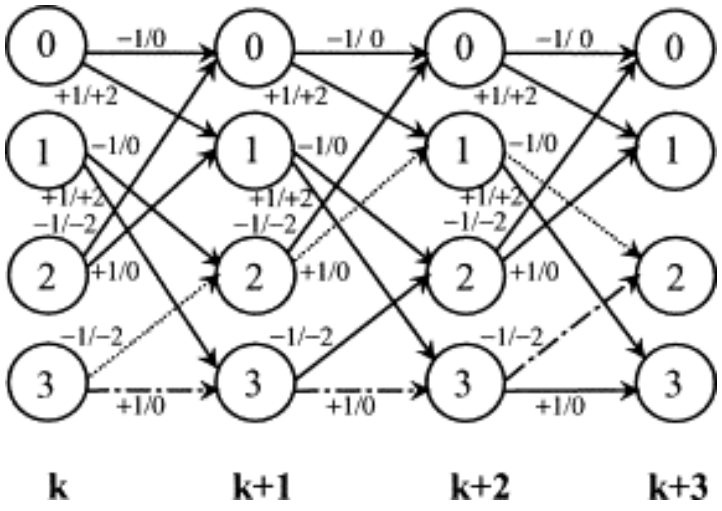
Three algorithms for POS tagging where implemented:

- greedy algorithm
- Viterbi dynamic programming algorithm
- An algorithm based solely on emission probabilities.

The project involved:

- Calculating emission and transition probabilities from the training data.
- Implementing the three POS tagging algorithms.
- Evaluating the accuracy of the algorithms on a test dataset.
- Visualizing the transition probabilities.

It was found that the Viterbi dynamic programming algorithm achieved the highest accuracy. The project provides a comprehensive analysis of different POS tagging methods and their performance on a real-world dataset.



Introduction

Dataset:

The project utilizes the "BROWN_TEI" [4] dataset, a variant of the Brown Corpus in XML format. This dataset is a collection of 500 samples of approximately 2,000 words each, totaling around one million annotated words.There are 84 tags in the dataset. This enables the training and evaluation of part-of-speech tagging models.

Task (T), Problem (P), Evaluation (E):

- **Task (T):** To create a part-of-speech tagger for the English language.
- **Problem (P):** The project focuses on accurately identifying the grammatical role (e.g., noun, verb, adjective) of each word in a sentence, considering the context.
- **Evaluation (E):** The effectiveness of the HMM model and its variants will be assessed based on their accuracy in predicting POS tags for words in unseen sentences. This will be done through:
    - Token-level accuracy, which is the percentage of correctly tagged individual words.
    - Sentence-level accuracy, reflecting the percentage of sentences where every word is tagged correctly

Implementation introduction

The project does not use any third-party libraries to implement HMM. Instead everything is written from scratch in python. “xml.etree.ElementTree” [3] is used to parse the brown corpus [4] and build the data structure of transition count and emission count. “matplotlib.pyplot” is used for data visualization mainly for the visualization of transition probabilities. Three models where implemented for part of speech tagging:

- **Greedy Algorithm**: A simplified approach that makes tagging decisions solely based on the most likely tag at each step.
- **Viterbi Dynamic Programming Algorithm**: A more sophisticated technique that utilizes dynamic programming to consider all possible tag sequences and select the most probable one.
- **Emission Probability-Based Tagging**: This method assigns tags based only on the emission probabilities (the likelihood of a word given a tag) without considering tag transitions

Results are discussed later in detailed manner, in brief we can say that:
emission based tagging >> greedy algorithm >> Viterbi Dynamic programming algorithm

Related work

Hidden Markov Models (HMMs) and the Viterbi algorithm have been widely used in various fields such as speech recognition, natural language processing (NLP), bioinformatics, and more. The foundational work on HMMs dates back to the 1970s, with Leonard E. Baum and colleagues formalizing the mathematical framework of HMMs. Since then, HMMs have been employed to model temporal and sequence-based data due to their ability to handle hidden states and probabilistic transitions between these states.

One of the earliest impactful applications of HMMs was in speech recognition, particularly through the work of Rabiner in the 1980s [1]. His tutorial on HMMs provided a detailed explanation of the forward-backward algorithm, parameter estimation using the Baum-Welch algorithm, and the Viterbi algorithm for decoding the most likely sequence of hidden states. This foundation laid the groundwork for applying HMMs in automatic speech recognition (ASR) systems.

In addition to speech recognition, One of the notable early works that employed the Viterbi algorithm for POS tagging is Kupiec's (1992) [2] study. In *Robust Part-of-Speech Tagging*, Kupiec introduced a robust statistical method for POS tagging using HMMs, demonstrated improvements in accuracy. Kupiec's work was pivotal in demonstrating that HMMs, combined with the Viterbi algorithm, could effectively handle noisy or incomplete data in natural language processing (NLP) tasks (Kupiec, 1992). This laid the groundwork for many subsequent advancements in POS tagging and other sequence prediction problems.

Recent advancements have seen the combination of HMMs with deep learning techniques, particularly in hybrid models that enhance sequence prediction tasks. For example, in hybrid architectures like Deep Hidden Markov Models (DeepHMMs), deep neural networks are used to represent emission probabilities, while the Viterbi algorithm remains central to decoding. These models have been applied to complex sequence data such as in NLP and video-based activity recognition.

Despite these advancements, many modern applications have shifted towards using Long Short-Term Memory (LSTM) networks and other deep learning-based approaches, especially in areas like NLP and time series prediction. However, HMMs retain their value in applications where interpretability, efficiency, and explicit state modeling are required.

In my work, I implement HMM from scratch, focusing on the Viterbi algorithm for decoding the most probable sequence of hidden states. While previous studies rely heavily on pre-built libraries and frameworks, this project emphasizes an in-depth understanding of HMMs by constructing the model and the decoding algorithm from first principles.

Methodology

This project aims to code up a Hidden Markov Model for part-of-speech tagging and compare the performance of different algorithms.The experiment involves training and testing three different algorithms on the BROWN_TEI dataset. The algorithms include:

- Viterbi greedy algorithm
- Viterbi dynamic programming algorithm
- An algorithm that only uses emission probabilities.

The specific programming language and environment used for this project are not mentioned in the sources. However, the code uses the following Python libraries:

- **xml.etree.ElementTree[3]**: This library is used to parse the content of the BROWN_TEI dataset, which is in XML format.
- **NumPy**: This library provides support for numerical operations and array manipulation, likely used for handling matrices like emission and transition matrices.
- **random**: Used for shuffling the sentences in the dataset.
- **copy**: Used for creating deep copies of dictionaries.
- **math**: Used for mathematical operations, such as calculating ceilings and logarithms.
- **matplotlib.pyplot**: Used for creating visualizations, potentially for analyzing the transition probabilities between different parts of speech.

The code is located in a file named **project.ipynb.**

Dataset Size, Feature Size, Results of Data Preprocessing

- The **BROWN_TEI [4]** dataset contains **500 samples**.
- The dataset contains a total of **1,081,150 words** .
- After preprocessing, the dataset has **54,496 unique words** and **84 unique part-of-speech tags**.
- The dataset is divided into a training set with **42,149 sentences** and a test set with **10,537 sentences**.
- **Emission and transition probabilities** are calculated from the training data.
- The corpus was divided into an **80:20 ratio**, with 80% used for training the model and 20% for testing

Results

This project implemented a Hidden Markov Model (HMM) for part-of-speech (POS) tagging on the Brown Corpus dataset [4].

The project achieved a token level accuracy of 91.546% using a greedy algorithm and 94.403% using the Viterbi algorithm.

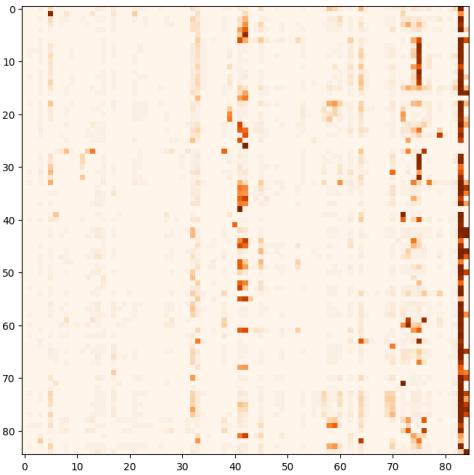
Sentence level accuracy was 29.429% using the greedy algorithm and 41.159% using the Viterbi algorithm.
Sentence level accuracy here refers to all the tokens being correctly tagged by the total sentences.

Token level accuracy based only on emission probabilities was 87.195%.

In the testing set 5,767 were new words where found out of the total 215,544 words. Which means 2.67% of the words are new and we don't have its emission probability.

```
token level acc (greedy algo): 91.54604164346955%
sentence level acc (greedy algo):  29.429628926639463%
token level acc (based only on emission):87.19518984522881
token level accuracy (viterbi algorithm) :94.40346286605055
new words:5767
total words:215544
sentence level accuracy (viterbi algorithm): 41.159722881275506
```

The results suggest that the **Viterbi algorithm** is **more effective** than the **greedy algorithm** for POS tagging.



this image of visualizing transition probabilities is just to fill page

Possible extensions

Implement a text tagger. A text tagger adds word equivalence classes, where the most common words are given their own tags. This can help to improve the accuracy of the model.

Test higher-order context. The current model only uses a **bigram context**, meaning that it only considers the previous tag when predicting the current tag. Testing higher-order contexts, such as **trigram or four-gram contexts**, could improve the **model's accuracy.**

Trying out advanced models. We can use deep learning models which are tested for part of speech tagging. Certain LSTM and deep learning models can perform very well for part of speech tagging.

Learning outcome

Link to the **project.ipynb** file (brown_tei/corpus.xml has to be **imported manually**)

https://colab.research.google.com/drive/1ItvTDjQjGB6oiNmFDFmGysIDp3U9XKSu?usp=sharing

Link to the **project (**all the files are available**)**

https://github.com/chidam333/pos

Skills used and things learned:

- Ability to program complex dynamic programming algorithms. Given it's description
- Learn about evaluation metrics used to compare models in machine learning
- Extensive use of python and libraries mentioned in this document.
- Understanding the internals and implementation of Hidden markov models.
- The first principle idea behind data smoothing.
- The reason behind the usage of log likelihood instead of likelihood.

Failed Experiments:

As generally python lists/numpy lists are used in data processing steps. I tried to convert the transition count and emission count hashmaps into lists. Which failed badly due to improper labeling. Improper labeling leads to improper indexing which in turn leads to wrong results.

Conclusion

The project successfully implemented a Hidden Markov Model (HMM) for part-of-speech (POS) tagging using the Brown Corpus dataset [4]. The implementation involved calculating transition and emission probabilities based on the frequencies of tags and words in the training data.The Viterbi algorithm, a dynamic programming approach, was utilized to efficiently determine the most likely sequence of POS tags for a given sentence.
The project demonstrated the effectiveness of HMMs for POS tagging and the importance of the Viterbi algorithm for achieving higher accuracy.
The problem statement was parsed to find the task and has been implemented and evaluated.

Certain limitations of the project :

- The functions built are not modular and can't be used easily with other data.
- There might be unexpected edge cases which break the code.
- It uses a very old model for a problem which has much more modern and better solutions.

Advantages:

- Good for understanding the internals of HMM.
- Easy to change code and experiment with the parameters and techniques.
- Uses legible and easy syntax which is easy to understand.

References

[1] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286. https://doi.org/10.1109/5.18626
[2] Kupiec, J. (1992). Robust part-of-speech tagging. In Proceedings of the First International Conference on Computational Linguistics (COLING), pp. 172-175. https://doi.org/10.1016/0885-2308(92)90019-Z.
[3] xml.etree.ElementTree — The ElementTree XML API https://docs.python.org/3/library/xml.etree.elementtree.htmlhttps://docs.python.org/3/library/xml.etree.elementtree.html
[4] *Brown Corpus (TEI XML Version)*, id: brown_tei; size: 8737738; author: W. N. Francis and H. Kucera; copyright: ; license: May be used for non-commercial purposes. https://www.nltk.org/nltk_data/