ELL884 NLP Assignment 1

Name(s): Pulkit Singal Entry number(s): 2023AIB2064

Part-of-Speech Tagging

Pulkit Singal

Abstract / Short summary

This work focuses on performing Part-of-Speech Tagging using Hidden Markov Model (HMM) and Maximum Entropy Markov Model (MEMM) frameworks. The models are trained on the given training dataset, and then the Viterbi algorithm is used to find the tags corresponding to the different tokens in the untagged sentences of the test dataset.

Introduction

Part-of-Speech tagging is the process of assigning a part-of-speech to each word in tagging a text. The input is a sequence w_1, w_2, ….. w_T of (tokenized) words and a tagset, and the output is a sequence t_1, t_2, ….. t_T of tags, each output t_i corresponding exactly to one input w_i.

HMM Tagger

Part-of-Speech tagging can be carried out using the Viterbi algorithm for HMM decoding, which is the process of determining the sequence of hidden variables (tags) corresponding to the sequence of observations (tokens). For part-of-speech tagging, the goal of HMM decoding is to choose the tag sequence that is most probable given the observation sequence of T words.

Building and Training of the Model

- The training data is read from the train.csv file.
- All the tokens in the training data are then converted into their lowercase version.
- The test data is read from the test_small.csv file.
- All the tokens in training data are stored in ascending order in list distinct_words.
- All the tags in training data are stored in ascending order in list distinct_tags.
- The following probability matrices are created using MLE estimates (N = total no. of tags, |V| = size of vocabulary):

1) pi $_{N \times 1}$ (prior probability matrix)
   = Probability that a tag (t_i) appears at the start of the sentence
   = P (t_i / start)
   = count (start, t_i) / Total no. of sentences in the training dataset
2) A $_{N \times N}$ (transition probability matrix)
   = Probability of a tag (t_i) occurring given the previous tag (t_{i-1})
   = P (t_i / t_{i-1})
   = count (t_{i-1}, t_i) / count (t_{i-1})
3) B $_{N \times |V|}$ (emission probability matrix)
   = Probability that given a tag (t_i), it will be associated with a given token (w_i)
   = P (w_i / t_i)
   = count (t_i, w_i) / count (t_i)

This completes the training of the HMM Tagger.

Testing of the Model

For each untagged sentence (consisting of T tokens) belonging to the test dataset:

- A viterbi $_{N \times T}$ (path probability matrix) and backpointer $_{N \times T}$ matrix consisting of all zero elements is initialized.
- In the first column of the viterbi matrix (corresponding to the first token in the untagged sentence), the sth element is set as pi[s] x B[s][token_index], where token_index is the position of the first token (lowercase) in list distinct_words.
- In the tth (except first) column of the viterbi matrix (corresponding to the tth token in the untagged sentence), the sth element is set as the maximum (over s_prime, where s_prime is the row index of the (t-1)th column) of viterbi[s_prime][t-1] x

Name(s): Pulkit Singal Entry number(s): 2023AIB2064

A[s_prime][s] x B[s][token_index], where token_index is the position of the tth token (lowercase) in list distinct_words.

- In the tth (except first) column of the backpointer matrix (corresponding to the tth token in the untagged sentence), the sth element is set as the argmax (over s_prime, where s_prime is row the index of the (t-1)th column) of viterbi[s_prime][t-1] x A[s_prime][s] x B[s][token_index], where token_index is the position of the tth token (lowercase) in list distinct_words.
- Variable bestpathpointer is set as the row index of the largest element in the last column of the viterbi matrix (corresponding to the last token of the sentence).
- Vector bestpath consists of the path obtained by following pointers back in the backpointer matrix, from the bestpathpointer row in the final column to retrieve the desired set of labels. This gives the most probable sequence of tags for the sequence of tokens in the untagged sentence.

Once all the untagged sentences are processed, the tokens of each sentence along with their respective tags are written to file submission.csv

Handling of out of vocabulary (OOV) words

Out of vocabulary (OOV) words are those tokens which are present in the test dataset but not in the training dataset. Any new untagged sentence in the test dataset is first checked for OOV words before it is processed. In case OOV words are present in the sentence, their position in the untagged sentence is stored in a list. Then only the procedure of creating the viterbi matrix is modified as below:

- If the first token is OOV word, then the sth element in the first column of the viterbi matrix is set as pi[s].
- If the tth (except first) token is OOV word, then the sth element in the tth column of the viterbi matrix is set as the maximum (over s_prime, where s_prime is the row index of the (t-1)th column) of viterbi[s_prime][t-1] x A[s_prime][s].

MEMM Tagger

Maximum Entropy Markov Model or MEMM is an augmentation of the basic Maximum Entropy classifier so that it can be applied to assign a class to each element in a sequence, just like HMMs. Whereas the HMM model includes distinct probability estimates for each transition and observation, the MEMM gives one probability estimate per hidden state, which is the probability of the next tag given the previous tag and the observation. Like HMM, the MEMM uses the Viterbi algorithm to perform the task of decoding.

Building and Training of the Model

- The training data is read from the train.csv file.
- All the tokens in the training data are then converted into their lowercase version.
- The test data is read from the test_small.csv file.
- All the tokens in training data are stored in ascending order in list distinct_words.
- All the tags in training data are stored in ascending order in list distinct_tags.
- The following probability matrices are created using MLE estimates (N = total no. of tags, |V| = size of vocabulary):

1) $C0_{|V| \times N}$
   = Probability that a tag (t_i) appears at the start of the sentence, and that it will be associated with a given token (w_i)
   = P (t_i / start, w_i)
   = count (start, t_i, w_i) / count (start, w_i)
2) $C_{N \times N \times |V|}$
   = Probability of a tag (t_i) occurring given the previous tag (t_{i-1}), and that it will be associated with a given token (w_i)
   = P (t_i / t_{i-1}, w_i)
   = count (t_{i-1}, t_i, w_i) / count (t_{i-1}, w_i)

This completes the training of the MEMM Tagger.

Name(s): Pulkit Singal Entry number(s): 2023AIB2064

Testing of the Model

For each untagged sentence (consisting of T tokens) belonging to the test dataset:

- A viterbi $_{N \times T}$ (path probability matrix) and backpointer $_{N \times T}$ matrix consisting of all zero elements is initialized.
- In the first column of the viterbi matrix (corresponding to the first token in the untagged sentence), the sth element is set as C0[token_index][s], where token_index is the position of the first token (lowercase) in list distinct_words.
- In the tth (except first) column of the viterbi matrix (corresponding to the tth token in the untagged sentence), the sth element is set as the maximum (over s_prime, where s_prime is the row index of the (t-1)th column) of viterbi[s_prime][t-1] x C[s_prime][s][token_index], where token_index is the position of the tth token (lowercase) in list distinct_words.
- In the tth (except first) column of the backpointer matrix (corresponding to the tth token in the untagged sentence), the sth element is set as the argmax (over s_prime, where s_prime is row the index of the (t-1)th column) of viterbi[s_prime][t-1] x C[s_prime][s][token_index], where token_index is the position of the tth token (lowercase) in list distinct_words.
- Variable bestpathpointer is set as the row index of the largest element in the last column of the viterbi matrix (corresponding to the last token of the sentence).
- Vector bestpath consists of the path obtained by following pointers back in the backpointer matrix, from the bestpathpointer row in the final column to retrieve the desired set of labels. This gives the most probable sequence of tags for the sequence of tokens in the untagged sentence.

Once all the untagged sentences are processed, the tokens of each sentence along with their respective tags are written to file submission.csv

Handling of out of vocabulary (OOV) words

Out of vocabulary (OOV) words are those tokens which are present in the test dataset but not in the training dataset. Any new untagged sentence in the test dataset is first checked for OOV words before it is processed. In case OOV words are present in the sentence, their position in the untagged sentence is stored in a list. Then only the procedure of creating the viterbi matrix is modified as below:

- If the first token is OOV word, then all the elements in the first column of the viterbi matrix are set as 1.
- If the tth (except first) token is OOV word, then the sth element in the tth column of the viterbi matrix is set as the maximum (over s_prime, where s_prime is the row index of the (t-1)th column) of viterbi[s_prime][t-1].

Results

Model	Kaggle Score
HMM Tagger	0.796
MEMM Tagger	0.748

Kaggle Notebook Links

HMM Tagger:
https://www.kaggle.com/code/pulkitsingal/hmm-tagger/edit/run/162419181

MEMM Tagger:
https://www.kaggle.com/code/pulkitsingal/memm-tagger/edit/run/162419229

References

1) Speech and Language Processing, Dan Jurafsky and James H. Martin
2) Slides and Notes by Prof. Tanmoy Chakraborty