CSCI 544: HW 2

Assignment Report

Task 1: Vocabulary Creation

What threshold value did you choose for identifying unknown words for re-placement?
What is the overall size of your vocabulary?
How many times does the special token "<unk>" occur following the replacement process?

Selected THRESHOLD = **2**
Vocabulary Size = **15568**
Special Token "<unk>" occurrences = **28581**

Approach

- Data Preparation
  - Read the JSON data as a pandas dataframe.
  - Convert the sentences to lowercasing.
- Vocabulary Generation
  - Find unique words in the sentences from the train dataset.
  - Calculated the frequency for every unique word.
  - Replaced words with frequency of less than or equal to THRESHOLD with <unk> token.
  - Grouped by words and aggregate by sum to collect all the <unk> token.
  - Split the dataset into two: one with special token and other with remaining words.
  - Concatenated both dataframes keeping the special token at top.
  - Added an index column.
  - Saved the vocabulary as txt file as per format requirements given by the task.

Task 2: Model Learning

How many transition and emission parameters in your HMM?

Number of Transition Parameters = **2025**
Number of Emission Parameters = **700560**

Approach

- Data Preprocessing
  - Extracted the unique POS tags from the train dataset.
  - For ease of access combined sequences and labels in a list having a sequence of tuples
    - First element: word
    - Second element: POS tag
- HMM Learning
  - Calculating the transition, emission, and prior probabilities
  - We initial the HMM with params and create 3 matrices prefilled with zeros.
    - Find some constants from training data and given list of POS tags
      - N = Number of states i.e. number of distinct tags
      - M = Number of observable symbols i.e. number of distinct words
    - State transition probability matrix of size N * N
    - Observation Emission probability matrix of size N * M
    - Prior probability matrix of size N * 1
  - For consistency in finding the index associated with tags and words we create 2 mapping dictionaries.
  - Computing the prior probabilities
    - Loop over each sentence in training data
      - Extract the first tag and its index from every sentence.

                - Calculate the occurrence of that tag.
            - Divide the occurrences of each tag by total number of sentences in the train data.
    - Computing the transition probability matrix
        - Iterate over sentences.
            - Extract label indices.
            - Iterate over label indices.
                - Get the indices of previous and current label.
                - Increment the corresponding entry of the transition matrix.
            - Normalize the transition matrix.
            - Apply Laplace smoothing (or add a small constant) to handle cases where the probability is zero.
    - Computing the emission probability matrix
        - Iterate over sentences.
            - Iterate over words and labels in the sentence.
                - Get the indices of word and tag.
                - Increment the corresponding entry of the emission matrix.
            - Normalize the emission matrix.
            - Apply Laplace smoothing (or add a small constant) to handle cases where the probability is zero.
    - Save the model parameters to a JSON file after formatting the matrices.

Task 3: Greedy Decoding with HMM

What is the accuracy on the dev data?

Accuracy = **0.9155**

Approach

- Setup dictionaries to map.
    - tags to their respective indices
    - words to their indices in vocabulary
- Precomputes scores for each word-tag pair by multiplying the prior probabilities with emission probabilities.
- Decoding sentences
    - Iterate over list of sentences.
        - Iterate over each word in the sentence.
        - For the first word, it calculates the scores based on priors and emissions.
        - For subsequent words, it combines the transition probabilities from the previous state with the emission probabilities for the current word.
        - Predicted tag for the current word is Tag with the highest combined score.
        - Store selected tag for next iteration

Task 4: Viterbi Decoding with HMM

What is the accuracy on the dev data?

Accuracy = **0.9323**

Approach

- Setup dictionaries to map.
    - tags to their respective indices
    - words to their indices in vocabulary
- Precomputes scores for each word-tag pair by multiplying the prior probabilities with emission probabilities.
- Initialize variables.
    - Viterbi matrix to store the score at every time step (size = number of sentences * num of states)
    - Path matrix to store the best path (same size as Viterbi matrix)
- Iterate over list of sentences.

- Compute the score for the first word using precomputed prior emission matrix.
- Then loop over time steps, starting from the second word in the sentence.
  - At each time step, calculate scores based on the previous Viterbi values, transition probabilities, and emission probabilities.
  - Update V and path based on the maximum score.
- Backtracking to find the best sequence of tags.
  - Starts from last word and iterate backwards, selecting tag with the highest score at each step.

Calculate Accuracy

- For every true and predicted sequences doing a tag-by-tag comparison.
- Calculated only for dev data.

Inferences

- After the model is fit on training data, the inferences are made using the model and both decoders are applied on the test dataset.
- The inferences obtained are formatted as per the given requirements and stored as JSON files.