CSE 4392 Special Topic: Natural Language Processing

Homework 7 - Spring 2024

Due Date: Mar 26th, 2024, 11:59 p.m. Central Standard Time

Welcome to this week's assignment focusing on sequence modeling techniques, specifically Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs). In this assignment, we delve into the realm of sequential data analysis, an essential aspect of natural language processing and various other domains. You will explore the application of HMMs and CRFs for sequence labeling tasks, such as part-of-speech (POS) tagging, where the goal is to assign the most likely sequence of labels to input sequences. By completing this assignment, you will gain practical experience in preprocessing sequential data, extracting relevant features, designing and training HMM and CRF models, and evaluating their performance using appropriate metrics. Understanding these techniques is vital for tackling real-world problems involving sequential data, making this assignment an invaluable opportunity to deepen your knowledge and skills in sequence modeling.

Problem 1 - 50%

For this task, you are provided with a very simple dataset consisting of labeled sequences of words, where each word is associated with a specific part-of-speech (POS) tag(ignore the syntax). Your goal is to implement a Hidden Markov Model (HMM) and a Conditional Random Fields (CRFs) **MANUALLY** for POS tagging. We assume that we are only four POS tags: NN, VB, JJ, and RB, and the vocabulary is restricted to the words that appear in the dataset. A simple dataset is shown below:

- Training Data:
  - "cats/NN jump/VB high/JJ" (Noun-Verb-Adjective)
  - "apples/NN grow/VB red/JJ" (Noun-Verb-Adjective)
  - "cats/NN run/VB quickly/RB" (Noun-Verb-Adverb)
  - "red/JJ dogs/NN quickly/RB grow/VB" (Adjective-Noun-Adverb-Verb)
- Test Data:
  - "cats/NN quickly/RB jump/VB" (Noun-Adverb-Verb)
  - "dogs/NN grow/VB high/JJ" (Noun-Verb-Adjective)

Hidden Markov Model (HMM) - 20%

Please calculate the following probability distributions for the HMM model based on the training dataset:

- Transition Probabilities: $P(t_i|t_{i-1})$
- Emission Probabilities: $P(w_i|t_i)$
- Initial Probabilities: $P(t_1)$

Conditional Random Fields (CRFs) - 30%

The CRF does not compute a probability for each tag at each time step. Instead, at each time step the CRF computes log-linear functions over a set of relevant features, and these local features are aggregated and normalized to produce a global probability for the whole sequence. In another word, the CRF map entire sequence of states S and observations O to a **global** feature vector. The probability of a sequence of states S given observations O is defined as:

$$P(S|O) = \frac{exp(\mathbf{w} \cdot \mathbf{f}(S, O))}{\sum_{S'} exp(\mathbf{w} \cdot \mathbf{f}(S', O))},$$

where $\mathbf{w}$ is the weight vector, and $\mathbf{f}(S, O)$ is the feature vector for the sequence S and observations O.

We can assume that the length of the weight vector and feature sequence is m, the formula can be rewritten as:

$$P(S|O) = \frac{exp(\sum_{m}^{k=1} w_k \cdot F_k(S, O))}{\sum_{S'} exp(\sum_{m}^{k=1} w_k \cdot F_k(S', O))},$$

where $F_k(S, O)$ is the k-th feature function for the sequence S and observations O. The **global** feature function $F_k(S, O)$ can be computed as a combination of **local** features:

$$F_k = \sum_{i=1}^{n} f_k(s_{i-1}, s_i, O, i),$$

where n is the length of the sequence, O is the observation sequence, i is the index of the sequence, s_i is the state at index i, and $si - 1$ is the state at index $i - 1$. Since here we are talking about linear chain CRFs, the feature function $f_k(s_{i-1}, s_i, O, i)$ is just related to the current state and the previous state. So that we can use Dynamic Programming to calculate the value of feature function like how we calculate HMM.

For simplicity, we will assume all CRF features take on the value 1 or 0. Below, we explicitly use the notation $\mathbb{1}\{x\}$ to mean "1 if x is true, and 0 otherwise". For feature designing, we can use the following feature functions:

- Word features: $f_k(s_{i-1}, s_i, O, i) = \mathbb{1}\{o_i = w_m\}$
- Previous POS tag features: $f_k(s_{i-1}, s_i, O, i) = \mathbb{1}\{s_{i-1} = t_k\}$
- Transition features: $f_k(s_{i-1}, s_i, O, i) = \mathbb{1}\{s_{i-1} = t_k \wedge s_i = t_l\}$

After we calculate the feature function, we can use the gradient descent algorithm to update the weight vector $\mathbf{w}$. But we won't do it manually here, in order to update the weight vector, we need to now the gradient of the loss function with respect to the weight vector. The simple loss function we can use here is the negative log likelihood function:

$$-\log P(S|O) = -\sum_{k=1}^{m}\sum_{n}^{i=1} w_k f_k(s_{i-1}, s_i, O, i) + \log \sum_{s'_1,\ldots,s'_n} \exp(\sum_{k=1}^{m}\sum_{n}^{i=1} w_k f_k(s'_{i-1}, s'_i, O, i))$$

And the gradient of the loss function with respect to the weight vector $\frac{-\partial \log P(S|O)}{\partial w_k}$ can be done efficiently using Dynamic Programming.

In this task, you are required to use the feature functions above to calculate the feature vector, and then set the weight vector $\mathbf{w}$ to be all 1s, calculate the $\hat{Y}$ of two test sentence, which is formularized as:

$$\begin{aligned}
\hat{Y} &= \arg\max_{Y \in y} P(Y|X) && (1)\\
&= \arg\max_{Y \in y} \frac{1}{Z(X)} \exp(\sum_{k=1}^{K} w_k F_k(X, Y)) && (2)\\
&= \arg\max_{Y \in y} \exp(\sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)) && (3)\\
&= \arg\max_{Y \in y} \sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i) && (4)\\
&= \arg\max_{Y \in y} \sum_{k=1}^{K}\sum_{i=1}^{n} w_k f_k(y_{i-1}, y_i, X, i) && (5)
\end{aligned}$$

Problem 2 - 50%

In this task, you are provided The English Penn Treebank (PTB) corpus with tagging, and in particular the section of the corpus corresponding to the articles of Wall Street Journal (WSJ), is one of the most known and used corpus for the evaluation of models for sequence labelling.

- Training Data: WSJ-2_21.pos
- Test Data: WSJ-24.pos

Your task is to implement a Hidden Markov Model (HMM) and estimate the transition and emission probabilities from the training data. And then implement the Viterbi forward-backward algorithm in Python to infer the POS tags for the sentences in the test data. Please compute your accuracy in the end.

Hint: Please go through the dataset first and use the Viterbi algorithm to calculate the possbilites.

Attach your codes and report.