CSCI544: Homework Assignment №2

Due on Sept 26, 2023 (before class)

Introduction

This assignment gives you hands-on experience on using HMMs on part-of-speech tagging. We will use the Wall Street Journal section of the Penn Treebank to build an HMM model for part-of-speech tagging. In the folder named *data*, there are three files: *train*, *dev* and *test*. In the files of *train* and *dev*, we provide you with the sentences with human-annotated part-of-speech tags. In the file of *test*, we provide only the raw sentences that you need to predict the part-of-speech tags.

File Formats

The *train.json*, *dev.json*, *test.json* are in the json format and can be read with

```python
import json

with open('train.json') as f:
    train_data = json.load(f)
```

JSON Schema in *train.json* and *dev.json*:
In the list of sentence, labels pairs, each "record" has an index, sentence (words), labels (tags)

```json
[
    {
        "index": 0,
        "sentence": ["This", "is", "a", "sample", "sentence."],
        "labels": ["label1", "label2", "label3", "label4", "label5"]
    },
    {
        "index": 1,
        "sentence": ["Another", "example", "sentence."],
        "labels": ["label1", "label2", "label3"]
    },
    {
        "index": 2,
        "sentence": ["Yet", "another", "sentence", "for", "analysis."],
        "labels": ["label1", "label2", "label3", "label4", "label5"]
    },
    // More records...
]
```

JSON Schema in *test.json*:
It is similar to the schema in *train.json* and *dev.json*, but we don't give you the labels.

```
[
    {
        "index": 0,
        "sentence": ["This", "is", "a", "sample", "sentence."]
    },
    {
        "index": 1,
        "sentence": ["Another", "example", "sentence."]
    },
    {
        "index": 2,
        "sentence": ["Yet", "another", "sentence", "for", "analysis."]
    },
    // More records...
]
```

Task 1: Vocabulary Creation (20 points)

The first task is to create a vocabulary using the training data. In HMM, one important problem when creating the vocabulary is to handle *unknown* words. One simple solution is to replace rare words whose occurrences are less than a threshold (e.g. 3) with a special token '< unk >'.

Task. Generate a vocabulary from the training data stored in the "train" file and save this vocabulary as "vocab.txt." The format of the "vocab.txt" file should adhere to the following specifications: Each line should consist of a word type, its index within the vocabulary, and its frequency of occurrence, with these elements separated by the tab symbol (\t). The initial line should feature the special token "< unk >," followed by subsequent lines sorted in descending order of occurrence frequency.

Please take into account that you are only allowed to use the training data to construct this vocabulary, and you must refrain from using the development and test data.

For instance:

$< unk >$	0	2000
$word_1$	1	20000
$word_2$	2	10000

Additionally, kindly provide answers to the following questions:
What threshold value did you choose for identifying unknown words for replacement?
What is the overall size of your vocabulary, and how many times does the special token "< unk >" occur following the replacement process?

Task 2: Model Learning (20 points)

The second task is to learn an HMM from the training data. Remember that the solution of the emission and transition parameters in HMM are in the following formulation:

$$\begin{aligned} t(s'|s) &= \frac{\text{count}(s \to s')}{\text{count}(s)} \\ e(x|s) &= \frac{\text{count}(s \to x)}{\text{count}(s)} \\ \pi(s) &= \frac{\text{count}(null \to s)}{\text{count}(num\_sentences)} \end{aligned}$$

$t(\cdot|\cdot)$ is the transition parameter.
$e(\cdot|\cdot)$ is the emission parameter.
$\pi(\cdot)$ is the initial state (The sentence begins with this state); also called as prior probabilities.

Task. Learning a model using the training data in the file *train* and output the learned model into a model file in json format, named *hmm.json*. The model file should contains two dictionaries for the emission and transition parameters, respectively. The first dictionary, named *transition*, contains items with pairs of (s, s') as key and $t(s'|s)$ as value. The second dictionary, named *emission*, contains items with pairs of (s, x) as key and $e(x|s)$ as value.

Additionally, kindly provide answers to the following questions:
How many transition and emission parameters in your HMM?

Task 3: Greedy Decoding with HMM (30 points)

The third task is to implement the greedy decoding algorithm with HMM.

Task. Implementing the greedy decoding algorithm and evaluate it on the development data. Predict the part-of-speech tags of the sentences in the test data and output the predictions in a file named *greedy.json*, in the same format of training data.

Additionally, kindly provide answers to the following questions:
What is the accuracy on the dev data?

Moreover, you are encouraged to utilize the subsequent algorithm as a point of reference.

Algorithm 1 Greedy Decoding for a sentence

```
Input:
  hmm with
    1. $\pi(s_i)$ as an element in initial state vector $\in \mathbb{R}^{|S|}$
    2. $t(s_i|s_j)$ as an element in transition state matrix $\in \mathbb{R}^{|S|\times|S|}$
    3. $e(w_i|s_i)$ as an element in emission matrix $\in \mathbb{R}^{|W|\times|S|}$
  where $S$ is a set of all tags and $W$ is a set of all words.
  sentence $\leftarrow \{w_1, w_2, ...., w_T\}$
Output: $\{y_1, y_2, ...., y_T\}$ (A list of tags)

  function DECODE($\{w_1, w_2, ...., w_T\}$)
    $y_1 \leftarrow \underset{s\in S}{argmax}\ \pi(s) * e(w_1|s)$

    for $i \leftarrow 2$ to $T$ do
      $y_i \leftarrow \underset{s\in S}{argmax}\ t(s|y_{i-1}) * e(w_i|s)$
    end for

    return $\{y_1, y_2, ...., y_T\}$
  end function
```

Task 4: Viterbi Decoding with HMM (30 Points)

The fourth task is to implement the viterbi decoding algorithm with HMM.

Task. Implementing the viterbi decoding algorithm and evaluate it on the development data. Predicting the part-of-speech tags of the sentences in the test data and output the predictions in a file named *viterbi.json*, in the same format of training data.

Additionally, kindly provide answers to the following questions:
What is the accuracy on the dev data?

For Viterbi Algorithm, adopt the algorithm on wikipedia for our use case:
`https://en.wikipedia.org/wiki/Viterbi_algorithm`
Please note that the meaning of x and y are interchanged in this url.

Submission

Please follow the instructions and submit a zipped folder containing:

1. A txt file named *vocab.txt*, containing the vocabulary created on the training data. The format of the vocabulary file is that each line contains a word type, its index and its occurrences, separated by the *tab* symbol '\t'. (see task 1).
   For example,

$< unk >$	0	2000
$word_1$	1	20000
$word_2$	2	10000

2. A json file named *hmm.json*, containing the emission and transition probabilities (see task 2).

3. Two prediction files named *greedy.json* and *viterbi.json*, containing the predictions of your model on the test data with the greedy and viterbi decoding algorithms.

4. python code and a README file to describe how to run your code to produce your prediction files. (see task 3 and task 4).

5. A PDF file which contains answers to the questions in the assignment along with brief explanations about your solution.