Assignment 4*

CMSC 473/673 — Introduction to Natural Language Processing

Due Monday November 26th, 2018, 11:59 AM

<table>
<thead>
<tr>
<th>Item</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigned</td>
<td>Wednesday November 7th, 2018</td>
</tr>
<tr>
<td>Due</td>
<td>Monday November 26th, 2018</td>
</tr>
<tr>
<td>Topic</td>
<td>Hidden Markov Models</td>
</tr>
<tr>
<td>Points</td>
<td>105</td>
</tr>
</tbody>
</table>

In this assignment you will understand and gain experience in both implementing and using Hidden Markov Models (HMM), with a special focus on part of speech tagging across multiple languages.

As with the previous assignments, you are to complete this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or public Piazza discussions).

The following table gives the overall point breakdown for this assignment.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
</tr>
</tbody>
</table>

**What To Turn In** Turn in a writeup in PDF format that answer the questions; turn in all requested code necessary to replicate your results. Be sure to include specific instructions on how to build (compile) your code. Answers to the following questions should be long-form. Provide any necessary analyses and discussion of your results.

**How To Submit** Submit the assignment on the submission site:

[https://www.csee.umbc.edu/courses/undergraduate/473/f18/submit](https://www.csee.umbc.edu/courses/undergraduate/473/f18/submit)

Be sure to select “Assignment 4.”

---

*Thanks to Chi Zhang, who helped write this assignment.*
1. **(25 points)** In this question you’ll be able to explore part of speech tags in more depth. There is script that’s already written that you can use to help in this question:


   This script finds occurrences of words (or parts of speech) within a left-right context window. You may use it, but you do not have to if you find it easier to answer this question through other means. See Appendix A for more on its usage.

   For this question, unless otherwise specified, answer using the training set for UD_English-EWT. You can reference and read about the UD part of speech tags (http://universaldependencies.org/u/pos/); these are the common, or UPOS, part of speech tags. The language-specific tags, or XPOS tags, will be separately defined for each language (UD_English-EWT follows the Penn Treebank set: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html).

   (a) How many different UD and language-specific part of speech tag types are defined?
   
   (b) What are the ten most common part-of-speech tuples centered around the word “to”, using one word to the left and one word to the right? Answer this for both UD tags and the corpus-specific tags. Record your observations, and hypothesize what linguistic patterns you may be observing. (Your exact results may change from run to run, as ties among equal count tuples are broken arbitrarily. With the provided script, use --store pos --word to.)

   (c) Recall that prepositions include words such as “from,” “for,” and “with.” Under both the UPOS and XPOS tags, what are all of the ways that the word “to” is defined? Discuss the differences in these tags; you may want to look through examples to help ground your discussion. (For example, if you notice that the FOO tag is used for “to,” then you can look at phrases around a FOO-based “to” with the flags --word to --pos FOO --left 1 --right 4.)

   (d) Let’s look at the XPOS-based (language-specific) RP tag. First, find a word $\omega$ that is tagged with both the RP tag and some other tag $X$. Compare, contrast and summarize how $\omega$ as an RP is used vs. as an $X$. This word can be tagged with other parts of speech as well, but you only must focus on RP vs. $X$.

   (e) Now pick one other language (call it L). Compare the distributions of UPOS and XPOS in L to that of UD_English-EWT (one effective way is to produce and compare histograms). Is the overall UPOS distribution similar? Is L’s XPOS set larger, smaller, or roughly the same size of the English one? What could be one effect of this?

   (f) Finally, repeat part (b) on L and compare what you find to what you found for part (b).


   In addition to discussing the basic methodology and findings of this paper, identify findings you found interesting, surprising, or confusing. What is the overall takeaway (for you) from this paper?

   The full Bibtex citation is

---

1 Single spaced, regular font, and one column is fine.
3. (50 points) In this question, you will implement a **supervised** HMM Part-of-speech tagger using Viterbi decoding. Use the training data to train, the development data to perform internal comparisons, and the test data to perform a final evaluation.

- **Your Task** Your task is to perform well-controlled examinations of the following models (described below) on part of speech tagging. Specifically,
  - Select two UD languages, both of which have non-trivial XPOS tagsets.
  - For each language and POS tagset pair, document internal progress and development by training on the training set and evaluating per-token accuracy on the development set.
  - When internal development is done, pick the best configuration for each model class (you should have four of these best configurations for HMMs, and four for the baselines).
  - Once that is done, evaluate these best-in-class models on the testing set on both per-token and per-sentence accuracy.
  - Present the results, both internal and final, in readable formats, e.g., in tables or plots.
  - For the HMM models only, identify, discuss, and compare the 5 POS tags that are most commonly identified correctly.
  - For the HMM models only, identify, discuss, and compare the 5 POS tags that are most commonly identified incorrectly.
  - Describe, and justify, any design decisions or assumptions you made along the way.

Turn in all code, including what is necessary for producing the plots. Clearly indicate in the writeup what model and model configurations resulted in the best test set performance.

- **How You Will Be Graded** Your grade on this problem will be from proper documentation of your code, **our ability to run and test/validate your code** (efficiently, in a reasonable time), experiments, and reporting of results on both the development and test portions.

- **System Specifics**
  - **Models** Implement the following taggers:
    1. A baseline POS tagger. One straightforward option is to predict each word’s tag independently of the others through class-based unigram models (e.g., an extension of the “3 coins”). With this baseline, given a sentence $w_1 w_2 \ldots w_N$, pick the POS $z_i$ for each word token $w_i$ as $z_i = \text{arg max}_k p(z_i = k | w_i) = \text{arg max}_k p(w_i | z_i = k) p(z_i = k)$.
    2. A supervised HMM.
  - **Data** Use two languages: UD.English-EWT and some other language $L$ of your choice. For each language, train and evaluate separate HMMs for the UPOS and XPOS tagsets.
**Implementation** Define the word vocabulary from the training set; note, your vocabulary could exclude words from the training set. How you smooth (either the transition or the emission probabilities) or define the vocabulary is up to you. Implement the models using log probabilities, and include beginning and ending tags and observations on each sentence.

**Predicting** Prediction for the baseline model should be done according to the baseline model (for the suggested baseline, this would be predicting each token’s tag separately). For the HMM, use Viterbi decoding.

**Evaluation** For dev, evaluate on per-token **accuracy** (how many tokens were tagged correctly). For test, evaluate on per-token and per-sentence accuracy (how many sentences were tagged completely correctly). Per-token accuracy (vs. $F_1$) is a legacy convention for POS tagging. You can imagine, for the final test evaluation, filling out the following table:

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-token Baseline</td>
<td>UPPOS</td>
<td>XPOS</td>
</tr>
<tr>
<td>HMM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per-sentence Baseline</td>
<td>UPPOS</td>
<td>XPOS</td>
</tr>
<tr>
<td>HMM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix A  Counting Script

This script is meant to aide analysis of part of speech tagged data. There are a number of flags that control its behavior. Run `python lexicalized_spans_ud.py --help` to see all the options.

The two most critical to filtering the input data are `--word <s>` and `--pos <s>`. Each take a string argument (e.g., “cat,” or “NOUN”) and only accumulate statistics for that word and/or part of speech. E.g., if `--word cat --pos NOUN` only considers spans where the word “cat” was tagged as a “NOUN,” while `--word` considers spans where there’s the word “cat” (for any part of speech), and providing neither will consider any otherwise relevant span.

The flag `--store [word,pos,both]` allows counts to be acquired for just words, just POS tags, or pairs of words with their respective POS tags. This allows sequence counts of words, POS tags, or both to be easily acquired.

The flag `--pos-type` can take one of two choices: either UPOS (for common, universal POS tags) and XPOS (for language specific tags).

The flags `--left <d1> --right <d2>` adjust the context by $d_1$ words to the left of a particular word and $d_2$ words to the right. If both are 0, no left or right context is considered and returns a list of unigrams, ranked by overall count. You can return the entire ranked list by setting `--most-common` to `-1` (default is 10).

For example, if you want to find the two most common tuples involving any particular word to the left and right of a progressive verb (VBG), use the command

```
$ python lexicalized_spans_ud.py train.conllu --pos VBG --store both --most-common 2 --left 1 --right 1
```

1345 unique tuples matching * word, VBG POS
34 times: ,/ according/TO to/TO
8 times: <BOS>/<BSP> According/TO to/TO

If you only want to store the words, set the `--store` flag to word

```
$ python lexicalized_spans_ud.py train.conllu --pos VBG --store word
```

1345 unique tuples matching * word, VBG POS
34 times: , according to
8 times: <BOS> According to

If you only want to store the part of speech tags, set the `--store` flag to pos

```
$ python lexicalized_spans_ud.py train.conllu --pos VBG --store pos
```

280 unique tuples matching * word, VBG POS
65 times: IN VBG DT
47 times: , VBG DT

(Note that the `<BSP>` and `<ESP>`) part of speech are defined separately; it is not an official part of the dataset. They represent special start/end POS tags.)