≡ CSCI 544 Fall 2022   Home   Assignments ⌄   Article Presentation   Project ⌄   Sch

Coding Assignment 3

Due: October 11, 2022, at 14:00 Pacific Time (2:00 PM)

This assignment counts for 10% of the course grade.

Assignments turned in after the deadline but on or before October 12 are subject to a 10% grade penalty.

Overview   ⌃

In this assignment you will write a Hidden Markov Model part-of-speech tagger for Italian, Japanese, and a surprise language. The training data are provided tokenized and tagged; the test data will be provided tokenized, and your tagger will add the tags. The assignment will be graded based on the performance of your tagger, that is how well it performs on unseen test data compared to the performance of a reference tagger.

Note: There are many implementations of Hidden Markov Models available on the internet and at other places. **You may not consult any of these implementations, even for reference.** You must implement the algorithm yourselves, using references such as Jurafsky and Martin, chapter 8 and appendix A, or the problem handout distributed on Blackboard.

ⓘ

Data   ⌃

A set of training and development data is available as a compressed ZIP archive on Blackboard. The uncompressed archive contains the following files:

- Two files (one Italian, one Japanese) with tagged training data in the word/TAG format, with words separated by spaces and each sentence on a new line.

- Two files (one Italian, one Japanese) with untagged development data, with words separated by spaces and each sentence on a new line.

- Two files (one Italian, one Japanese) with tagged development data in the word/TAG format, with words separated by spaces and each sentence on a new line, to serve as an answer key.

- A readme/license file (which you won't need for the exercise).

The submission script will train your model on the tagged training data and test it on the development data (separately for Italian and Japanese). The grading script will train your model on the combined tagged training and development data (separately for Italian and Japanese), and test the model on unseen data in a similar format. The grading script will do the same for the surprise language, for which all of the training, development and test data are unseen.

Programs   ⌃

You will write two programs in **Python 3** (Python 2 has been deprecated): `hmmlearn.py` will learn a hidden Markov model from the training data, and `hmmdecode.py` will use the model to tag new data.

The learning program will be invoked in the following way:

```
> python hmmlearn.py /path/to/input
```

The command-line argument is a single file containing the training data; the program will learn a hidden Markov model, and write the model parameters to a file called `hmmmodel.txt`. The format of the model is up to you, but it should follow the following guidelines:

- The model file should contain sufficient information for `hmmdecode.py` to successfully tag new data.

- The model file should be human-readable, so that model parameters can be easily understood by visual inspection of the file.

The tagging program will be invoked in the following way:

```
> python hmmdecode.py /path/to/input
```

The command-line argument is a single file containing the test data; the program will read the parameters of a hidden Markov model from the file `hmmmodel.txt`, tag each word in the test data, and write the results to a text file called `hmmoutput.txt` in the same format as the training data.

The accuracy of your tagger is determined by a scoring script which compares the output of your tagger to a reference tagged text. Note that the tagged output file `hmmoutput.txt` must match line for line and word for word with the input to `hmmdecode.py`. A discrepancy in the number of lines or in the number of words on corresponding lines will cause the scoring script to fail.

Submission   ⌃

All submissions will be completed through Vocareum.

Multiple submissions are allowed; only the final submission will be graded. Each time you submit, a submission script trains your model (separately for each language) on the training data, runs your tagger on the development data, and reports the results. Do not include the data in your submission: the submission script reads the data from a central directory, not from your personal directory. You are encouraged to **submit early and often** in order to iron out any problems, especially issues with the format of the final output.

The performance of you classifier will be measured automatically; failure to format your output correctly may result in very low scores, which will not be changed.

For full credit, make sure to submit your assignment well before the deadline. The time of submission recorded by the system is the time used for determining late penalties. If your submission is received late, whatever the reason (including equipment failure and network latencies or outages), it will incur a late penalty.

If you have any issues with Vocareum with regards to logging in, submission, code not executing properly, etc., please make a post on Piazza so the instructional team can look into the issue.

Grading   ⌃

After the due date, we will train your model (separately for each language) on a combination of the training and development data, run your tagger on new (unseen) test data, and compute the accuracy of your output compared to a reference annotation. Your grade will be the accuracy of your tagger, scaled to the performance of a reference HMM tagger developed by us. Since part-of-speech tagging can achieve high accuracy by using a baseline tagger that just gives the most common tag for each word, only the performance above the baseline will be scaled:

- If your accuracy <= baseline accuracy, your grade is your accuracy.

- If baseline accuracy < your accuracy < reference accuracy, your grade is baseline + (1 – baseline) × (yours – baseline) / (reference – baseline).

- If reference accuracy <= your accuracy, your grade is 100%.

For example, if the baseline is 90%, the reference is 95%, and your accuracy is 93%, then your grade will be 0.9 + 0.1 × 0.03 / 0.05 = 96%.

Each language (Italian, Japanese, and the surprise language) is worth one-third of the overall grade for this assignment.

Notes   ⌃

- **Tags.** Each language has a different tagset; the surprise language will have some tags that do not exist in the Italian and Japanese data. You must therefore build your tag sets from the training data, and not rely on a precompiled list of tags.

- **Slash character.** The slash character '/' is the separator between words and tags, but it also appears within words in the text, so be very careful when separating words from tags. Slashes never appear in the tags, so the separator is always the last slash in the word/tag sequence.

- **Smoothing and unseen words and transitions.** You should implement some method to handle unknown vocabulary and unseen transitions in the test data, otherwise your programs won't work.

    - **Unseen words:** The test data may contain words that have never been encountered in the training data: these will have an emission probability of zero for all tags.

    - **Unseen transitions:** The test data may contain two adjacent unambiguous words (that is, words that can only have one part-of-speech tag), but the transition between these tags was never seen in the training data, so it has a probability of zero; in this case the Viterbi algorithm will have no way to proceed.

    The reference solution will use add-one smoothing on the transition probabilities and no smoothing on the emission probabilities. For unknown tokens in the test data, it will ignore the emission probabilities and use the transition probabilities alone, and also limit the tag inventory to only the tags with a large associated vocabulary (*open-class* items). You may use more sophisticated methods which you implement yourselves.

- **End state.** You may choose to implement the algorithm with transitions ending at the last word of a sentence (as in Jurafsky and Martin, figure 8.10), or by adding a special end state after the last word (see for example an older draft of Jurafsky and Martin, figure 9.11). The reference solution will use an end state.

- **Runtime efficiency.** Vocareum imposes a limit on running times, and if a program takes too long, Vocareum will kill the process. Your program therefore needs to run efficiently. One common source of runtime inefficiency in Viterbi decoding is multiplying a lot of incoming transitions by an emission probability of zero; testing for that zero and skipping all that multiplication can cut the runtime by 90%. Run times for the reference solution are approximately 1 second for running hmmlearn.py on the training data and 3 seconds for running hmmdecode.py on the development data, running on a MacBook Pro from 2016.

Collaboration and external resources   ⌃

- This is an individual assignment. You may not work in teams or collaborate with other students. You must be the sole author of 100% of the code you turn in.

- You may not look for solutions on the web, or use code you find online or anywhere else.

- You may not download the data from any source other than the files provided on Blackboard, and you may not attempt to locate the test data on the web or anywhere else.

- You may use packages in the Python Standard Library, as well as numpy. You may not use any other packages.

- You may use external resources to learn basic functions of Python (such as reading and writing files, handling text strings, and basic math), but the extraction and computation of model parameters, as well as the use of these parameters for tagging, must be your own work.

- Failure to follow the above rules is considered a violation of academic integrity, and is grounds for failure of the assignment, or in serious cases failure of the course.

- We use plagiarism detection software to identify similarities between student assignments, and between student assignments and known solutions on the web. **Any attempt to fool plagiarism detection, for example the modification of code to reduce its similarity to the source, will result in an automatic failing grade for the course.**

- Please discuss any issues you have on the Piazza discussion boards. Do not ask questions about the assignment by email; if we receive questions by email where the response could be helpful for the class, we will ask you to repost the question on the discussion boards.