Intro to Comp Ling: Assignment

Due: Monday 27 July by email to instructor.

General instructions

Please choose *one* of the two options to submit for your assignment. You may work with one other student from the class. Your submission should include both students' names, and a statement of how each student contributed to the final submission.

Option 1

Note that the amount of credit given for each question doesn't necessarily reflect how difficult each part is, especially for *you*, depending on your background. Q2 will take more time and is conceptually more difficult than Q1; if you find you are having a lot of trouble with it you might want to move on to Q3 where it should be easier to get at least partial credit.

1. (25%) For this question, we consider a simple HMM tagger with only five tags (plus the beginning and end of sentence markers, `<s>` and `</s>`). The transition probabilities for this HMM are given by the table on the left below, with cell $[i,j]$ the probability of transitioning from state i to j (i.e., $P(\text{state}_j|\text{state}_i)$). A subset of the output probabilities are given by the table on the right, with cell $[i,j]$ the probability of state i outputting word j (i.e., $P(\text{word}_j|\text{state}_i)$). (We assume there are other possible output words not shown in the table, and that the `<s>` and `</s>` states output `<s>` and `</s>` words, respectively, with probability 1.)

	CD	PRP	NN	VB	VBD	</s>
<s>	.5	.2	0	.3	0	0
CD	.2	0	.3	.2	.2	.1
PRP	.1	.1	0	.3	.4	.1
NN	.05	.15	.2	.25	.3	.05
VB	0	.2	.6	0	0	.2
VBD	0	.1	.6	0	0	.3

	one	cat	dog	bit	...
CD	.1	0	0	0	
PRP	.02	0	0	0	
NN	0	.03	.04	.007	
VB	0	0	.03	0	
VBD	0	0	0	.06	

For the sentence $\vec{w}$ = `<s> one dog bit </s>` and tag sequence $\vec{t}$ = `<s> CD NN NN </s>`, what is $P(\vec{w},\vec{t})$, the probability of observing that sentence with that tag sequence? Give the equation used to compute the answer, and show what numbers you plug into that equation.

2. (25%) Now, using the same HMM, hand-simulate the Viterbi algorithm to compute the *best* tag sequence for the given sentence. That is, fill in the cells in the following table, where cell $[i,j]$ should contain the Viterbi value for state i at time j, and you should also use backpointers to keep track of the best path. The rows of the table are already labeled with the different states, and the columns are already labeled with the observations at each time step.

	<s>	one	dog	bit	</s>
<s>					
CD					
PRP					
NN					
VB					
VBD					
<s>					

Your solution should give the best tag sequence, and show the computation involved to get the values in each cell as well as the values obtained. You could do this by sending a photo of your (neat!) work on paper, or (preferably) by typing up your solution, listing the computation for each cell [i,j]. You only need to list computation/results for cells that have non-zero values. Assume the cell numbering starts with 0, or use the labels provided. For example:

$[0, 0]$: *show computation and result here*
$[1, 0]$: *show computation and result here*
$[2, 0]$: *show computation and result here*
...

or

[<s>,<s>]: *show computation and result here*
[CD, <s>]: *show computation and result here*
[PRP, <s>]: *show computation and result here*
...

Note: You only need to list computation/results for cells that have non-zero values.

3. (50%) For this part, choose any *one* of the following three questions:

   a. Question 1 from the "Going further" section of Lab 1 (Good-Turing estimation).

   b. Question 1 from the "Going further" section of Lab 2 (POS tagging for Twitter). If you choose this question, you may want to look at the sample of Twitter data that is provided for download under Option 2. However please see the note there about anonymity: usernames have been replaced with digit strings, and the words in these tweets have been scrambled, so you shouldn't assume anything about word order in Tweets based on this sample. (In other words, you should only use this data to give you some ideas about the kinds of words that appear in Tweets, not the syntax.)

   c. Question 2 from the "Going further" section of Lab 2 (Comparing unigram and HMM taggers).

You should be able to answer these questions well with only a page or so of text, possibly with some additional space if you need to include figures or tables. Please do not turn in anything longer than three pages. If you use any outside resources to provide examples, evidence, or ideas for your answers, please make sure to cite those resources properly.

Option 2

This option requires more programming than the first option. You do not need to do the programming in Python, but please do not attempt this question unless you have good programming skills in some language.

Develop a program that uses an n-gram character language model to filter English Tweets from non-English Tweets. That is, your program should learn the probabilities of n-grams of English characters from text that is known to be written in English. Using this model, you should be able to compute the average per-character probability of each Tweet. If a Tweet is written in English, it should (in general) have a higher probability under your model than if it is not written in English, so if you sort the Tweets by their probability under your model, you should find that the English ones mostly cluster at the top of the list.

To complete your filter, you would also need to choose a cutoff point somewhere in the list to classify anything with higher probability than that cutoff as English, and anything else as non-English. However you do not need to actually implement this part.

In order to develop your program, you will need the following data:

- English text to train the character language model. You can download a small subset of English Europarl data from here, or you might wish to use your own text: `http://homepages.inf.ed.ac.uk/sgwater/teaching/lsa2015/data/training.en`.
- To test your program you can download a small sample of Tweets from here: `http://homepages.inf.ed.ac.uk/sgwater/teaching/lsa2015/data/tweets.txt`. Note that as a condition of using them, these Tweets have been preprocessed to anonymize them: usernames have been replaced by strings of numbers, and the words in the Tweets have been reordered (so they are effectively bags of words).

Hints:

- You may want to start with simple filters like removing words that contain non-alphanumeric characters.
- It may be necessary to use some simple smoothing in your model, but you shouldn't need to do anything more complicated than add-alpha because unlike a word n-gram model, the "vocabulary" size in a character n-gram model is not very large.
- You are free to use NLTK or write your own code for estimating the model, whichever you prefer.

Your submission for this question should include the following:

1. A brief (1-2 paragraph) description of how your program solves the task: e.g., what (if any) preprocessing does it do on the data? What information does it collect from the training data and what does it do with that information? If you tried more than one version of your model, you can mention briefly any differences you might have seen.

2. A brief description of what else you would need to do to turn this into a real filter (i.e., how would you decide at what point in the list to set the boundary between English and non-English?)

3. A list of the top 10 and bottom 10 Tweets, as ranked according to the probability under your model. Also, take a look at some Tweets in the middle of the list. Is there a clear cutoff between English and non-English, or is it more fuzzy? Can you point to any particular reasons why some Tweets are harder to classify than others? Are there ways you could think of to try to improve the system's performance?

Please limit your submission to at most three pages. You might have a lot more you could say, but please choose the things you feel are most important.