HW Assignment 2 (Due date: Thu, Mar 17, by 9:00am)

1 Manual POS Tagging [50 points]

Exercises 5.1 and 5.2, page 171 in J&M. For each exercise, save the correct POS tagging in a file, one sentence per line, using the format shown in 5.1.

2 Viterbi Algorithm [50 points]

Consider the following parameters for a very simple HMM:

Noun ($\pi = 0.5$)		Verb ($\pi = 0.2$)		Prep ($\pi = 0.1$)		Det ($\pi = 0.2$)	
time	$b = 0.5$	time	$b = 0.1$	like	$b = 0.3$	a	$b = 0.3$
flies	$b = 0.2$	flies	$b = 0.4$	on	$b = 0.3$	an	$b = 0.2$
arrow	$b = 0.3$	like	$b = 0.5$	in	$b = 0.4$	the	$b = 0.5$

Table 1: Start and Observation probabilities.

	Noun	Verb	Prep	Det
Noun	$a = 0.2$	$a = 0.5$	$a = 0.3$	$a = 0.0$
Verb	$a = 0.3$	$a = 0.0$	$a = 0.3$	$a = 0.4$
Prep	$a = 0.4$	$a = 0.0$	$a = 0.1$	$a = 0.5$
Det	$a = 0.9$	$a = 0.0$	$a = 0.1$	$a = 0.0$

Table 2: Transition probabilities.

Create the trellis for the sentence *time flies like an arrow*. Show the computed parameters $\delta_i(t)$ and $\psi_i(t)$ at each node in the trellis. What is the most likely sequence of tags? What is its probability? Show all possible sequences of tags that receive non-zero probability in this model. List them together with their probability, ranked from most likely to least likely.

3 Maximum Entropy Tagging [50 points]

The probability of a tag sequence in Ratnaparkhi's MaxEnt tagger is given by:

$$p(t_1...t_n|w_1...w_n) = \prod_{i=1}^{n} p(t_i|h_i) \tag{1}$$

where $h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$. Is the forward procedure for tagging (shown on slide 45) quaranteed to find the most likely sequence of tags? If yes, explain why. If not, describe/name an efficient approach/algorithm that would find the most likely sequence.

4 HMM Tagging [50 points]

Show how to compute the probability of a sequence of k tags appearing at a given position in an observation sequence, given the HMM model parameters and the factors α and β. Can you think of a possible use for this probability?
More exactly, show how to compute $p(x_{t+1} = s_{i_1}, ..., x_{t+k} = s_{i_k}|O, \mu)$ as a function of $\alpha_i(t)$, $\beta_j(t)$, a_{ij} and b_{jo_t}, where $t \in [1, T]$, $i, j \in [1, K]$.
Hint: when $k = 2$ this should reduce to the transition probability $\xi_t(i, j)$ (slide 49).

5 POS Tag Entropy [50 points]

Using the NLTK package, compute the tag entropy for each word in two corpora: the Penn Treebank, and the Brown corpus. Ignore words that appear less than 5 times, and use case-insensitive counting. For each corpus, create lists with the highest entropy 10 words in each corpus. Create a separate list for words that can have 2 tags, words that can have 3 tags, etc. Each list entry should display the word, its total count, its possible POS tags, and its tag entropy.

6 HMM POS Tagging [150 points]

In this exercise, you will use the NLTK package to train and evaluate HMM POS taggers. For training, you will use the section 00 of the Wall Street Journal section of the Penn Treebank (i.e. files `wsj_00??.pos`). For testing, you will use section 01 from the same corpus (i.e. files `wsj_01??.pos`).

- Train the following systems: HMM with MLE estimates, HMM with Laplace smoothing, HMM with Witten-Bell smoothing, HMM with Simple Good-Turing smoothing, and a Most Likely Tag tagger using NN for unknown words. Evaluate all these models on the test data and report accuracy for 1) all words, 2) out-of-vocabulary words, 3) all words without punctuations, 4) the top 10 highest entropy words. Discuss your results.

- For the best HMM model, compute two types of accuracy results: one for using *Viterbi tagging*, one for using the *most likely tag* for each word (computed using $\gamma_t(i)$ on slide 49). Discuss your results.

- For the best HMM model, compute and plot a learning curve by training on the first 10%, 20%, ..., 100% of the files in section 00.

For this exercise, you might consider using the `nltk.probability` and `nltk.tag.hmm` modules, the `HiddenMarkovModelTrainer` class and its `train_supervised` method.

7 Discriminative HMMs & Word Embeddings [200+ bonus points]

Implement Michael Collins' Discriminative HMM and train and test the averaged and non-averaged versions on the same WSJ sections as in problem 6 above, in the following scenarios:

1. Using global features derived from the local features of a traditional HMM. Ignore features that appear less than 5 times in the training corpus.

2. Use word embeddings to create an additional set of features and add them to the features above.

Compare the performance of the systems above with the traditional HMM performance at problem 6. For additional bonus points, repeat the same experiments with global features derived from Ratnaparki's local features, with and without word embedding features.

8 Submission

Please turn in a hard copy of your homework report at the beginning of class on the due date.

Electronically submit a directory that has your working code, and a concise README file describing these before class. Do not send NLTK code! Create a gzipped, tar ball archive of your directory, and upload it on Blackboard.

For example, if the name is John Williams, creating the archive can be done using the following commands:

> tar cvf williams_john.tar williams_john

> gzip williams_john.tar

These two steps will create the file 'williams_john.tar.gz' that you can upload on Blackboard.

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.

2. Use adequate comments, both block and in-line to document your code.

3. Type and nicely format the project report, including discussion points, tables, graphs etc. so that it is presentable and easy to read.

4. Working code and/or correct answers is only one part of the assignment. The project report, including discussion of the specific issues which the assignment asks about, is also a very important part of the assignment. Take the time and space to make an adequate and clear project report. On the non-programming learning-theory assignment, clear and complete explanations and proofs of your results are as important as getting the right answer.