CSC 483/583: Assignment #4 (75 pts)

Due by 11:59 P.M., Sunday November 21
(Upload code for problems 4 and 5 to GitHub Classroom; problems 1 – 3 in Gradescope)

Because the credit for graduate students adds to more than 75 points, graduate students' grades will be normalized at the end to be out of 75. For example, if a graduate student obtains 80 points on this assignment, her final grade will be $80 \times 75/105 = 57.1$. Undergraduate students do not have to solve the problems marked "grad students only." If they do, their grades will not be normalized but will be capped at 75. For example, if an undergraduate student obtains 80 points on this project (by getting some credit on the "grad students only" problem), her final grade will be 75.

Note that only problems 4 and 5 require coding.

Problem 1 (20 points): Binary Independency Model

- Query `Obama health plan`
- Doc1 `Obama rejects allegations about his own bad health`
- Doc2 `The plan is to visit Obama`
- Doc3 `Obama raises concerns with US health plan reforms`

Estimate the Retrieval Status Value for the above documents and the query using the Binary Independence Model with smoothing (add $\frac{1}{2}$). These are the only three documents in the collection. Show all work, e.g., what is N, S, s? What is the formula for each term's c_t? What is the formula for each document's RSV?

Problem 2 (20 points undergrad, 30 grad): Naive Bayes

Your task is to classify words as English or not English. Words are generated by a source with the following distribution:

event	word	English?	probability
1	ozb	no	4/9
2	uzu	no	4/9
3	zoo	yes	1/18
4	bun	yes	1/18

1. (15 pts) Compute the parameters (priors and conditionals) of a multinomial NB classifier that uses the *letters* (not the words as in the lecture example) `b`, `n`, `o`, `u`, and `z` as features. Assume a training set that reflects the probability distribution of the source perfectly. Make the same independence assumptions that are usually made for a multinomial classifier that uses terms as features for text classification. Compute parameters using a simple smoothing approach, in which computed-zero probabilities are smoothed into probability 0.01, and computed-nonzero probabilities are untouched. (This simplistic smoothing may cause $P(A) + P(A) > 1$. Solutions are not required to correct this.)
2. (5 pts) How does the classifier classify the word `zoo`?
3. (GRAD STUDENTS ONLY – 10 pts) Classify the word `zoo` using a multinomial classifier as in part (1), but do not make the assumption of positional independence. That is, estimate separate parameters for each position of a letter in a word. For example, in the word `zoo` there will be two letters `o`: $\texttt{o}_2$ and $\texttt{o}_3$, corresponding to the two positions in which they appear. You only need to compute the parameters you need for classifying `zoo`.

Problem 3 (20 points): Language Models

Suppose we have a collection that consists of the 4 documents given in the table below.

docID	Document text
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

Build a query likelihood language model for this document collection. Assume a mixture model between the documents and the collection, with both weighted at 0.5. Maximum

likelihood estimation (MLE) is used to estimate both as unigram models. Work out the model probabilities of the queries `click`, `shears`, and `click shears` for each document, and use those probabilities to rank the documents returned by each query. Fill in these probabilities in the table below. Show work for all probability computations.

Query	Doc 1	Doc 2	Doc 3	Doc 4
click				
shears				
click shears				

Problem 4 (15 points): Implementing a Language Model

Consider the same set of documents as in Assignment 3:

```
Doc1 information retrieval is the most awesome class I ever took.
Doc2 the retrieval of private information from your emails is a job that the NSA loves.
Doc3 at university of arizona you learn about data science.
Doc4 the labrador retriever is a great dog.
```

Implement from scratch a query likelihood language model over the above documents. Note: using Lucene or another existing IR library is not allowed for this problem.

- (2 pts) Tokenize and lemmatize the above documents using an existing NLP library, e.g., Stanford's CoreNLP (`http://stanfordnlp.github.io/CoreNLP/`) for Java, Arizona's Processors (`https://github.com/clulab/processors`) for Scala or Java, and SpaCy for Python (`https://spacy.io`).
- (10 pts) Implement the query likelihood language model (lecture 12) over lemmas. Additionally, your implementation must support at least one of the smoothing algorithms covered in the lecture.
- (3 pts) What is the ranking of the above documents under this model for the query "information retrieval"? (Your code must support queries of any length!) Show the scores for each document. What are the scores if your model does *not* implement smoothing?

You will implement and submit this problem using GitHub Classroom:

- If you program in Python, click on this link and follow the instructions on the screen: TO BE POSTED SOON

- If you program in Java, click on this link and follow the instructions on the screen: TO BE POSTED SOON

Very important note: **make sure the unit tests in your project pass on GitHub, after you submit your pull request!** If they do not, you will lose 50% of the credit for this problem, i.e., 15 points for graduate students, and 10 points for undergraduates.

Problem 5 (GRAD STUDENTS ONLY – 20 points): Implementing a Naive Bayes Model

Implement from scratch the multinomial Naive Bayes (NB) classifier with add-one smoothing from lecture 13. Note: using an existing Naive Bayes implementation from a machine learning package is not allowed for this problem. This code **must** be submitted through GitHub Classroom, in the same project as the code used for Problem 4, and using the same Classroom links.

- (10 pts) Implement the NB training procedure using the training portion of the spam dataset (the dataset is included in the GitHub project). How long does it take to train this classifier?
- (10 pts) What is the accuracy of the classifier over the testing portion of the same dataset? What is the F1 score with respect to the spam label? That is, compute the precision/recall of the spam label alone, and then average them into an $F_{\beta=1}$ score.