CS540 Summer 2020 Epic Section

Home
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 9
Week 10
Week 11
Week 12
Week 13
Midterm
Final

Previous: P2, Next: P4

Back to week 5 page: Link
Back to week 6 page: Link

Official Due Date: July 5

Programming Problem Instruction

• Enter your ID here: grlow and click Confirm
• The same ID should generate the same set of parameters. Your answers are not saved when you close the browser. You could either copy and paste your console output into the text boxes or print your output to text files (.txt) and load them using the button above the text boxes.
• Please report any bugs on Piazza.

• (Introduction) In this programming homework, you will build and simulate a simple Markov chain model based on a movie script. You will use it to generate new sentences that hopefully contain sensible words maybe even phrases. In addition, you will build a Naive Bayes classifier to distinguish sentences from the script and sentences from another fake script. Due to the English vocabulary size, you will use characters as features instead of words. In practice, you could replace the 26 letters by (more than 170,000) English words when training these models.

• (Part 1) Download the script of one of the following movies: "Batman Begins", "Warrior", "The Prestige" from IMSDb: Link. If you have another movie you really like, you can use that script instead. Go to the website, use "Search IMSDb" on the left to search for this movie, click on the link: "Read ... Script", and copy and paste the script into a text file.

• (Part 1) Make everything lower case, then remove all characters except for letters and spaces. Replace consecutive spaces by one single space and make sure that there are no consecutive spaces.

• (Part 1) Construct the bigram character (letters + space) transition probability table. Put "space" first then "a", "b", "c", ..., "z". It should be a 27 by 27 matrix.

• (Part 1) Construct the trigram transition probability table. It could be a 27 by 27 by 27 array or a 729 by 27 matrix. You do not have to submit this table.

• (Part 1) Generate 26 sentences consists of 1000 characters each using the trigram model starting from "a", "b", "c", ..., "z". You should use the bigram model to generate the second character and switch to the bigram model when the current two-character sequence never appeared in the script. For example, when you see "xz", instead of using the trigram model for the probabilities of Pr{? | xz}, switch to use the bigram model for the probabilities of Pr{? | z}. Find and share some interesting sentences.

• (Part 2) The following is my randomly generated script written according to an non-English language model.

```
zzqq xjqqjz fqz m
jqq q zuxxqxz
```

You can either use the Download button to download a text file, or copy and paste from the text box to a text file. Please do not change the content of the text box.

• (Part 2) Train a Naive Bayes classifier based on the characters. You should use an uniform prior, i.e. Pr{Document = your script} = Pr{Document = my script} = 0.5, compute the likelihood Pr{Letter | Document} based

on your script and my script, and compute the posterior probabilities Pr{Document | Letter}, and test your classifier on the 26 random sentences you generated.

Submission

Question 1 [1 points]

• Please enter the name of the movie script you used.
• Answer:

Question 2 [5 points]

• (unigram) Input the unigram probabilities (27 numbers, comma-separated, rounded to 4 decimal places, "space" first, then "a", "b", ...).

Question 3 [5 points]

• (bigram) Input the bigram transition probabilities without Laplace smoothing (27 lines, each line containing 27 numbers, comma-separated, rounded to 4 decimal places, "space" first, then "a", "b", ...).

Question 4 [5 points]

• (bigram_smooth) Input the bigram transition probabilities with Laplace smoothing (27 lines, each line containing 27 numbers, comma-separated, rounded to 4 decimal places, "space" first, then "a", "b", ...).
• Hint: to make sure auto-grading marks your answer correct, please make sure all probabilities are strictly larger than 0 after rounded to 4 decimal places, for example, 0.00002 should be rounded up to 0.0001 instead of 0.0.

Question 5 [10 points]

• (sentences) Input the 26 sentences generated by the trigram and bigram models (Laplace smoothed). (26 lines, each line containing 1000 characters, line 1 starts with "a", line 2 starts with "b" ...).

Question 6 [2 points]

• Find one interesting sentence that at least contains English words.
• Answer:

Question 7 [5 points]

• (likelihood) Enter likelihood probabilities of the Naive Bayes estimator for my script. (27 numbers, comma separated, rounded to 4 decimal places, Pr{"space" | D = my script}, Pr{"a" | D = my script}, Pr{"b" | D = my script}, ...). The likelihood probabilities for your script should be the same as your answer to Question 2.

Question 8 [5 points]

• (posterior) Enter posterior probabilities of the Naive Bayes estimator for my script. (27 numbers, comma

separated, rounded to 4 decimal places, Pr{D = my script | "space}, Pr{D = my script | "a"}, Pr{D = my script | "b"}, ...).

Question 9 [5 points]

• (predictions) Use the Naive Bayes model to predict which document the 26 sentences your generated in Question 5. Remember to compare the sum of log probabilities instead of the direct product of probabilities. (26 numbers, either 0 or 1, 0 is the label for your script, 1 is the label for mine)
• Hint: your prediction is supposed to be all 0s, but if there are few 1s, it is okay, but make sure you generated the sentences in Question 5 correctly according to the instruction (especially switching to bigram from trigram when necessary).

Question 10 [1 points]

• Please enter any comments and suggestions including possible mistakes and bugs with the questions and the auto-grading, and materials relevant to solving the question that you think are not covered well during the lectures. If you have no comments, please enter "None": do not leave it blank.

• Answer: .

Grade

Grade
***** ***** ***** ***** *****

***** ***** ***** ***** *****

• Warning: grading may take around 10 to 20 seconds. Please be patient and do not click "Grade" multiple times.

• Please copy and paste the text between the *****s (not including the *****s) and submit it on Canvas, P3.
• Please submit your code and outputs on Canvas, P3S.
• You could also save your output as a single text file using the button Download and submit this to P3S (with your code).
• Warning: the load button does not function properly for all questions, please recheck everything after you load. You could load your answers using the button Load from the text field:
Browse... No file selected.

• Saving and loading may take around 10 to 20 seconds. Please be patient and do not click the buttons multiple times.

Hints and Solutions (frequently updated)

• The tolerance of error for Question 5 has decreased: as long as the unigram and bigram probabilities inferred from sentences you generated is within plus or minus 0.1 of the probabilities you enter in Question 2 and 4, you will receive full points.
• The posterior probabilities are defined as: Pr{D|a} = Pr{a|D} Pr{D} / (Pr{a|D} Pr{D} + Pr{a|not D} Pr{not D}), where Pr{D} = Pr{not D} = 0.5 and Pr{a|D} is the unigram probability of a for your script and Pr{a|not D} is the unigram probability of a for my script.
• For both all (smoothed or not smoothed) probabilities make sure each line add up to 1.0000.
• A sample solution in Java and Python is posted below.
Important notes:

(1) Smoothing is not done. You have to implement it.
(2) Classification is not done, you should implement it using the hint. Basically, label 0 iff log Pr{D = your script | first letter} + log Pr{D = your script | second letter} + log Pr{D = your script | third letter} + ... > log Pr{D = my script | first letter} + log Pr{D = my script | second letter} + log Pr{D = my script | third letter} + ... Each of these posterior probabilities should be from your answer in Question 8. [Explanation: Unigram model implies the probability of observing a sentence is the product of the probability of observing each character, but taking the product between small numbers will cause numerical problems (i.e. you cannot compare two numbers that are too close to 0 because they are both stored on a computer as 0). Taking the log will preserve the inequality, and log of a product is equal to the sum of the logs, that is why you get the above expression.]
(3) You are allowed to copy and use parts of the TA's solution without attribution. You are allowed to use code from other people and from the Internet, but you must state in the comments clearly where they come from!

Python code by Hugh Liu: Link
Java code by Ainur Ainabekova: Link.

Last Updated: July 17, 2020 at 3:31 AM

UNIVERSITY OF WISCONSIN-MADISON