University of Arizona	Homework#2	ngrams, language models
CSC 585 Algorithms for NLP	Due date:	September 19, 2023
Fall 2023	**Name:**	
Instructor: Eduardo Blanco		

Instructions

- Bring a hard copy to class with your answers to Questions 1–3, the written part of Question 4, and Question 5 if you choose to also answer the last question.
- Submit your implementation for Questions 4 on GitHub Classroom. You will find the invite link for the repository in d2l. Follow these steps:

```
# Accept the invite link for the assignment available at d2l.
# It will create a repo named 02-ngrams-lms-GITHUB_USER
#   Please use a github user name that is your name (or something close to it).
# Then set up the environment
> git clone YOUR_REPO
> cd 02-ngrams-lms-GITHUB_USER
> python3 -m venv .env
> source .env/bin/activate
> pip install -r requirements.txt
```

Grading is automatic. You have access to all the public test cases. You can run the test cases with the following command: `> python -m pytest`. You can also run individual test cases (e.g., `python -m pytest -k test_calculate_counts_unigrams`).

Question 1 [10pt]
Read the following paper and answer the questions below:

> Holgate, E., Cachola, I., Preoţiuc-Pietro, D. and Li, J.J., 2018. Why swear? analyzing and inferring the intentions of vulgar expressions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 4405-4414).
>
> Pdf file: `https://aclanthology.org/D18-1471/`
> Video presentation: `https://vimeo.com/306123618`
>
> Note: There are many offensive words in the paper. If you think you can be offended by reading a discussion of why people swear, please talk to the instructor and we will find an alternative.

- How do they perform intrinsic evaluation? Spell out the task (specifically, the input and output: what problem are they solving?)
- How do they perform extrinsic evaluation? Spell out the task (specifically, the input and output: what problem are they solving?). Why is it an extrinsic evaluation?

Note: You have to read the paper and understand the evaluation. You do not need to understand all the details about how they solve the problems. The authors do not use the term intrinsic or extrinsic anywhere in the paper.

Question 2 [5pt]
Read the README file for the *Web 1T 5-gram Corpus Version 1.1 (LDC2006T13)*. We discussed this resource briefly in class.

In any given corpus there are more 5-grams than 4-grams. The counts in Section 5 of the README file, however, state that the count of 4-grams is larger than the count of 5-grams in the *Web 1T 5-gram Corpus Version 1.1 (LDC2006T13)*:

```
Number of fourgrams: 1,313,818,354
Number of fivegrams: 1,176,470,663
```

How can justify this?

Note: the counts are correct, and the justification is somewhere (kind of) hidden in the README file.

Question 3 [10pt]
Listen to (or read the transcript of) the radio piece entitled *Christopher Marlowe Credited As Shakespeare's Co-Author On Henry VI Plays* (All Things Considered, NPR).[1] Answer the following questions based on the paragraph below (you still need to read the whole interview):

> TAYLOR: For one example, the word glory is not all that unusual in plays of the period. And the verb droopeth, you know, it occurs in a number of different writers. But if you put those two words together right next to each other, glory droopeth, that occurs in one of these disputed passages in "Henry VI, Part 1." The only other place it occurs in all the plays of the period is in a play by Marlowe.

- Whose language are their language models for?
- P(glory) is much lower than the probability of most words in plays of the period.
- There is only one author for which P(droopeth) is not zero.
- P(droopeth | glory) is higher in the corpus of Shakespeare's plays than in the corpus of Marlowe's plays.

The answer to all questions is either True or False and a brief justification (1-3 sentences).

Question 4 [75pt]
Language identification is the problem of determining the language a text is written in. N-gram language models can solve this problem. We will focus on identifying whether a text is written in English or Spanish. You have to create a language model for English and another language model for Spanish. Then, use these models to predict the language a new text is written in. Your language models should be built at the character level.

The starter code includes some basic logic but you are responsible for most of the work. The training and test documents come from Project Gutenberg.[2] `language_detector.py` will create models for English and Spanish, and use them to predict the language of the test documents. To run it, use the following command (you may have to change the paths):

[1] `https://www.npr.org/2016/10/24/499199341/christopher-marlowe-credited-as-shakespeares-co-author-on-henry-vi-plays`

[2] `http://www.gutenberg.org/`.

```
$ python language_detector.py \
    ../data/train/en/all_en.txt \
    ../data/train/es/all_es.txt \
    ../data/test/
INFO:root:Prediction for *en* documents:
INFO:root:pg345.txt       unk
INFO:root:pg1497.txt      unk
INFO:root:pg3526.txt      unk
INFO:root:pg103.txt       unk
INFO:root:pg16.txt        unk
INFO:root:news1.txt       unk
INFO:root:news3.txt       unk
INFO:root:news2.txt       unk
INFO:root:
INFO:root:Prediction for *es* documents:
INFO:root:pg15725.txt     unk
INFO:root:pg21906.txt     unk
INFO:root:pg14311.txt     unk
INFO:root:pg25956.txt     unk
INFO:root:news1.txt       unk
INFO:root:news3.txt       unk
INFO:root:news2.txt       unk
INFO:root:pg31465.txt     unk
INFO:root:
Accuracy: 0.0
```

The code does not do anything useful until you implement two functions: `create_model(path)` and `predict(file, model_en, model_es)`. Your job is to implement a **character bigram model**. Your implementation does not work if you do not output the right language for all files.

A few notes:

- Remember to add special characters (e.g., "$") before and after each token (not each sentence) prior to calculating the n-grams. Add one special character when using bigrams (*the*: *the*), and two if you choose to work with trigrams (*the*: *$$the$$*).
- Use add-one smoothing to account for unseen n-grams during training. In general,

$$P(y|x) = \frac{\text{number of times xy occurs} + 1}{\text{number of times x occurs} + 26}$$

  This formula is equivalent to adding one count of each possible bigram in your corpus (assuming there are 26 characters, which is true if you only account for lower case letters).
- Remember to use log probabilities (and adding) instead of raw probabilities (and multiplying). Otherwise you are likely to get zero probabilities and the predictions will be useless.
- **A good implementation runs in less than 5 seconds. Yours must run in less than 1 minute to receive credit.**
- **An implementation that always predicts the right language will not receive full credit if if does not use a bigram language model or does not properly smooth probabilities.**

Submit your implementation on Github. Write a short report (up to 4 pages, but a perfect report can be as short as 2 pages or even less) describing the challenges you found (bug fixes, results obtained, and any other findings). Also, discuss the following:

1. Do you think it makes sense to create a language model at the character level instead of at the word level for this task? What are the advantages of working with characters instead of words?
2. Take a look at the test documents for English and Spanish. Are documents written only in one language?
3. What is the minimum number of tokens you need to process to always make the right predictions? You can try with 100 tokens, 200 tokens, 300, etc. instead of the whole document. You do not need an exact number.
   Just like humans, your code doesn't need to read the whole book to figure out whether it is written in English or Spanish, assuming books are written in one language.
4. If you create several models for English and Spanish (using different training data, using different preprocessing, etc.), how can you compare them? Does your answer change if all the models correctly predict the right language? In other words, how would you declare a winner if two models always predict the correct language?
5. Can you train with less training data and still get the right predictions? How does training size affect predictions? A graph showing amount of training data and accuracy of your language models is probably the best way to answer this questions (and a short interpretation of the graph).
6. Get some document written in French (e.g., `http://www.gutenberg.org/files/36460/36460-0.txt`) and use your models to predict the language. What does your language detector predict? Can you justify the prediction?

You must answer the above questions in your report. Your implementation is worth up to 45 points. The report is worth 30 points. Breakdown for the implementation:

- `test_calculate_counts_unigrams`: 5 points
- `test_calculate_counts_bigrams`: 5 points
- `test_calculate_probabilities_unsmoothed`: 10 points
- `test_calculate_probabilities_smoothed`: 10 points
- `test_calculate_log_prob`: 10 points
- `test_language_detector`: 5 points

We will run your code with other (private) files to test that you calculate the counts and probabilities correctly.

Question 5 [5pt]

Extra credit: The maximum grade of the homework is 105. This question is extra credit.

Read the following paper and answer the questions below:

> Gonen, Hila and Yoav Goldberg. 2019. Language Modeling for Code-Switching: Evaluation, Integration of Monolingual Data, and Discriminative Training. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*
>
> Available here: `https://aclanthology.org/D19-1427/`

- What is code switching?

- Why is code switching an issue for language models?

- What would your language detector predict if you give it the sentences in the first column of Table 1?

Answers to these questions are short (1-3 sentences, but you can write more if you wan to).