Lab 2: Language Models

This assignment has several different parts and it is due in one week. The starter code and required data files are available in Canvas as a zip file.

(a) **Simple Character LM**. Examine the Python code provided (`charlm.py`). Then train an order 4 (i.e., 5-gram) model from the provided file `subtitles.txt`.

```
mylm = train_char_lm('subtitles.txt', 4)
```

What are the continuations of 'atio' ? What about 'nivi' ? And 'supe' ?

```
print_probs(mylm, 'atio')
```

The `generate_text` method can produce random strings from a trained model. Generate some sample strings (of up to length 80 characters) from the model. Show us three of your favorite examples produced by the model.

- 3 points: Appropriate continuations for atio, nivi, and supe.
- 1 point: Random sentences

(b) **Calculate perplexity**. Extend the code from `charlm.py` to calculate a perplexity score for a provided string against a particular model of a specified order.[1] Return positive infinity if any zero probabilities are encountered. Hint 1: you will want to work in log space. Hint 2: we advise reading the assigned chapter from the textbook and reviewing the lecture materials before implementing your perplexity function.

```
perplexity('The boy loves his mother', mylm, 4)
> 3.9092  (your result should be similar)
```

To test your implementation, provide perplexity values for the following strings using a 5-gram model from subtitles.txt

- The student loves homework
- The yob loves homework
- It is raining in London
- asdfjkl; qwerty

- 2 points: Provides requested test cases
- 3 points: Handles zero probabilities
- 7 points: Correct implementation

(c) **Naive smoothing**. Further extend the code to perform some trivial smoothing for the perplexity function you implemented earlier. Instead of returning positive infinity for zero probabilities, implement a `smoothed_perplexity` method that uses 1.0e-7 for "zeros". As a consequence the sum of all probabilities in your model no longer add up to 1, but the effect will not be very dramatic. Test your method on the same strings from section (b).

- 8 points: Working smoothed perplexity and provided requested test cases

(d) **Language Identification**. Training data for six European languages is provided. Create unigram (history=0) models for all six languages. Then make predictions for the provided test file which contains 1,200 lines. Note: each line in `test.txt` has two fields separated by a tab character; the first field is the correct language code:

```
fr      Quels que soient les témoins qui viendront déposer à la barre, M. Milosevic aura le droit de
les contre-interroger.
de      Sachsens Ministerpräsident Georg Milbradt verlangt, dass im Falle einer Mehrwertsteuererhöhung
die zusätzlichen Einnahmen zweckgebunden eingesetzt werden.
it      Peraltro, il vincolo ministeriale diviene oggi un importante strumento contrattuale.
```

[1] We encourage you to work in Python, but if you use another language you will have to reimplement the starter code.

For each input string calculate the smoothed perplexity for the six separate models and return the language code for the model with lowest smoothed_perplexity. For the first line of the test file (only), show perplexity scores for all six languages. Calculate and report accuracy for the six languages - each number correct should be between 0 and 200 (e.g., "de: 180 correct out of 200 lines - 90.0%"). Finally, repeat the experiment with higher-order n-grams. Try both a bigram and 4-gram model. How do your results change? You'll need to write code to load data, train models, calculate and sort scores, choose the lowest perplexity, and score results for the six languages.

- 1 point: Provide scores (perplexities) for each language on first test sentence.
- 2 points: Provides predictions for unigram model
- 2 points: Predictions for higher order models
- 3 points: Quality of results

(e) **Gender Bias**. For a different classification problem examine this data that contains questions that were asked to male or female tennis professionals after a televised match. Can you predict whether a question is addressed to a male or to a female player? The files are: `tennis.{train,test}.txt` and there are two fields separated by a tab. Like the language ID task, see how well you can classify male-directed vs. female-directed questions. See which length model works best. What is your best accuracy on the test set?

```
M       What was it like out there, Kevin?
M       That was pretty high quality. Have many people played that well against you on a grass court?
M       What do you take away from this?
M       If Andy can maintain that level of tennis, how do you see his Wimbledon chances?
F       Last year's champion is really into art. Who are some of your favorite artists?
F       You touched on your support of Argentina. How about the World Cup in general. What does the
World Cup mean to you and what is your passion for the event?
F       Do you think Wimbledon does that to an extent?
F       What goes through your mind when you watch Messi?
```

These data are from this paper: https://www.cs.cornell.edu/home/llee/papers/gender-tennis.pdf

- 2 points: Calculate accuracies for both classes
- 3 points: Show results for models of at least three n-gram orders.
- 3 points: Quality of results.

Comments on assignment:

- Tools. If you wish, you may use a publicly available language modeling tool for parts (d) and (e). For example, NLTK would be fine. However, if you have a working smoothed_perplexity function from part (c) then you should be able to complete these parts of the lab without relying on a third-party package.
- You are not required to case-normalize text. But if you do, you may get different (possibly better) results.
- Characters vs. Words. For parts (d) and (e) you may (but are not required) to use word-based models instead of, or in addition to character-based models. A small amount of extra credit is available if you do so.

  - 1 point: Extra credit for comparing both word and character LMs on at least one of (d) or (e)

There are 40 possible (regular) points for this lab and 1 available extra credit point. You should submit (a) results and output for each section, and (b) your source code, all together as one PDF file. You do not need to provide a file of your predictions for any task. Including your name prominently on the first page is very helpful, as is including a very brief synopsis of your work (what tools you used, any special processing, what works or does not work, any key observations from the lab).