IDS566: Advanced Text Analytics *Updated: 9/16/2022*

Mini-Project #1

Instructor: Moontae Lee *Total Points: 200*

Policy

1. You are allowed to work as a group of up to three or four students.
2. Having wider discussions is not prohibited. Put all the names of students that you discuss with beyond your group members. However individual groups must write their own solutions.
3. Put your write-up and results from the coding questions in a single pdf file. Compress Python source codes into one zip file. Each student/group must submit only two files. (Your solution will not be graded if the answers for coding questions are not included the pdf report)
4. If you would include some graphs, be sure to include the source codes together that were used to generate those figures. Every result must be reproducible.
5. Maximally leverage Piazza to benefit other students by your questions and answers. Try to be updated by checking notifications in both Piazza and the class webpage.
6. Late submissions will be penalized 20% per each late day. No assignment will be accepted more than 3 days after its due date. For Mini-Project 1, it will be due by 9/30/2022 11:59am.

Problem 1: Language Modeling [100 points]

This part is an open-ended programming assignment in which you are to implement a collection of N-gram language models on the brown corpus.[1]. Divide the dataset into three parts: training (70%), developing (10%), and test data (20%) by sequentially allocating first 40138 sentences to D_{train}, next 5734 sentences to D_{val}, and all remaining ones to D_{test}. You should write every part of your code by yourself without using any language modeling library or functions. Python 3 is the default language for this assignment.

(a) Write a code from the scratch that learns unigram and bigram models on the training data as Python dictionaries. Report the perplexity of your unigram and bigram models on the both **training data** and **test data**.

[1] from nltk.corpus import brown

(b) Implement add-λ smoothing method. With varying λ values. Draw a curve that measures your perplexity change over different λ values on the **developing data**.

(c) Pick the best λ value(s) and train again your unigram and bigram models on **training data + developing data**. Report new perplexity of your unigram and bigram models on the **test data**.

(d) Generate random sentences based on the unigram and bigram language models from part (c). Report 5 sentences per model by sampling words from each model continuously until meeting the stop symbol $\langle /s \rangle$.

(e) Choose at least one additional extension to implement. The available options are trigram, Good-Turing smoothing, interpolation method, and creative handling of unknown words. Verify quantitative improvement by measuring 1) the perplexity on test data; and qualitative improvement by retrying 2) the random sentence generation in part (d).

Problem 2: Application [100 points]

Pick your open-ended project from one of the following three applications:

- Spell-checking
- Auto-complete

Brainstorm first the interface design. It could be with or without context. Also, it could be in the middle of typing or after finishing the typing. Your appoarch must be different based on which interface that your group adopts. Relying on your design decision, you could excitingly combine the ideas of character n-grams and word n-grams.