IDS 566 Mini Project 1

Problem 1: Language Modeling

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 both training data and test data.

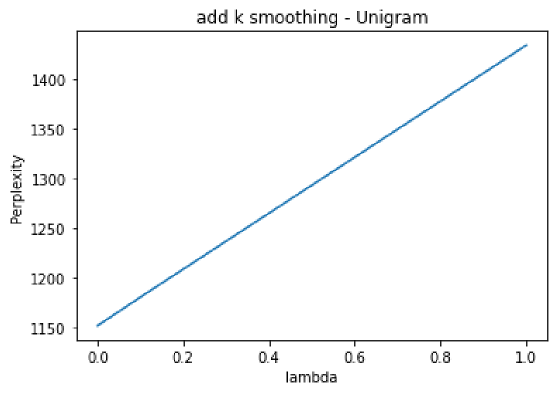
The perplexity of bigram is smaller than unigram, which means unseen data is more surprising to unigram.

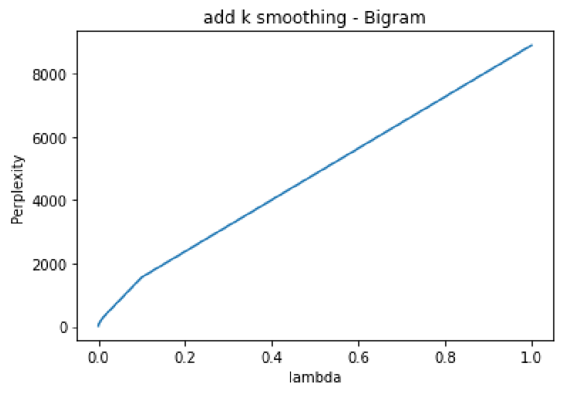
- The perplexity of the unigram model on training data: 1291.0674052557827
- The perplexity of the unigram model on test data: 831.1539904025091
- The perplexity of the unigram model on training data: 79.86967038783126
- The perplexity of the bigram model on test data: 42.10487732106337

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.

The graphs show the perplexity increase with the increase of lambdas. The reason is that unseen words steal away the mass too much.

- λ = [0.0000001,0.000001,0.00001,0.0001,0.001,0.01,0.1,1]





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.**

   Unigram minimum perplexity is 1151.880052947906 with λ value 1e-07, and the Bigram minimum perplexity is 30.861610145695586 with λ value 1e-07. Based on the best λ value, we then train again our unigram and bigram models on training data + developing data. The results are:
   - The Unigram Train + Develop Data perplexity is: 1335.4415053604012
   - The Unigram Test Data perplexity is: 831.1540021462384
   - The Unigram Train + Develop Data perplexity is: 83.14523688369151
   - The Unigram Test Data perplexity is: 42.12102592419681

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 ⟨/s⟩.**

   After reading the sentence generated for both unigram and bigram, we could find that sentences in bigram are more readable.
   Random five sentences based on unigram:
   - '<s> in present </s>'
   - '<s> to nothing share or interest a certain the handsome effectiveness in </s>'
   - '<s> the his time interesting prevent </s>'
   - '<s> write more he is reassurance <s> perhaps characteristics so which building been and <s> and and their is by face consists what this there </s>'
   - '<s> single level into time roads also <s> church know if of is to church land rule them his hutchinson people and jump case upon them at knowing laboratories administration and played <s> life with folk for speak </s>'

   Random five sentences based on bigram:
   - '<s> its name of males by pentagon and jackanapes </s>'
   - '<s> so he had the mild activity which any financial aspects however at the obviously steeles efforts to keep within </s>'
   - '<s> all other three others </s>'

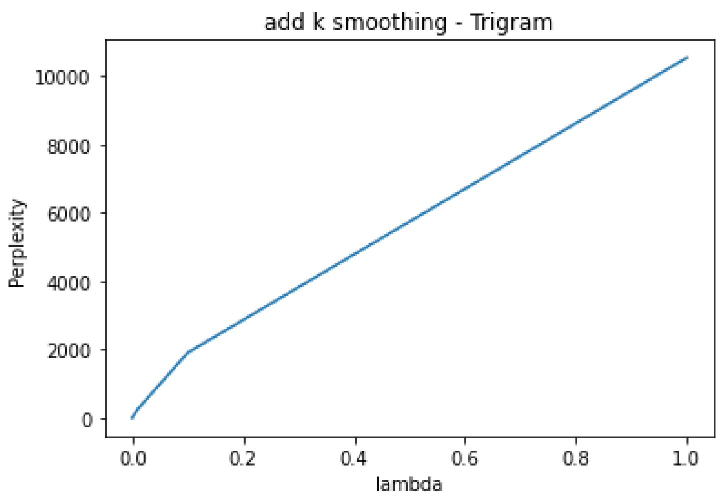
- '<s> i have failed them to linger under intensive application of foods helps you and denied due to go outside books that the slaves who made no one thousand mark had more tolerant or in </s>'
- '<s> jackson mississippi and the government research on where the memory serves </s>'

E. **Choose at least one additional extension to implement. The available options are tri- gram, Good-Turing smoothing, interpolation method, and creative handling of unknown words. Verify quantitative improvement by measuring**

Compared to bigram, trigram's perplexity is lower with the same lambda input**.**

a. the perplexity of test data

- Trigram minimum perplexity is 3.0319548430274548 with λ value 1e-07.
- The Trigram Train + Develop Data perplexity is: 5.815267268511102
- The Trigram Test Data perplexity is: 4.03054224638238



b. Qualitative improvement by retrying the random sentence generation in part (d).

The sentences in trigram are more reasonable than in bigram.
Random five sentences based on trigram:

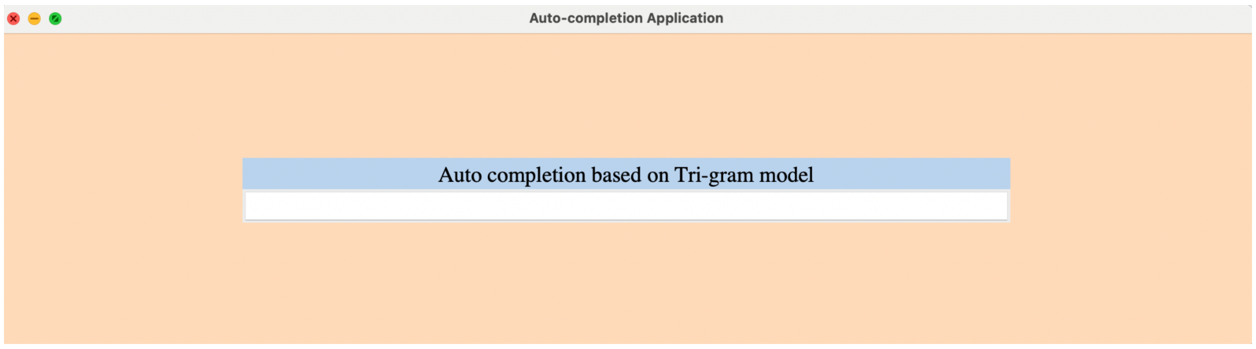
- '<s> the purified product was stored at lower overall costs than freezing or without the decay of capitalism is genuinely masterful </s>'
- '<s> <s> gorton and company however promptly bought land at pawtuxet on the reader one in sight </s>'
- '<s> <s> in addition the motor pool along with drying oils to carry the direct thinking zennist was a clean handkerchief through the hotels service entrance in the northeast some tips for shooting in the opening scene </s>'
- '<s> <s> the last tread mama would enjoy the full life promised for 1980 they and their sum can be effective negotiation with the exception of professional standards and an equal opposite force af against load l under actual cutting conditions is new york franchise is headed by harold arlen and mercer found a parking place half a century of exploration had established this democratic procedure in bw for an antique effect on the study is valuable and finally a balance must be known and loved living together </s>'
- '<s> upton possessed intellectual power ample means and made an alliance of sovereign nation states </s>'

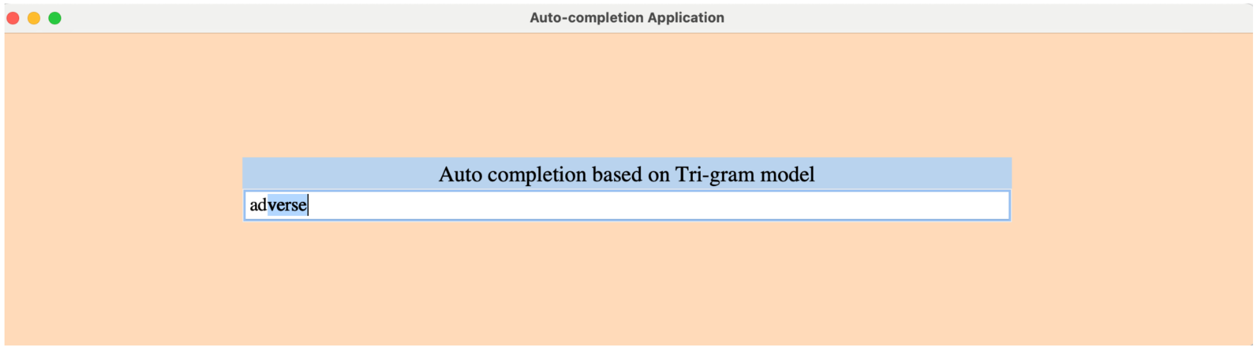
Problem 2: Application

Auto-complete

Team 6 decided to build an auto-completion model and has chosen to make a GUI application in python with the help of the Tkinter package. Our team split the design into the following steps:

1. Window application for users:


Auto-completion Application
Auto completion based on Tri-gram model


Auto-completion Application
Auto completion based on Tri-gram model
adverse

2. Concepts of our application:

   a. Language Models: Auto-Complete using the word N-gram language model
      We use word-based n-gram models to build an auto-completion program, and the dataset used for this program is an extension of the previous problem based on the brown corpus. Essentially, the n-gram model is a probabilistic model for predicting the next item in a sequence of items. According to the application, the items can be letters, words, phrases, or any linguistic entity. In short, the n-gram model is word-based and used for predicting the next word in a particular sequence.
   b. Preprocess data and calculate the n-gram count for the entire brown corpus
   c. Develop smoothed N-gram language to predict the next word given the context
   d. Find out the probability over all the words in the vocabulary
   e. Build up the auto-complete system by returning the following word's probability
   f. Get suggestions from looping over n-gram models

3. Steps of building the application:

   a. Basic Setup - import packages
      To implement the application, our team installs 'ttkwidgets' and 'pillow' libraries, and we will import the AutocompleteEntry method from "ttkwidgets.autocomplete" functionality. Then, We create an Entry widget using AutocompleteEntry and supply a parameter for completevalues of it rather than just creating an Entry widget using Tkinter. Then, when the user enters similar keywords, the completevalues parameter's values turn into options that will be completed. For example, completevalues = ['Dog', 'Cat'], now every time if 'd' will be typed, the

user will see autocomplete suggestion as ‘Dog,’ similarly in the case of ‘C’ the user will see ‘Cat’ as an option.

b. Pre-Processing pipeline
c. Splitting into Train, Developing, and Test
d. Cleaning the Data
e. Creating a Closed Vocabulary

   One of the most essential steps in dealing with textual data is handling out-of-vocabulary words. This helps the model to handle words which are not present in the training corpus.

f. Adding UNK Tokens

   We add <unk> tokens, to those words which are not in the closed vocabulary.

g. Building The "Model"
h. The Auto-Complete System

   Our team uses an instance of the Tkinter frame, i.e., Tk (), to build an application. We will keep the geometry of the Tkinter frame to “1200x300” to get a large view. We will define the name of the Tkinter frame, i.e., Auto-completion Application. We will select the hexadecimal color peachpuff1 as a background color for the application to have proper readability of text. We will choose the font size as 21 and 18 for the label in the label widget and for the text entered in the entry widget. The font type used will be “Times”. The width for the entry widget will be kept at 80.