CS 535 - Natural Language Processing

Due Date: Wednesday, October 7th by 11:55pm..

Assignments are to be done individually. No late assignments will be accepted.
Submissions that do not comply with the specifications given in this document will not be marked and a zero grade will be assigned.
Write your name and e-mail id in a comment line in on top of each source file. You are required to submit a single zip file containing an archive of your documentation and ipython notebook on Google Classroom. you should name your notebook as i19-XXXX.ipynb where i19-XXXX represents your student id.

Poetry Generation in Urdu

1 Introduction

In this assignment, you will use *n*-gram language modeling to generate some poetry using the **spaCy** library for text processing. For the purpose of this assignment a poem will consist of three stanzas each containing four verses where each verse consists of 3—7 words. For example, following is a manually generated stanza.

،دل سے نکال یاس کہ زندہ ہوں میں ابھی
،ہوتا ھے کیوں اداس کہ زندہ ہوں میں ابھی
،مایوسیوں کی قید سے خود کو نکال کر
،آ جاؤ میرے پاس کہ زندہ ہوں میں ابھی

،آ کر کبھی تو دید سے سیراب کر مجھے
،مرتی نہیں ھے پیاس کہ زندہ ہوں میں ابھی
،مہر و وفا خلوص و محبت گداز دل
،سب کچھ ھے میرے پاس کہ زندہ ہوں میں ابھی

،لوٹیں گے تیرے آتے ھی پھر دن بہار کے
،رھتی ھے دل میں آس کہ زندہ ہوں میں
،نایابؔ شاخ چشم میں کھلتے ھیں اب بھی خواب
سچ ھے ترا قیاس کہ زندہ ہوں میں ابھی

The task is to print three such stanzas with an empty line in between. The generational model can be trained on the provided Poetry Corpus containing poems from Faiz, Ghalib and Iqbal. You will train unigram, bigram and trigram models using this corpus. These models will be used to generate poetry. Several online solutions are available for English that use the NLTK library. However, we will be using the spaCy library to accomplish this task!

2 Assignment Task

The task is to generate a poem using different models. We will generate a poem verse by verse until all stanzas have been generated. The poetry generation problem can be solved using the following algorithm:

- Load the Poetry Corpus
- Tokenize the corpus in order to split it into a list of words
- Generate *n*-gram models
- For each of the stanzas
  - For each verse
    * Generate a random number in the range [3...7]
    * Select first word
    * Select subsequent words until end of verse
    * **[bonus]** If not the first verse, try to rhyme the last word with the last word of the previous verse * Print verse
  - Print empty line after stanza

2.1 Implementation Challenges

Among the challenges of solving this assignment will be the selection of subsequent words once we have chosen the first word of the verse. But, to predict the next word, what we want to compute is, what is the most likely next word out of all of the possible next words? In other words, find the word that occurred the most often after the condition in the corpus. We can use a Conditional Frequency Distribution (CFD) to figure that out! A CFD tells us: given a condition, what is likelihood of each possible outcome. [bonus] Rhyming verses is also a challenge. You can built your dictionary for rhyming.

The Urdu sentence is written from **right to left**, so makes your *n*-gram models according to this style.

2.2 Standard *n*-gram Models

We can develop our model using the Conditional Frequency Distribution method. First develop a unigram model (Unigram Model) and then the bigram model (Bigram Model). Select the first word of each line randomly from the high frequency words in the vocabulary and then use the bigram model to generate the next word until the verse is complete. Generate the next three lines similarly. Follow the same steps for the trigram model and compare the results of the two n-gram models. **[bonus]** Can we make the sonnet rhyme? (Hint: Built a pronunciation dictionary)

2.3 Backward Bigram Model

Standard *n*-gram language models model the generation of text from left to right. However, in some cases, tokens might be better predicted from their right context rather than their left context. The next task is to produce a backward bigram model that models the generation of a sentence from right to left. Think of a very simple way to very quickly using BigramModel to produce a BackwardBigramModel that is identical except for the modeling direction. Compare the results of the backward bigram model with previous implementations.

2.4 Bidirectional Bigram Model

Next, build a BidirectionalBigramModel that combines the forward and backward model. Both the **Backward BigramModel** and **BidirectionalBigramModel** should take the same input and produce the same style of output as BigramModel. Compare the output with the previous models.

Honor Policy

This assignment is a learning opportunity that will be evaluated based on your ability to think in a group setting, work through a problem in a logical manner and write a research report on your own. You may however discuss verbally or via email the assignment with your classmates or the course instructor, but you are to write the actual report for this assignment without copying or plagiarizing the work of others. You may use the Internet to do your research, but the written work should be your own. **Plagiarized reports or code will get a zero**. If in doubt, ask the course instructor.