**CS 591 Homework 3**

Name: Amal Almuteb

Number of attendance sheet: 1

**22.1 This exercise explores the quality of the n-gram model of language. Find or create a monolingual corpus of 100,000 words or more. Segment it into words, and compute the fre-quency of each word. How many distinct words are there? Also count frequencies of bigrams (two consecutive words) and trigrams (three consecutive words). Now use those frequencies to generate language: from the unigram, bigram, and trigram models, in turn, generate a 100- word text by making random choices according to the frequency counts. Compare the three generated texts with actual language. Finally, calculate the perplexity of each model.**

**Solution:**

Before I started, I installed NLTK and then I installed the *NLTK Book Collection*

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

Once the data is downloaded to my machine, now I can load some of it using the Python interpreter and here I used **jupyter notebook.**

For this exercise I choose the **Brown Corpus** from **Brown University** and it have been categorized by genre.

A screenshot of a cell phone

Description automatically generated

The category I have chosen to use for this exercise is (categories='learned') which is contain **181888** words.

A picture containing screenshot

Description automatically generated

* **To segment it into words and compute the frequency of each word**

A screenshot of a cell phone

Description automatically generated

* **For the question” How many distinct words are there? “ I used the function fdist.N() to show the result**

A screenshot of a cell phone

Description automatically generated

**Now, let’s suppose if I have sample of words and want to count the frequency of each**

A screenshot of a cell phone

Description automatically generated

* **To count frequencies of bigrams (two consecutive words) :**

A screenshot of a social media post

Description automatically generated

* **To count frequencies of trigrams (three consecutive words) :**

A screenshot of a social media post

Description automatically generated

* **Now use those frequencies to generate language: from the unigram, bigram, and trigram models, in turn, generate a 100- word text by making random choices according to the frequency counts:**

1. **Generate 100 words from the unigram**

A screenshot of a cell phone

Description automatically generated

**The text generated by using the unigrams frequency:**

The incomplete septation, new growth have something more exclusive reception for wetting. Of Poesy before reversing majority the most Rhode Islanders were extracted with water and illogical comparisons . ) and reviewing stand at no apparent prospect of incompletely developed marked with changes , is , namely Af , to marry the court ( thus confused with an equal or electronic switches may have to the granular texture thus confused some extent that this unique but only one which there is uncluttered and the cortico-fugal discharges induced on a central measuring the attribute their

sense of desegregation

1. **Generate 100 words from the bigrams:**

A screenshot of a social media post

Description automatically generated

**The text generated by using the bigrams frequency:**

With completion ages and supplied to greet them successive locations of each sheet is expected to four years , acknowledged as in his admiration of a number of possibility is Hrothgar ) the same components . Besides the years , or system tended to learn that they had developed by fishing in a burrow has been experimentally verify this is chosen , than have survived by mechanical potential . We observe the level of semester of the aging properties are a full pollen and depth of this final analysis . For any one which could be anatomically more apt

1. **Generate 100 words from the trigram:**

A screenshot of a social media post

Description automatically generated

**The text generated by using the tigrams frequency:**

Introduction It has recently become practical to use the term alienation in two different ways . Sometimes he uses it as a living hero . The other had assumed the terrifying inertia of inanimate matter . Being with him was like being swept away by a torrent of falling stones . Now Dylan Thomas and Charlie Parker have a great deal of thought behind them . This added a personal zest to class discussions and participation . Both sexes reported that the discussions on sex adjustment within marriage were extremely enlightening . The writer began this special class by explaining his

* **Compare the three generated texts with actual language:**

After my reading all three texts. I found the text generated on a trigram-model is the most logical one, because it links the words into groups of three words. Unlike the text generated on a unigram-model is less significant. and relevant. While the text generated on a bigram-model is less quality than trigram text but better than the unigram model, but it still makes sense relatively.

* **Finally, calculate the perplexity of each model:**

1. **Unigrams perplexity**

A screenshot of a cell phone

Description automatically generated

1. **Bigram perplexity**

A screenshot of a cell phone

Description automatically generated

1. **Trigram perplexity**

A screenshot of a cell phone

Description automatically generated

**22-2 Write a program to do segmentation of words without spaces. Given a string, such as the URL “thelongestlistofthelongeststuffatthelongestdomainnameatlonglast.com,” return a list of component words: [“the,” “longest,” “list,” . . .]. This task is useful for parsing URLs, for spelling correction when words runtogether, and for languages such as Chinese that do not have spaces between words. It can be solved with a unigram or bigram word model and a dynamic programming algorithm similar to the Viterbi algorithm.**

A screenshot of a cell phone

Description automatically generated

**The code:**

def segment(text, segs):

words = []

last = 0

for i in range(len(segs)):

if segs[i] == '1':

words.append(text[last:i+1])

last = i+1

words.append(text[last:])

return words

text = "thelongestlistofthelongeststuffatthelongestdomainnameatlonglast.com"

seg1 = "0000000000000100000000000000001000000000000000001000000000000000000"

seg2 = "0010000001000101001000000100001010010000001000001000101000100011001"

print(segment(text, seg1))

print(segment(text, seg2))