Summary

We are creating a program that will transform the way books are published. A high-powered professional grammar program is the goal. It will revolutionize the way people read. The product will be user friendly. The goal of the program is to analyze individual books or a set of documents.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The program will use the results of ***matt\_wc.py*** and ***matt\_wc2.py.***

The program ***matt\_wc.py*** calculates unique words sorted by frequency.

This will be called the **Vocabulary List**.

The program ***matt\_wc2.py*** calculates phrases that appear more than once.

This will be called the **Phrase List**.

The Vocabulary List and the Phrase List will be input for the next phase of the program.

The program will ouput count of *letters* forthe entire document:

{‘a’: 75003, ‘b’: 16286, ‘c’: 20355, ‘d’: 34206, ‘e’: 110455, ‘f’: 16183, ‘g’: 22554, ‘h’: 47539, ‘i’: 63197, ‘j’: 2136, ‘k’: 12845, ‘l’: 40522, ‘m’: 28624, ‘n’: 59483, ‘o’: 83633, ‘p’: 14940, ‘q’: 724, ‘r’: 48730, ‘s’: 55743, ‘t’: 90398, ‘u’: 27667, ‘v’: 8373, ‘w’: 23594, ‘x’: 1626, ‘y’: 21810, ‘z’: 557}

This is known as ***ActualNumberOfLetterCountsForTheEntierDocument.py***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Next Steps**:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Top 20 List** is the vocabulary list created in this program not to be confused with the Vocabulary List used as input. The Top 20 List is determined on a case-by case basis. That is for each phrase.

A ***Bank*** is created with the Letter Frequency Chart.

A ***Pool*** will be created with each phrase on the Phrase List.

This will create so many ***Pools***.

The program will determine the top 20 unique words for each phrase analyzed sorted by frequency ( Most frequent to least frequent.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Methods**

Import data from matt\_wc.py and matt\_wc2.py.

Calculate the letter frequencies in Typoglycemia.docx.

Given the letter frequencies you count how many times you may make a phrase given the Bank of Letters.

This will create a Pool.

There will be many Pools as there are many phrases.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

With each Pool you will analyze the vocabulary. The number of times you may make a word will determine the vocabulary list for each pool. The Pool will reset for each word checked.

You may do this by creating a list of letter counts for each word. Then you just multiply the letter counts until it fills the Pool. This will determine the frequency of the word in the Pool. Reset the Pool value after you check each word.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

After the Pools’ vocabulary is determined the top 20 phrases will be extracted. This will be done for each Pool. The whole list will be consolidated. The word frequency for each Pool will be combined and added together if there is a duplicate word. The list should not exceed 500 pages. If the list is longer I will adjust the requirements of the program, perhaps selecting only the top 10 words for each Pool.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Output**

A list of vocabulary will be created. The list will be sorted by frequency (Most frequent to least frequent.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Other Programs I Am Considering Or You Have Already Written:**

Combinations, Dictionary, Letter Count, Magic Mirror, Phrase, Sentence, Typoglycemia and Word. The Word programs will include: adjectives, adverbs, conjunctions, gerund, nouns, participles, prepositions, pronouns, typoglycemia, verbs, word frequency and word length.

**Programs**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Combinations**

*Program 1*

1. Enter key words.

2. Check each sentence in a file.

3. If a sentence contains all key words then it is a match.

4. The first order is words in sequence.

5. The second order is words not in sequence.

6. Output matches as whole sentences.

7. Below each sentence output key words in question.

*Program 2*

This program will list all of the combinations of 6! from largest to smallest.

**Dictionary**

This program will identify all words used in the document with their part(s) of speech.

**Letter Count**

***ActualNumberOfLetterCountsForTheEntierDocument.py***

**Magic Mirror**

This program will check all the combinations and verify if the same letters are being used in sentences with different arrangements.

***matt\_wc3.py***

**Phrase**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Program 1*

**Infinitive**

This program will identify infinities used in a document.

An infinitive is the form of the verb with to followed by the verb.

you may substitute with a noun, adjective or adverb.

Example,

To know, to see, to go, to shop, to be...

*Program 2*

**Phrasing**

This program will identify all of the phrases used in a book.

Right now the program is not working properly**.**

***matt\_wc2.py***

**Sentence**

This program will list all sentences in a book in alphabetical order.

**Typoglycemia**

This program will list a Typoglycemic version.

The words for now will be first and last letter the same and the

middle letters in alphabetical order.

**Word Programs:**

**Adjectives**

Adjectives modify verbs, adjectives or their substitutes.

example

red, yellow, blue, short, tall, big, good...

**Adverbs**

This program will list all words ending in -ly.

**Conjunctions**

This program will identify all sentences which use a conjunction.

Example:

And, either, since, but...

**Nouns**

This program will identify all nouns.

A noun is a person, a place or a thing.

**Gerund**

This program will identify gerunds.

A gerund is a word ending in -ing, that is a noun.

**Participles**

This program will identify participles.

Participles are words that usually end

in -ing, -ed, -en, that is used as an adjective.

**Prepositions**

This program will identify all the prepositions given in a document.

**Pronouns**

The following 7 programs will list the sentences in the document that contain each of the 7 pronouns.

**He**

This program will identify all sentences using the word he.

**I**

This program will identify all sentences using the word I.

**It**

This program will identify all sentences using the word it.

**She**

This program will identify all sentences using the word she.

**They**

This program will identify all sentences using the word they.

**We**

This program will identify all sentences using the word we.

**You**

This program will identify all sentences using the word you.

**Typoglycemia**

This program will rearrange the letter in words keeping

the first and last letter the same and putting the middle

letters in alphabetical order in words longer than three

letters and words that contain identical middle letters.

**Verbs**

This program will identify all verbs.

**Word Frequency**

This program will list the complete dictionary of words used in a document.

The words or the vocabulary will be listed in order of frequency.

***matt\_wc.py***

**Word Length**

The program called Word Length will sort all the words

from longest word to shortest word. The data will

include number of letters in each word.

**Example:**

**One letter words.**

**i, a**

**Two letter words.**

**By, at, in, on, ad, am, an, as, be, so...**

**Three letter words.**

**act, sit, was, has...**

**Four letter words.**

**here, that, this, good...**

**Five letter words.**

**drink, seven, knows, ...**

**Six letter words.**

**fixing, letter...**

**Seven letter words.**

**washing, letters...**

**Eight letter words.**

**finances, rosemary...**

**Nine letter words.**

**Imagining…**

**Ten letter words.**

**Eleven letter words.**

**Twelve letter words.**

**Thirteen letter words.**

**Fourteen letter words.**

**Fifteen letter words.**

**Sixteen letter words.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Things I need to do**

Try to find a way to incorporate all the data collected and add to the program. Help develop a method to help run the program. Provide specifications which are manageable. For example we would not want the program to run too long. Create a Typoglycemic version of the file. Review files such as the output for the Word Frequency and Phrase Frequency lists. Decide how I may incorporate the star data into the program. Label each word in my book with the part(s) of speech. Study vectors.

**Things you need to do**

Allow for modifications. Program the program. Keep in contact by email. Choose a method I may understand. Such as how big the data is. Include suggestions. Include problems. Reply to all emails as needed or reply as time allows you to in your schedule. Create a List of Top 20 Vocabulary for each Pool analysis. Let me know how the computer program works in ordinary terms. Give thought to new program additions.

**Some Questions**

How does your program work? The definition of phrase is? It appears that a pronoun is not included in the definition of phrase. Should we be considering phrases that appear only once? Should we consider changing the definition of phrase? Yes. Should you consider taking advantage of the ability to distinguish between lower and upper case letters? How may we incorporate the star data? Too much data? What are 26-dimensional vector values?

**Goals**

The goal is to narrow down the solution. The solution will be equal letter counts for Old Book to New Book. By studying Grammar we may create limits to the number of word combinations to be found in the text. We will be able to limit the amount of processing the program has to do by creating many files in different orders. Structuring the code properly is crucial.