Social Sphere Machine:

Document Classification

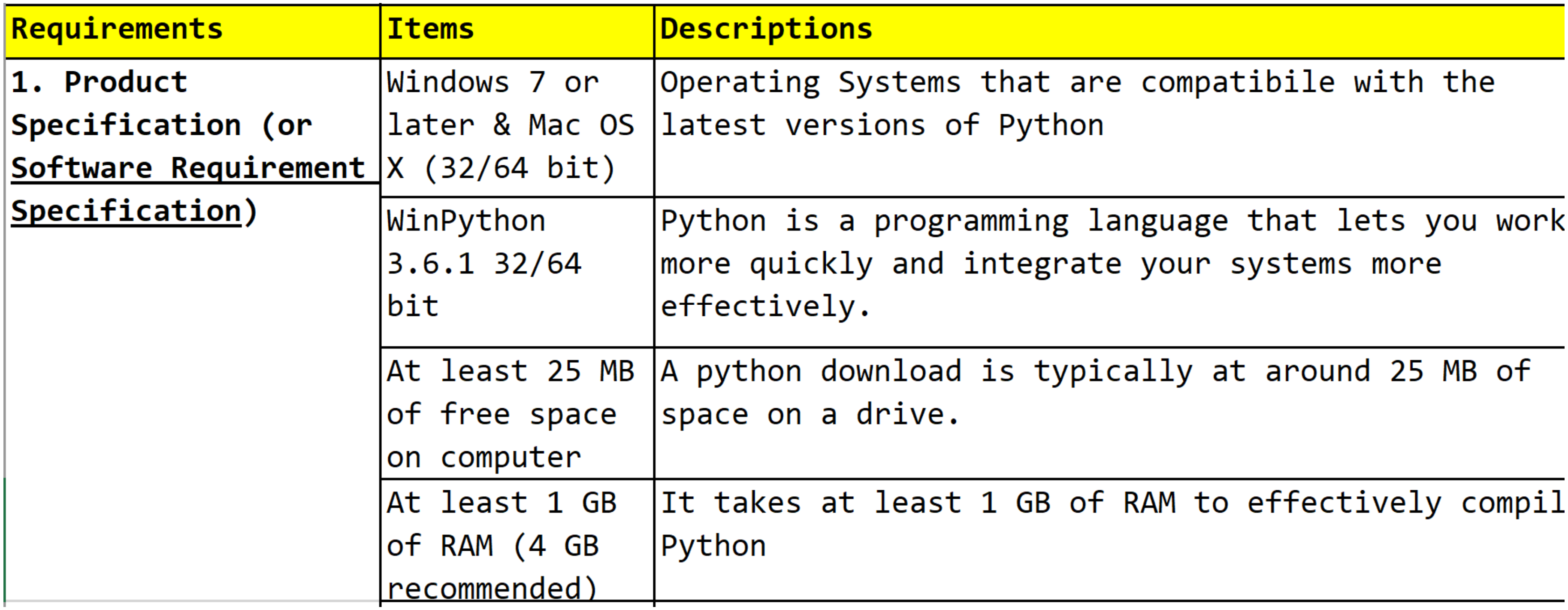
By: Jedidiah Agbenu, Nana-Akua Ofosu

Advisor: Dr. Charles Kim

4/20/2021

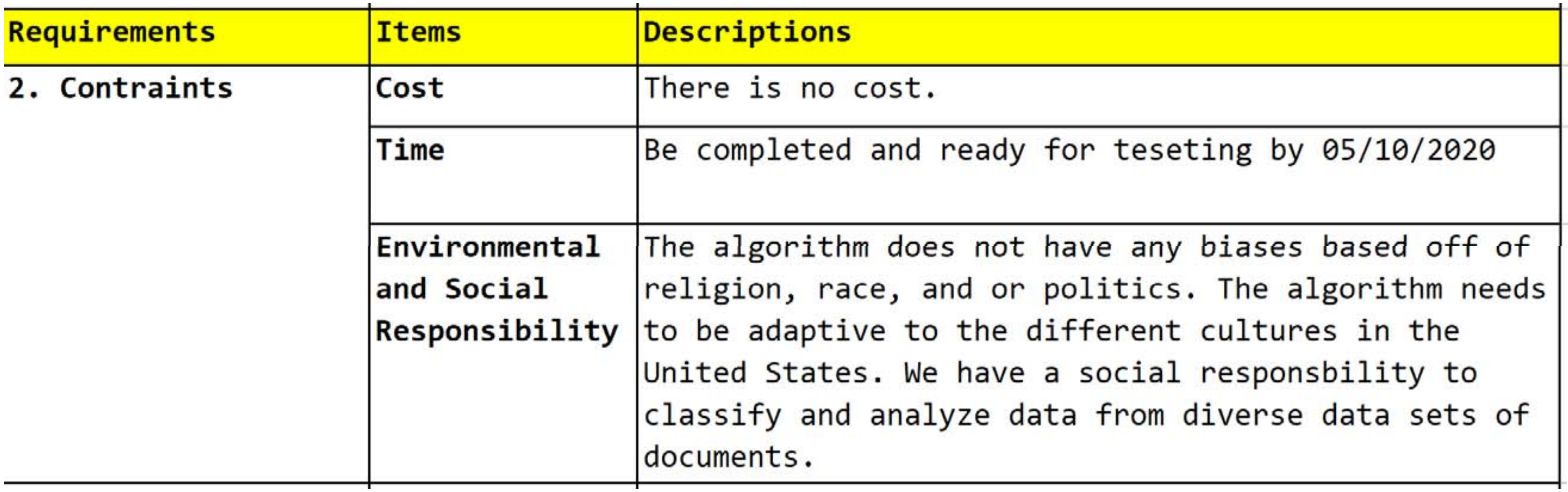
Problem Statement

- We need a systematic way to classify documents and detect fake documents reduces the amount/number of untruthful articles circulating to the general public; therefore, preventing misinformation from misleading public opinion, increasing the accuracy of determining whether a document is fake or not, and increasing the speed of identifying fake documents.



Requirements	Items	Descriptions
1. Product Specification (or Software Requirement Specification)	Windows 7 or later & Mac OS X (32/64 bit)	Operating Systems that are compatibile with the latest versions of Python
	WinPython 3.6.1 32/64 bit	Python is a programming language that lets you work more quickly and integrate your systems more effectively.
	At least 25 MB of free space on computer	A python download is typically at around 25 MB of space on a drive.
	At least 1 GB of RAM (4 GB recommended)	It takes at least 1 GB of RAM to effectively compil Python

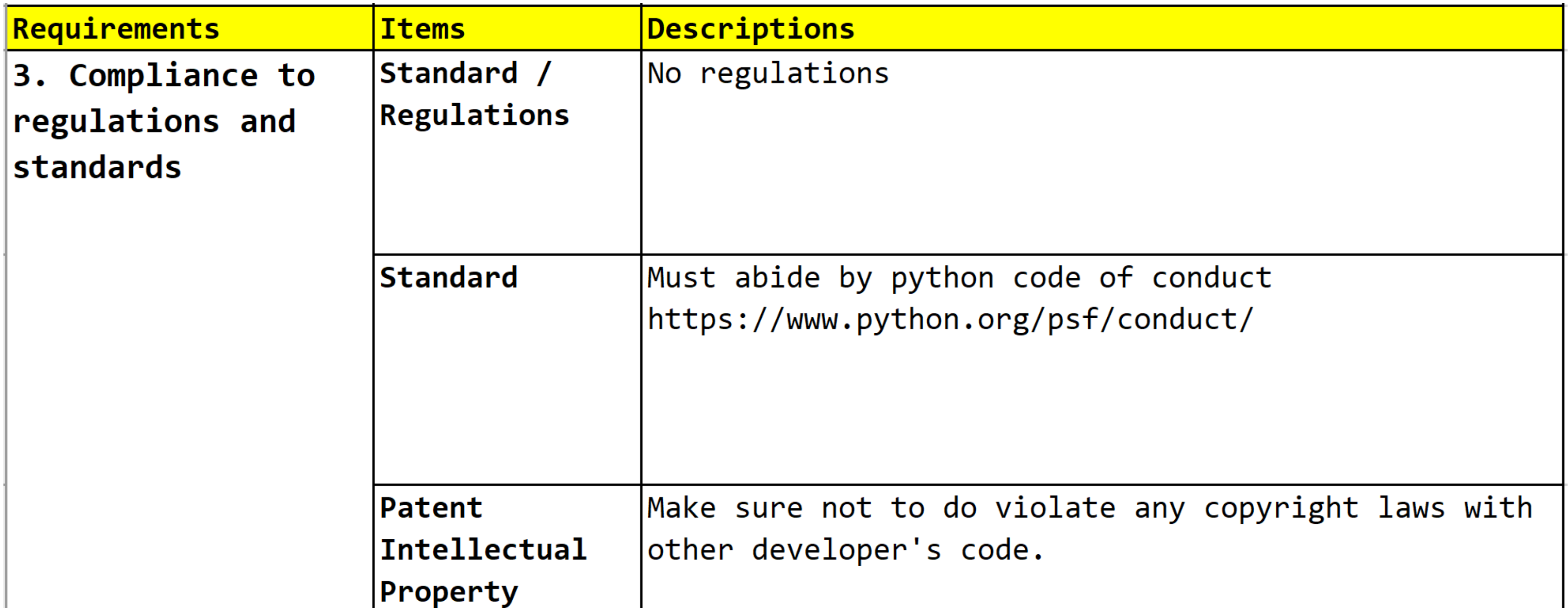
Design Requirements: Product



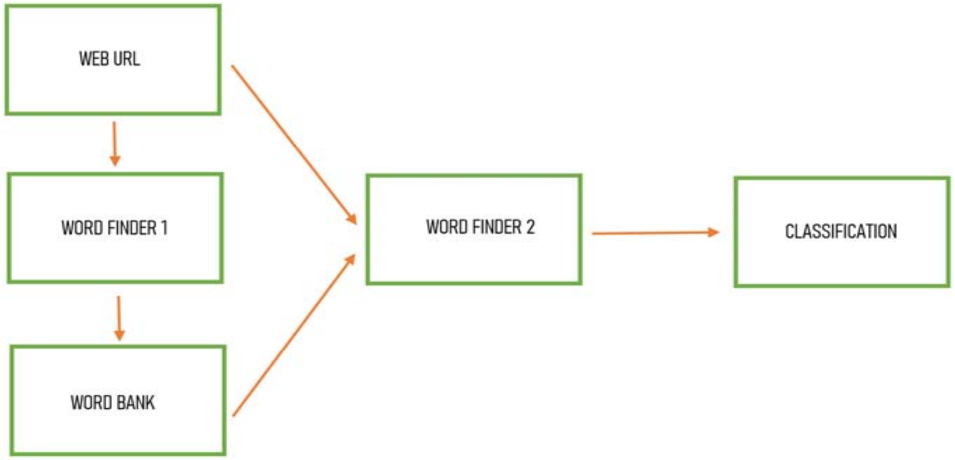
Requirements	Items	Descriptions
2. Contraints	**Cost**	There is no cost.
	Time	Be completed and ready for teseting by 05/10/2020
	Environmental and Social Responsibility	The algorithm does not have any biases based off of religion, race, and or politics. The algorithm needs to be adaptive to the different cultures in the United States. We have a social responsbility to classify and analyze data from diverse data sets of documents.

Design Requirements: Constraints

Design Requirements: Compliance to standards



Requirements	Items	Descriptions
3. Compliance to regulations and standards	**Standard / Regulations**	No regulations
	Standard	Must abide by python code of conduct https://www.python.org/psf/conduct/
	Patent Intellectual Property	Make sure not to do violate any copyright laws with other developer's code.



WEB URL

WORD FINDER 1

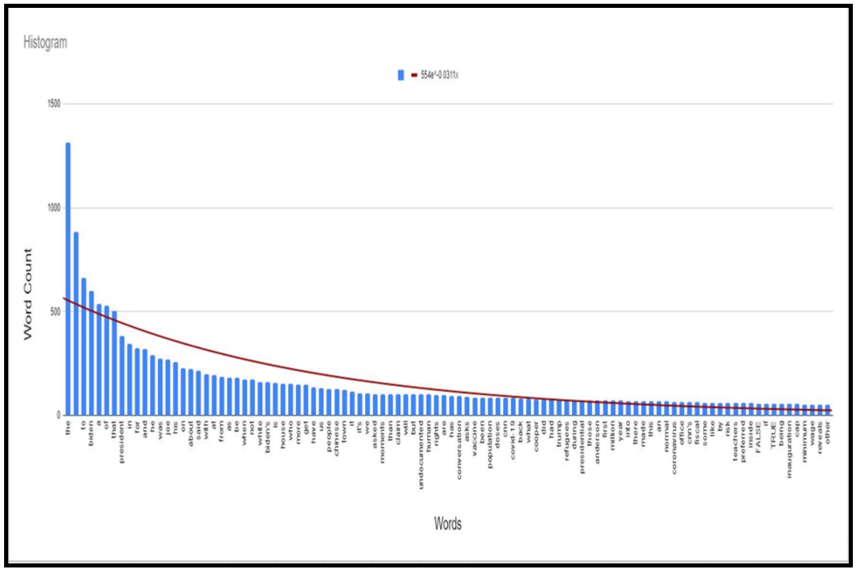
WORD BANK

WORD FINDER 2

CLASSIFICATION

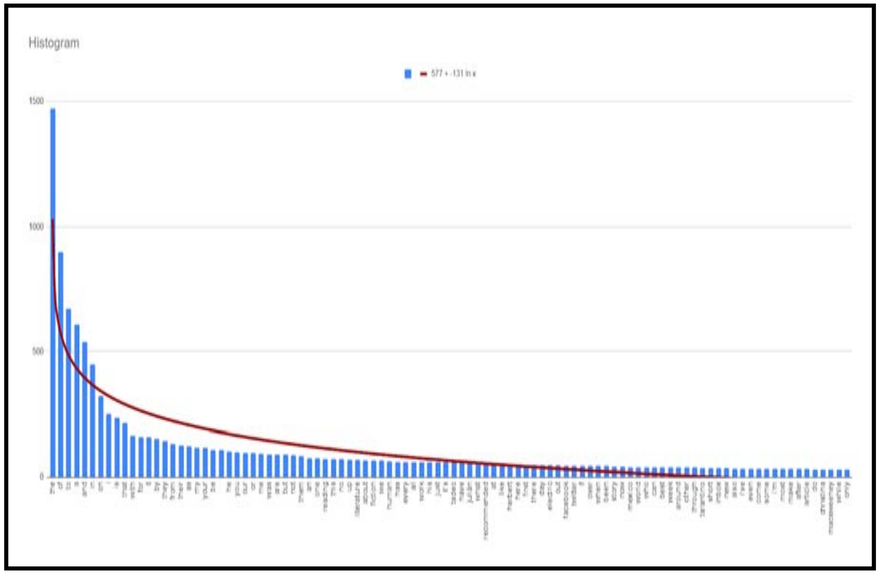
Final Solution Diagram

SPRINT 1



- ❑ Piece: Document Analyzation
- ❑ Week 1:Developed classes and criteria to place documents in
- ❑ Week2: Reviewed the Python word counter
- ❑ Went well: Graphed an exponential line of best fit based on documents word counts
- ❑ What was pivotal: It is important to remove the articles and change the equation

SPRINT 2



❑ Piece: Graph

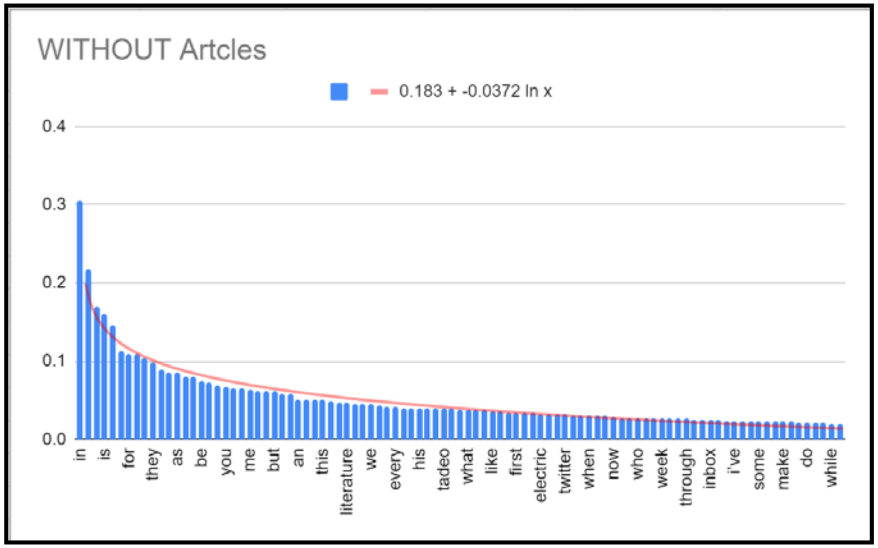
❑ Week 1: Focus on the classification variable

❑ Week2:Work on the graph that will be used for classifying documents

❑ Went well: Graphing with a logarithmic line of best fit is good

❑ What was pivotal: Find more classification variables

SPRINT 3



❑ Piece: Database

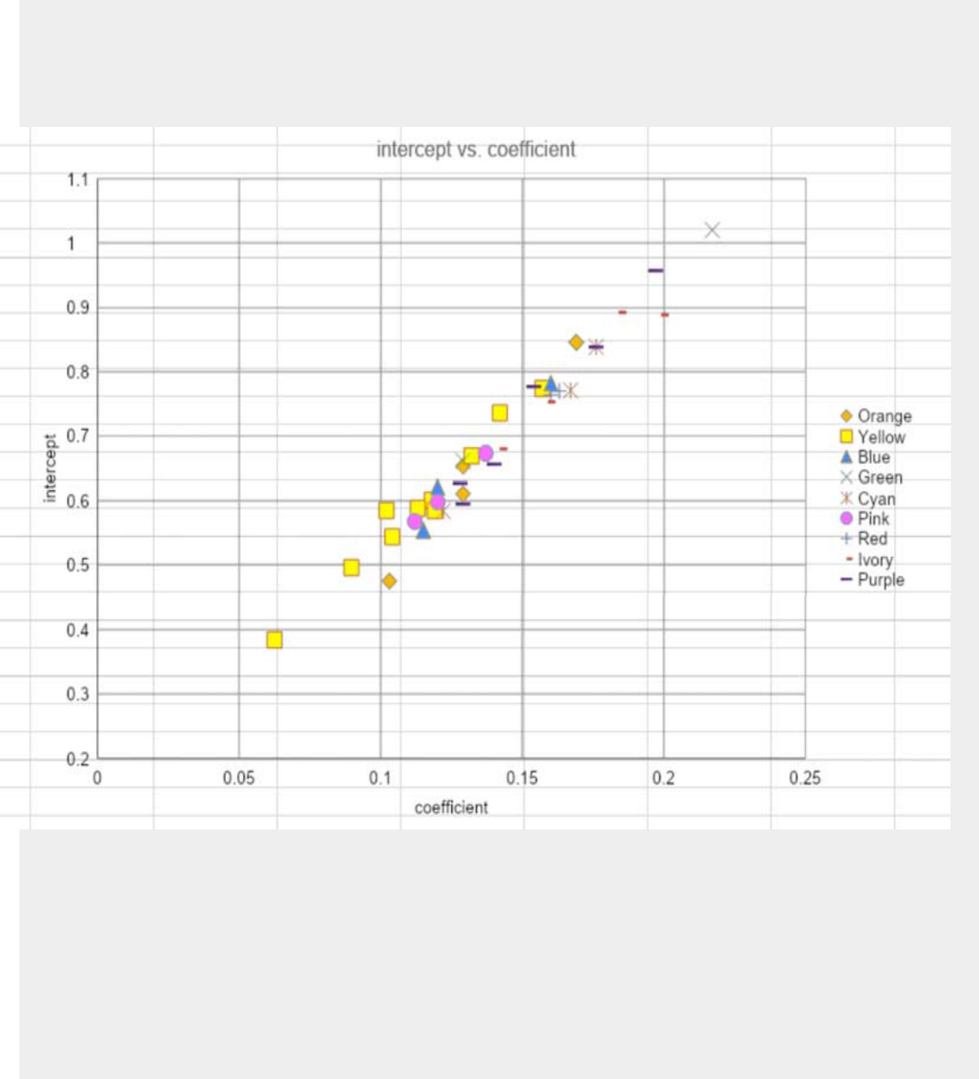
❑ Week 1:Normalize the functions created from the word count

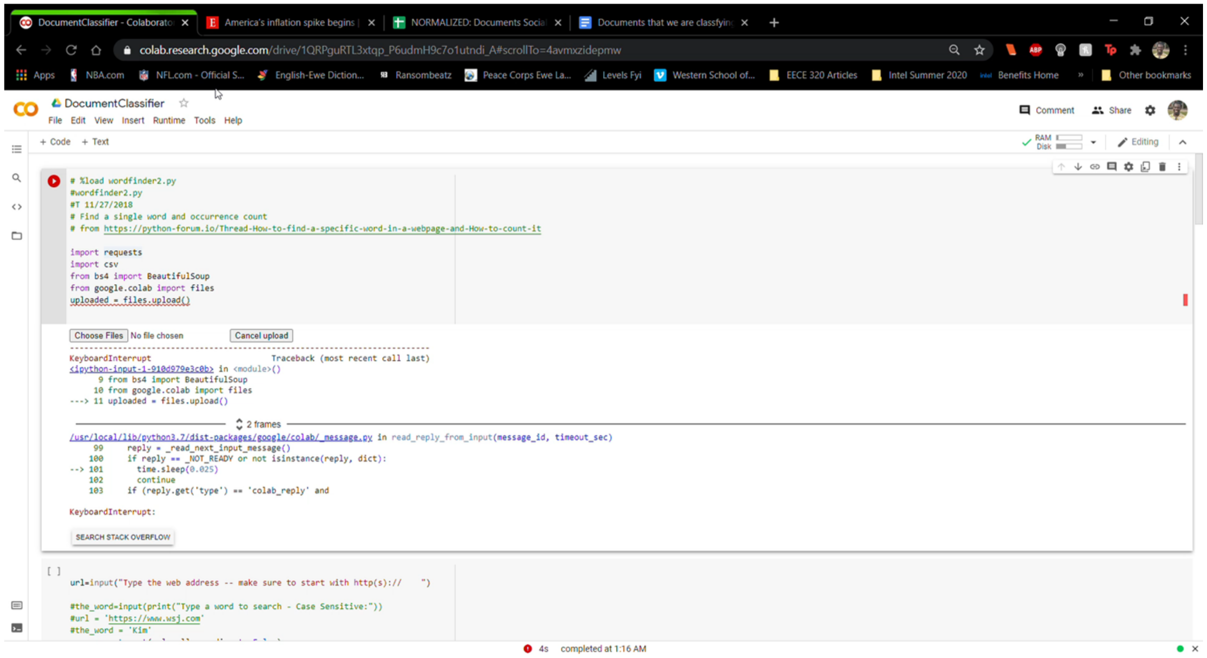
❑ Week2: Create the scatter plot based off of the coefficient of the word count formulas

❑ Went well: Normalized our logarithmic functions and simplified of the direction of the final product

❑ What was pivotal: We decided to take our project in a different direction

SPRINT 4


intercept vs. coefficient
intercept
coefficient
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
0.05
0.1
0.15
0.2
0.25
Orange
Yellow
Blue
Green
Cyan
Pink
Red
Ivory
Purple



```
# %load wordfinder2.py
#wordfinder2.py
#T 11/27/2018
# Find a single word and occurrence count
# from https://python-forum.io/Thread-How-to-find-a-specific-word-in-a-webpage-and-How-to-count-it

import requests
import csv
from bs4 import BeautifulSoup
from google.colab import files
uploaded = files.upload()
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-1-910d979e3c0b> in <module>()
      9 from bs4 import BeautifulSoup
     10 from google.colab import files
---> 11 uploaded = files.upload()

2 frames
/usr/local/lib/python3.7/dist-packages/google/colab/_message.py in read_reply_from_input(message_id, timeout_sec)
     99     reply = _read_next_input_message()
    100     if reply == _NOT_READY or not isinstance(reply, dict):
--> 101       time.sleep(0.025)
    102       continue
    103     if (reply.get('type') == 'colab_reply' and

KeyboardInterrupt:
```

```
url=input("Type the web address -- make sure to start with http(s)://    ")

#the_word=input(print("Type a word to search - Case Sensitive:"))
#url = 'https://www.wsj.com'
#the_word = 'Kim'
```

DEMO

Conclusion

HIGHLIGHTS & LOWLIGHTS

WE WERE ABLE TO FIND AN ALTERNATE SOLUTION TO CLASSIFY DOCUMENTS INDEPENDENT OF THE SCATTER PLOT

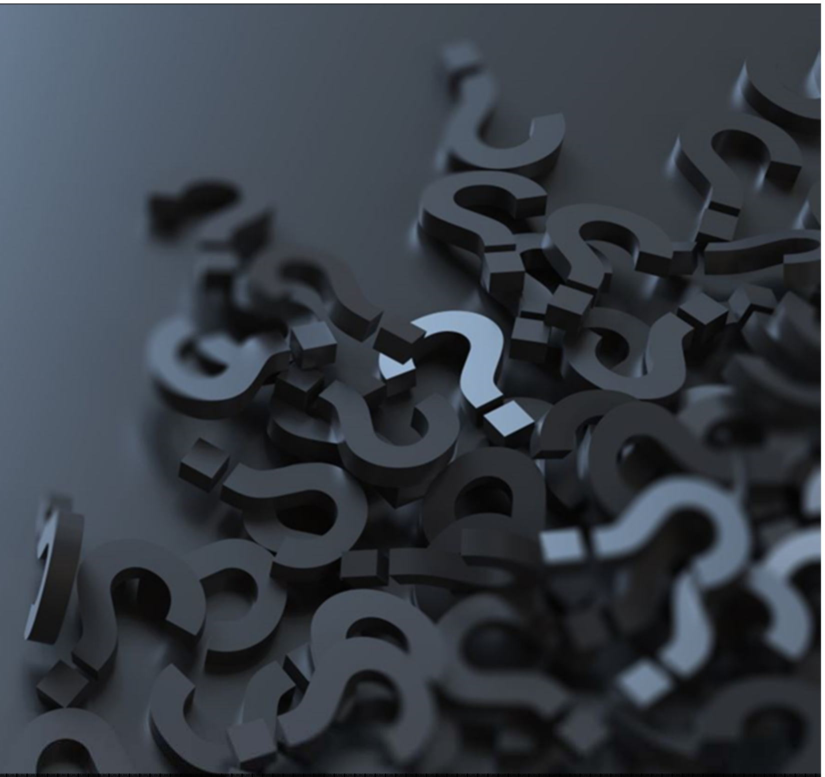
THIS METHOD CANNOT BE USED FOR MORE CATEGORIES YET, AS WE WILL HAVE TO TRAIN OUR MODEL TO RECOGNIZE WORDS FROM OTHER DOCUMENT CATEGORIES

Next Steps

To include more categories

Improve the word counter

Train the algorithm developed



Thank you! Any Questions?