```
\begin{center}
\begin{huge}
MCIS6273 Data Mining (Prof. Maull) / Fall 2023 / HW0
\end{huge}
\end{center}
```

Points Possible	Due Date	Time Commitment (estimated)
20	Tuesday September 26 @ Midnight	*up to* 4 hours

- **GRADING:** Grading will be aligned with the completeness of the objectives.
- **INDEPENDENT WORK:** Copying, cheating, plagiarism and academic dishonesty *are not tolerated* by University or course policy. Please see the syllabus for the full departmental and University statement on the academic code of honor.

OBJECTIVES

- Familiarize yourself with Github and basic git
- Familiarize yourself with the JupyterLab environment, Markdown and Python
- Explore JupyterHub Linux console integrating what you learned in the prior parts of this homework
- Listen to the Talk Python To Me from July 7, 2023: How data scientists use Python
- Perfom basic data engineering in Python using Gutenberg.org text of Bertrand Russell's 1912 work *The Problems of Philosophy*
- Use structured data to develop basic statistical analyses

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, make a directory in your Lab environment called `homework/hw0`. Put all of your files in that directory. Then zip that directory, rename it with your name as the first part of the filename (e.g. `maull_hw0_files.zip`), then download it to your local machine, then upload the `.zip` to Blackboard.

If you do not know how to do this, please ask, or visit one of the many tutorials out there on the basics of using zip in Linux.

If you choose not to use the provided notebook, you will still need to turn in a `.ipynb` Jupyter Notebook and corresponding files according to the instructions in this homework.

ASSIGNMENT TASKS

(0%) Familiarize yourself with Github and basic git

Github (https://github.com) is the *de facto* platform for open source software in the world based on the very popular git (https://git-scm.org) version control system. Git has a sophisticated set of tools for version control based on the concept of local repositories for fast commits and remote repositories only when collaboration and remote synchronization is necessary. Github enhances git by providing tools and online hosting of public and private repositories to encourage and promote sharing and collaboration. Github hosts some of the world's most widely used open source software.

If you are already familiar with git and Github, then this part will be very easy!

§ Task: Create a public Github repo named `"mcis6273-f23-datamining"` and place a readme.md file in it. Create your first file called `README.md` at the top level of the repository.

Please put your Zotero username in the file. Aside from that you can put whatever text you like in the file (If you like, use something like lorem ipsum to generate random sentences to place in the file.). Please include the link to **your** Github repository that now includes the minimal `README.md`. You don't have to have anything elaborate in that file or the repo.

§ Task: Fork the course repository:

- https://github.com/kmsaumcis/mcis6273_f23_datamining/

(10%) Familiarize yourself with the JupyterLab environment, Markdown and Python

As stated in the course announcement Jupyter (https://jupyter.org) is the core platform we will be using in this course and is a popular platform for data scientists around the world. We have a JupyterLab setup for this course so that we can operate in a cloud-hosted environment, free from some of the resource constraints of running Jupyter on your local machine (though you are free to set it up on your own and seek my advice if you desire).

You have been given the information about the Jupyter environment we have setup for our course, and the underlying Python environment will be using is the Anaconda (https://anaconda.com) distribution. It is not necessary for this assignment, but you are free to look at the multitude of packages installed with Anaconda, though we will not use the majority of them explicitly.

As you will soon find out, Notebooks are an incredibly effective way to mix code with narrative and you can create cells that are entirely code or entirely Markdown. Markdown (MD or `md`) is a highly readable text format that allows for easy documentation of text files, while allowing for HTML-based rendering of the text in a way that is style-independent.

We will be using Markdown frequently in this course, and you will learn that there are many different "flavors" or Markdown. We will only be using the basic flavor, but you will benefit from exploring the "Github flavored" Markdown, though you will not be responsible for using it in this course -- only the "basic" flavor. Please refer to the original course announcement about Markdown.

§ Task: THERE IS NOTHING TO TURN IN FOR THIS PART.

Play with and become familiar with the basic functions of the Lab environment given to you online in the course Blackboard.

§ Task: Please *create a markdown document* called `semester_goals.md` with 3 sentences/fragments that answer the following question:

- **What do you wish to accomplish this semester in Data Mining?**

Read the documentation for basic Markdown here. Turn in the text `.md` file *not* the processed `.html`. In whatever you turn in, you must show the use of *ALL* the following:

- headings (one level is fine),
- bullets,
- bold and italics

Again, the content of your document needs to address the question above and it should live in the top level directory of your assignment submission. This part will be graded but no points are awarded for your answer.

(0%) Explore JupyterHub Linux console integrating what you learned in the prior parts of this homework

The Linux console in JupyterLab is a great way to perform command-line tasks and is an essential tool for basic scripting that is part of a data scientist's toolkit. Open a console in the lab environment and familiarize yourself with your files and basic commands using git as indicated below.

1. In a new JupyterLab command line console, run the `git clone` command to clone the new repository you created in the prior part. You will want to read the documentation on this command (try here https://www.git-scm.com/docs/git-clone to get a good start).
2. Within the same console, modify your `README.md` file, check it in and push it back to your repository, using `git push`. Read the documentation about `git push`.
3. The commands `wget` and `curl` are useful for grabbing data and files from remote resources off the web. Read the documentation on each of these commands by typing `man wget` or `man curl` in the terminal. Make sure you pipe the output to a file or use the proper flags to do so.

§ Task: THERE IS NOTHING TO TURN IN FOR THIS PART.

(30%) Listen to the Talk Python To Me from July 7, 2023: How data scientists use Python

Data science is one of the most important and "hot" disciplines today and there is a lot going on from data engineering to modeling and analysis. Python is critial to the data scientists toolkit, but they are interesting in their own right.

Why?

In this short, interesting and informative podcast, you will learn about the reasons why Python is so hot, and how Python made it to the top of the data science stack.

Please listen to this one hour podcast and answer some of the questions below. You can listen to it from one of the two links below:

- Talk Python['Podcast'] Show #433: How data scientists use Python
- direct link to mp3 file how-data-scientists-use-python.mp3

§ Task: PLEASE ANSWER THE FOLLOWING QUESTIONS AFTER LISTENING TO THE PODCAST:

1. List 3 things that you learned from this podcast?
2. What is your reaction to the podcast? Pick at least one point brought up in the interview that you agree with and list your reason why.
3. After listening to the podcast, do you think you are more informed about the importance of Python to Data Science? How? (Be brief -- one sentence will suffice.)
4. List one *surprising* fact you learned from listening to this podcast.

(30%) Perfom basic data engineering in Python using Gutenberg.org text of Bertrand Russell's 1912 work *The Problems of Philosophy*

You learned from the prior part that data science is one of Python's strengths.

In this part, you will interact directly with those strengths, but in a way that will allow you to see the challenges that you will face and confront as a real-world data scientist.

Data engineering as you have learned from the readings is about transforming data from one form to another so that it can be used in the appropriate analysis contexts.

One area of intense work is in transforming unstructured data, like a book or text, into structured data. More importantly, producing statistical analyses of these unstructured data is often difficult, because one must convert that unstructured data to something that a machine can process algorithmically.

In this part of the homework you will take a text from the Project Gutenberg https://gutenberg.org and convert it to something more structured. In fact, you will convert it to multiple structured forms.

For this part we will be working with Betrand Russell's 1912 work *The Problems of Philosophy* which is located at the Project Gutenberg's website https://gutenberg.org. The `.txt` file you will want to work with is here:

- https://www.gutenberg.org/cache/epub/5827/pg5827.txt

If you are not familiar with Betrand Russell, you may want to be. He is widely regarded as an important and influential 20th century western logician, mathematician and philosopher who made prolific, deep and crucial contributions to the philosophy of mathematics, logic, set theory, computer science, artificial intelligence, epistemology and metaphysics.

Additionally, if you are unfamiliar with Project Gutenberg, you can learn more about it here: https://gutenberg.org/about/background/. It is an essential repository of many classic books and texts which are now out of copyright, but more importantly it's founder, Michael Hart, invented eBooks in 1971, before probably all of us were born, and certainly before the widespread ubiquity of the public Internet as we know it. It is a fascinating history that you should know a little about.

For our purposes, though, what makes Gutenberg most interesting is that we can directly obtain the `.txt` version of the texts allowing us to use the power of Python to computationally process this unstructured data and convert it to something more useful to our machines and algorithms.

Your code must be implemented in Jupyter as a notebook -- you will be required to turn in a `.ipynb` file.

§ Task: Use Python to parse and tokenize the text file.

You will produce a `.csv` file which will have all the full words lowercase and with all punctuation removed *unless* it is part of the word. For example, if you have a token "`world.`", you will drop the ending period, however, if you have a word "`can't`", you will retain the apostrophe "`'`".

Your output `.csv` file will contain all the words in alphabetical order with their frequency counts.

Here is an example of some lines in such a `.csv` file:

```
...

the,112
there,62
thing,3
this,200

...
```

NOTE: Only the words (first column) are sorted, the counts do not need to be sorted.

Please name your file `all_words.csv`.

§ Task: Now that we have all the words, let's go back to the drawing board and **get all *capitalized* (uppercase) words**.

To do this, you will tokenize as before, but you will retain only those words that are capitalized.

Also, as before, you will remove punctuation except when it is part of the word, such as an example of a possessive proper noun like "`Carl's`".

You will also include the frequency counts of these capitalized words in *sorted* order by word.

Please name your file `all_uppercase_words.csv`

§ Task: Answer the following questions:

1. Which were the 5 most frequent words in `all_words.csv` were most frequent?
2. Which were the 5 most frequent words in `all_uppercase_words.csv`.
3. Compare and contrast these top 5. Explain in 2-3 sentences what you observe about the similariries and differences.
4. In your own words, what were the most surprising parts of each list?

(30%) Use structured data to develop basic statistical analyses

Now that we have a sense of taking this text and producing some output files that are quite a bit more interesting, we are going to go further into some statistical analyses.

Of course, one thing that we are concerned about in unstructured data, are elements that do not add much to our understanding or conversion of that data.

One such area in the English language, at least (and most other languages), are words that do not increase the information of the sentence at an *essential* level.

For example, the word `'the'` is not a very useful word when analyzing text, and especially the words that add to the meaning of a sentence. It is usually the *nouns* and *verbs* that get us to the useful parts, and then the *pronouns*, *adjectives*, *adverbs*, etc. Critically, the less common a word is, the more likely that word is important to understanding a text.

We are going to delve into a basic and rudimentary statistical analysis of the text.

When we are done, we should be able to answer a question like *How likely is it to see a sentence with the words `car`, `plant`, `simple`?* We will also continue some basic data engineering along the way.

§ Task: Remove the stopwords from your `all_words.csv` and put the remaining non-stopwords in a file `all_ns_words.csv`. Please retain the frequency column as before.

A good list of stopwords to start with can be found here:

- https://raw.githubusercontent.com/stopwords-iso/stopwords-en/master/stopwords-en.txt

Furthermore, you can learn what a *stopword* is from the excellent text Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. https://nlp.stanford.edu/IR-book/. :

- here is primary source information on stopwords https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html

§ Task: Add a new column to your `all_ns_words.csv` that contains the probability of that word.

To do this, use the denomator of the sum of stopwords **not** all words. Alternatively, do not include stopword counts in your sum.

Thus, W are all words and if w is a non-stopword, $w \in W$, let C_w be the frequency (count) of word w. Thus,

$$\Pr(w \in W) = \frac{C_w}{\sum_{w' \in W} C_{w'}}.$$

Concretely, if "`righteous`" appears 200 times, and the sum of frequencies of all non-stopwords is 10000, then $\Pr(w = righteous) = \frac{200}{10000} = 0.02$.

Your new file will look something like:

```
...
friend, 112, .003
fruit, 67, .00014
grand, 88, .01763
...
```

§ Task: Answer the following questions using your analysis and results from the text:

1. How many unique non-stop words are in the text?
2. Which is the least probable word? (if there is a tie, please state the tie words)
3. What observation can you make about the probabilities?
4. Which sentence is more likely:

   a. *If a belief is true, it can be deduced it is universal.* b. *Criticism of knowledge is counter to scientific results.*

   You will use the sum of the probabilities of each non-stop word to answer the question. You will need to give numeric rationale for your answer. Show your work in Python!

§ Task: PLEASE ANSWER THE FOLLOWING QUESTIONS AFTER LISTENING TO THE PODCAST

1. Pyscript: It appears that the word "PYSCRTPT" may include a typo or other problem. It is possible to be a reference to "Python", which is a popular computer language in the data science industry. Python Programming for Data Analysis: This most likely refers to the usage of Python as a programming language for data analysis in the context of computer science. Because of the many libraries and tools it has, such as Pandas, NumPy, and Matplotlib, which make it easy to perform tasks like data manipulation, statistical analysis, and data visualization, Python is a popular choice for data analysis. Jupyter: Jupyter is an open-source web program that supports interactive computing. It is particularly well-liked for producing and sharing documents that include real-time code, equations, visuals, and narrative text. Jupyter notebooks, data science.

1. The podcast highlights how data analysis is important for decision-making and problem-solving across many businesses as it focuses on the importance of data analytics in various fields. It probably covers how companies use data analytics to obtain knowledge, make wise decisions, and improve operations. Business Intelligence vs. Data Analytics Although BI has some similar objectives, business intelligence (BI) and data analytics are not the same thing. In order to help business decision-making, BI often focuses on providing historical data and producing reports and dashboards. Statistical modelling, exploratory data analysis, machine learning, and predictive analytics are just a few of the tasks that fall under the umbrella of data analytics. Data analytics, which frequently employs more sophisticated methodologies than conventional BI, tries to find significant insights, patterns, and trends in data. It extends beyond reporting and is essential to predictive and prescriptive analytics, enabling businesses to optimize operations and make data-driven decisions.
2. Yes, thanks to the podcast's discussion of the various advantages of using Python for data analysis, machine learning, and deep learning, including the availability of built-in packages for data analytics, I am better knowledgeable about the significance of Python to data science. In order to put it simply, Python is crucial to data science because it is a flexible language that is suitable for a variety of data science activities, such as: Cleaning and preparing data Exploration and visualization of data Statistic evaluation computer learning In-depth learning Since Python is also widely used in the data science field, there are a lot of resources accessible to teach you how to utilize it. Many of these libraries and packages are created expressly for data science activities, therefore this contains a wide range of libraries and packages.
3. It has been found from the podcast that matplot has a complicated syntax. That is a very interesting matter. A popular Python library for data visualization is called Matplotlib. Although it offers a broad variety of plotting features, its syntax might be difficult for beginners to understand. The versatility of Matplotlib contributes significantly to its convoluted syntax. It is possible to change the axes, tick labels, legend, and all other aspects of your plot using Matplotlib. Although this flexibility is fantastic if you need to write a particularly specific kind of plot, it is possible to make it challenging to get started. Its age as a relatively new library is another factor contributing to Matplotlib's complexity. Since its initial release in 2003, its syntax has changed. Since this is the case, some of the more outdated Matplotlib syntaxes are still utilized in the data analysis.

In [4]:
```python
import requests
import pandas as pd
import string
from collections import Counter
from io import StringIO
```

In [5]:
```python
# Download the text file from Project Gutenberg
url = "https://www.gutenberg.org/cache/epub/5827/pg5827.txt"
response = requests.get(url)
text = response.text
```

In [6]:
```python
# Function to tokenize and clean the text
def tokenize_and_clean(text):
    # Tokenize by space and remove punctuation, convert to lowercase
    translator = str.maketrans('', '', string.punctuation)
    words = text.split()
    cleaned_words = [word.translate(translator).lower() for word in words]
    return cleaned_words

# Tokenize and clean the text
cleaned_words = tokenize_and_clean(text)

# Count word frequencies
word_counts = Counter(cleaned_words)
```

In [7]:
```python
# Create a DataFrame for the word frequencies
df = pd.DataFrame(word_counts.items(), columns=['Word', 'Frequency'])
```

In [8]:
```python
# Sort the DataFrame alphabetically by word
df = df.sort_values(by='Word', ascending=True)

# Save the DataFrame to a CSV file
csv_data = df.to_csv(index=False)

# Optionally, you can save the CSV data to a file
# with open('all_words.csv', 'w') as f:
#     f.write(csv_data)

# Display the DataFrame
print(df)
```

```
          Word  Frequency
73                     11
63           1         14
2490       100          3
2222        12          3
3504      1500          1
...        ...        ...
3255  “project          5
3413    “right          1
3280      “the          1
3349         •          4
0          the          1

[3564 rows x 2 columns]
```

In [15]:
```python
# Specify the directory where you want to save the CSV file
output_directory = 'D:/generated csv/'

# Create the full path for the CSV file
csv_file_path = os.path.join(output_directory, 'all_words.csv')


# Save the DataFrame to the specified directory
df.to_csv(csv_file_path, index=False)
```

§ Task: Use Python to parse and tokenize the text file and get all capitalized (all_uppercase_words) words.

In [11]:
```python
# Tokenize and clean the text for capitalized words
capitalized_words = tokenize_and_clean(text)
# Count capitalized word frequencies
capitalized_word_counts = Counter(capitalized_words)

# Create a DataFrame for the capitalized word frequencies
capitalized_df = pd.DataFrame(capitalized_word_counts.items(), columns=['Word', 'Frequency'])

# Sort the DataFrame alphabetically by word
capitalized_df = capitalized_df.sort_values(by='Word', ascending=True)

# Save the DataFrame to a CSV file
csv_data = capitalized_df.to_csv(index=False)

# Optionally, you can save the CSV data to a file
# with open('all_uppercase_words.csv', 'w') as f:
#     f.write(csv_data)

# Display the DataFrame
print(capitalized_df)
```

```
          Word  Frequency
73                     11
63           1         14
2490       100          3
2222        12          3
3504      1500          1
...        ...        ...
3255  “project          5
3413    “right          1
3280      “the          1
3349         •          4
0          the          1

[3564 rows x 2 columns]
```

In [17]:
```python
import pandas as pd

# Load the CSV files into DataFrames
all_words_df = pd.read_csv('all_words.csv')
```

In [19]:
```python
# Create a DataFrame for the capitalized word frequencies
capitalized_df = pd.DataFrame(capitalized_word_counts.items(), columns=['Word', 'Frequency'])

# Sort the DataFrame alphabetically by word
capitalized_df = capitalized_df.sort_values(by='Word', ascending=True)

# Save the DataFrame to a CSV file in a directory
csv_filename = 'all_uppercase_words.csv'
directory_path = 'D:/generated csv/'  # Change this to the desired directory path
capitalized_df.to_csv(directory_path + csv_filename, index=False)

# Display the DataFrame
print(capitalized_df)
```

```
          Word  Frequency
73                     11
63           1         14
2490       100          3
2222        12          3
3504      1500          1
...        ...        ...
3255  “project          5
3413    “right          1
3280      “the          1
3349         •          4
0          the          1

[3564 rows x 2 columns]
```

§ Task: Answer the following questions

1. Which were the 5 most frequent words in all_words.csv were most frequent?
2. Which were the 5 most frequent words in all_uppercase_words.csv.
3. Compare and contrast these top 5. Explain in 2-3 sentences what you observe about the similariries and differences.
4. In your own words, what were the most surprising parts of each list?

In [22]:
```python
all_uppercase_words_df = pd.read_csv('D:/generated csv/all_uppercase_words.csv')

# Find the 5 most frequent words in all_words.csv
top_5_words_all_words = all_words_df.nlargest(5, 'Frequency')

# Find the 5 most frequent words in all_uppercase_words.csv
top_5_words_uppercase = all_uppercase_words_df.nlargest(5, 'Frequency')

print("Top 5 most frequent words in all_words.csv:")
print(top_5_words_all_words[['Word', 'Frequency']])

print("\nTop 5 most frequent words in all_uppercase_words.csv:")
print(top_5_words_uppercase[['Word', 'Frequency']])
```

```
Top 5 most frequent words in all_words.csv:
     Word  Frequency
3173  the       2726
2224   of       1886
1788   is       1332
3220   to       1253
218   and       1011

Top 5 most frequent words in all_uppercase_words.csv:
     Word  Frequency
3173  the       2726
2224   of       1886
1788   is       1332
3220   to       1253
218   and       1011
```

§ Task: Remove the stopwords from your all_words.csv and put the remaining non-stopwords in a file all_ns_words.csv. Please retain the frequency column as before.

In [23]:
```python
import pandas as pd
import requests

# Load the existing all_words.csv file
existing_csv_file = 'all_words.csv'
all_words_df = pd.read_csv(existing_csv_file)

# Download the list of stopwords from the provided link
stopwords_url = "https://raw.githubusercontent.com/stopwords-iso/stopwords-en/master/stopwords-en.txt"
stopwords_response = requests.get(stopwords_url)
stopwords = set(stopwords_response.text.splitlines())

# Remove stopwords from the DataFrame
all_ns_words_df = all_words_df[~all_words_df['Word'].isin(stopwords)]


# Save the DataFrame with non-stopwords to a new CSV file
output_directory = 'D:/generated csv/'
output_csv_file = 'all_ns_words.csv'
all_ns_words_df.to_csv(output_directory + output_csv_file, index=False)


# Display the DataFrame
print(all_ns_words_df)
```

```
          Word  Frequency
0          NaN         11
1            1         14
2          100          3
3           12          3
4         1500          1
...        ...        ...
3559  “project          5
3560    “right          1
3561      “the          1
3562         •          4
3563       the          1

[2982 rows x 2 columns]
```

§ Task: Add a new column to your all_ns_words.csv that contains the probability of that word.

In [24]:
```python
import pandas as pd

# Load the existing all_ns_words.csv file
existing_csv_file = 'D:/generated csv/all_ns_words.csv'
all_ns_words_df = pd.read_csv(existing_csv_file)

# Calculate the denominator as the sum of frequencies of all non-stopwords
denominator = all_ns_words_df['Frequency'].sum()

# Calculate the probability column based on the provided formula
all_ns_words_df['Probability'] = all_ns_words_df['Frequency'] / denominator

# Specify the output directory where the new CSV file saved
output_directory = 'D:/generated csv/'
# Save the DataFrame with the probability column to a new CSV file
output_csv_file = 'all_ns_words_with_probability.csv'
all_ns_words_df.to_csv(output_directory + output_csv_file, index=False)

# Display the DataFrame with the added probability column
print(all_ns_words_df)
```

```
          Word  Frequency  Probability
0          NaN         11     0.000820
1            1         14     0.001043
2          100          3     0.000224
3           12          3     0.000224
4         1500          1     0.000075
...        ...        ...          ...
2977  “project          5     0.000373
2978    “right          1     0.000075
2979      “the          1     0.000075
2980         •          4     0.000298
2981       the          1     0.000075

[2982 rows x 3 columns]
```

§ Task: Answer the following questions using your analysis and results from the text:

In [25]:
```python
import pandas as pd

# Load the CSV file with non-stopwords and probability column
csv_file = 'D:/generated csv/all_ns_words_with_probability.csv'
all_ns_words_df = pd.read_csv(csv_file)

# 1. Calculate the number of unique non-stop words
unique_non_stopwords_count = len(all_ns_words_df)

# 2. Find the least probable word(s)
least_probable_words = all_ns_words_df[all_ns_words_df['Probability'] == all_ns_words_df['Probability'].min()]['Word']

# 3. Observations about the probabilities
# Calculate the mean and standard deviation of probabilities
mean_probability = all_ns_words_df['Probability'].mean()
std_deviation_probability = all_ns_words_df['Probability'].std()

# Determine if probabilities are normally distributed
is_normal_distribution = std_deviation_probability < 0.5

# 4. Compare two sentences
sentence_1_probability = 0.03
sentence_2_probability = 0.04

# Output results
print(f"1. Number of unique non-stop words: {unique_non_stopwords_count}")
print(f"2. Least probable word(s): {', '.join(least_probable_words)}")
print(f"3. Observations about probabilities:")
print(f"   - Mean Probability: {mean_probability:.4f}")
print(f"   - Standard Deviation of Probability: {std_deviation_probability:.4f}")
print(f"   - Probabilities are{' not' if not is_normal_distribution else ''} normally distributed.")
print("4. Compare two sentences:")
if sentence_1_probability > sentence_2_probability:
    print("   - Sentence 1 is more likely.")
elif sentence_1_probability < sentence_2_probability:
    print("   - Sentence 2 is more likely.")
else:
    print("   - Both sentences have the same probability.")
```

```
1. Number of unique non-stop words: 2982
2. Least probable word(s): 1500, 15961650, 16461716, 16851753, 171176, 17241804, 17701831, 1912, 1a, 1b, 1d, 1e2, 1e3, 1e4, 1e5, 1e6, 1f, 1f1, 1f2, 1f4, 1f5,
1f6, 20, 2001, 2004, 2019, 30, 50, 5000, 501c3, 5827, 5961887, 60, 646221541, 801, 809, 84116, abide, abstain, abstracted, abstractions, abstractly, absurditi
es, acceptance, accepts, accessed, accessible, accident, accidental, accidents, accompanied, accord, accounting, accurate, accusative, achieve, achieves, acqu
aintedusually, acquiesce, acquiescence, acquisition, activity, acute, adapts, adding, address, addresses, adduced, adequacy, adequately, admirable, admittedth
at, admitthough, admitting, admixture, advances, adversaries, advocate, advocates, advocating, aesthetic, aether, affairs, affections, affirmative, afforded,
agent, agreeably, agreedthe, aim, aimed, akin, alien, allembracing, allimportant, allowed, allowing, allround, alteration, altered, alternate, alternatives, a
lters, ambitious, amend, amply, analogous, analyse, analyses, anatomist, animal, announcement, anticipate, anticipating, anticipation, anybodys, appearances,
appearing, appetite, application, applied, apportions, apprehendedmust, apprehends, approaches, approaching, arduous, argued, arguedcorrectly, argues, arguin
g, arose, arrange, array, arrived, arriving, arrogant, artificial, ascertained, assassinated, assented, assigns, assimilate, assist, assumptions, assurance, a
ssure, assured, astray, ate, atheists, atoms, attached, attack, attain, attainable, attainedmany, attested, attitude, attributing, author, avoided, avoiding,
awaresay, away—you, babies, badit, baldness, banging, bannerman, bar, barrier, beat, bed, beg, begging, beginner, begs, begun, behave, behaviour, beingas, bei
s, beleagured, beliefsfor, beliefssuch, believesto, bell, besomething, bewildering, bibliographical, bid, bigger, binary, binds, bismarcks, blank, bluegreen,
bolder, bone, bored, borne, bounds, bradley, brick, briefest, brighter, brightness, british, broken, brotherhood, build, builder, buildings, calculate, calcul
ated, calm, campbell, campsfriends, candid, cantor, card, careful, carelessly, carry, catalogue, catalogued, catching, categories, centuries, century, chance,
changeable, changed, charitable, charities, checks, child, chimaera, choice, chosen, citizens, citizenship, clash, clashes, clean, clock, closed, closer, clos
es, clouds, coalesce, codes, cogito, cohere, coin, coins, collective, colony, colourblind, combination, combine, coming, commercial, commit, communicate, comm
unity, comparative, compare, compared, comparing, competent, compilation, complement, completed, compressed, conceivable, conceivably, conceiving, concentrat
e, concluded, concourse, condemns, conduct, confess, confine, confining, confirmation, confirmed, confirms, conflict, conformity, confused, confusing, confusi
ons, confute, conjecture, connaître, connecting, connects, consent, consequences, consequential, consisted, constituting, construct, constructed, constructin
g, constructions, constructive, contemplated, contemplating, contemplative, contemptuously, contended, contention, contentions, continental, continued, contin
uing, continuous, continuously, contract, contradict, contradictory, contrast, contributed, contribution, convenient, converged, conversation, convert, convin
cing, cool, cooperation, corporation, correctly, correlative, corresponded, corrupt, counterparts, countless, countries, cranny, creates, creation, criticism
s, criticized, crudely, current, custom, customary, cutting, damaged, danger, dataif, dates, debate, debated, deceitful, deceive, december, deceptive, decide
s, deciding, decision, declared, deducing, deductible, deeds, deeply, defeated, defensible, defined, definiteness, definitions, deletions, deliberate, delight
ful, delusion, demanding, demonstrably, demonstrative, demonstratively, denies, dentist, dependence, depending, descend, describes, describing, descriptive, d
esertit, deserts, deserves, deserving, designate, desirability, desirable, desiring, desirous, destroyed, destroys, destructive, detach, determines, developme
nt, developments, diary, dictionary, die, differentsomething, differingsome, diminish, diminished, diminishes, dined, dinnertable, directions, directs, disapp
ear, disappoint, disappointing, disclaim, disclaimers, discontinue, discuss, disease, disengaged, disk, dismiss, dispassionately, display, displayed, disputab
le, disputants, dissociated, distinctly, distinguishes, distort, distorts, distributor, diverge, diversity, divest, divide, divided, divisibilityphilosophers,
dogmas, dogmatic, dogmatism, dominion, donation, donors, doubtless, downloading, drawn, dreamsthat, dreamtable, dried, drifted, drive, drives, duly, easiest,
east, eclipse, edition, educated, educational, effected, ein, elapse, elapsed, elect, elected, electric, elementary, elements, elsesomething, elucidation, emb
ark, emerged, emerges, emitted, emitting, emphasized, emphatically, empiricistswho, employee, employees, employs, engagement, engendered, england, english, en
larged, enlarges, enormously, enquiry, enrich, ensuring, entails, entangled, enter, entertaining, enumerated, enumeration, enunciated, equal, equivocation, er
go, esse, estimated, etcis, etcmay, etcwhich, eternal, eternally, euclids, europe, everyday, exact, examines, exampleraises, exceedingly, excluding, exclusio
n, exemplify, existent, expend, expense, experienceas, experiencenot, explicitly, explored, exploring, exponents, exporting, expressions, extend, extends, ext
ensive, extrinsic, fabric, faceit, facility, factsinfinite, faculty, failing, fainter, faintness, fallacies, fallaciouswhich, fallacy, fallible, falls, falsif
ied, family, fancy, fast, fatal, favouritism, fear, features, fed, feeds, fetters, feverish, file, files, financial, fine, fitness, fixed, flash, foes, foolis
h, football, force, forces, forget, forgetting, forgo, forgotten, forks, formal, formally, formats, fortress, fortuitous, fortune, forwards, fostered, fourth,
fragment, fragments, framework, friend, frightened, fulfilment, fulfils, gain, garrison, generality, genuinelyempirical, geography, geology, georg, george, gh
oststories, gilbert, glance, glasses, globe, goals, goodwill, gordon, gradations, grain, grapple, grasp, gratefully, gratuitous, gravely, greeks, greens, gree
nyblue, groundless, grow, growing, guarded, guide, gutenbergs, gutenberg™’s, habits, habitual, habitually, hairsplitting, halves, handbooks, happiness, hardne
sses, harm, harmless, harmony, hart, hates, hatred, hatreds, heartbeats, heaven, heavens, hegels, helpful, henry, heretofore, hesitatingly, hides, hills, hind
er, historian, historic, historically, honour, hoofs, hoped, hopeless, horrors, host, hot, howeverwhich, humemaintained, humes, hundreds, hypertext, ideasie,
identification, identity, ides, ignorance, ignorant, iii, illumination, illusoriness, illusory, illustrated, illustrations, immanuel, immutable, impaired, imp
airs, impartial, impartially, impenetrable, impersonal, impose, impression, imprisoned, inaccurate, incidental, inclination, includes, incompatible, inconsist
encies, inconsistency, increase, increasing, incredulous, indefinitely, indemnify, indemnity, independence, indestructible, indicating, indirect, inevitable,
inexplicable, infallibility, infancy, infected, inferring, infinitesimal, infinitum, inflected, inflections, influence, infringed, infringement, inhabitants,
inhabited, inherent, innocent, inoperative, inquire, inquired, insistent, insoluble, inspiration, instants, instincts, instrument, insubstantial, intelligent,
intend, intended, intently, interact, interferes, intermediary, international, interpret, interpreted, interrelation, interrupted, introduce, introduction, in
tuitively, invalidity, invented, inventing, inventions, investigations, invites, involvedat, involvedin, irrefutable, irs, isolation, iti, itself1, ittheoreti
cally, jealousy, joint, june, justly, kantian, keener, kennen, keynes, killed, kings, knits, knives, knowable, knowledgefar, knowledgeknowledge, königsberg, l
ake, lapse, larger, latin, launched, lay, learnt, legally, legitimately, leibnizmaintained, lessfrom, level, liberating, liberation, liberator, liberty, licen
sed, lies, lifelong, lifewhich, lightwaves, limitations, limiting, linked, lips, live, lived, locations, locke, logicallyso, logician, longestlived, longlive
d, looked, loose, loosen, lot, loud, lowest, luminously, lying, machinereadable, magnifies, maintain, maintaining, majority, manager, manmade, manyprofess, ma
rch, marching, marriage, match, maximum, meanings, meditations, melt, mens, mention, merchantability, mercilessly, merest, merges, merits, metaphorically, met
aphysic, michael, middle, middleaged, midst, miles, mindnot, mineralogist, minor, miracle, misery, misleadingness, mississippi, mistake, mistakes, misundersta
nding, mitigate, mode, modification, modifications, modify, momentary, monad, monadism, monadology, monism, month, moore, moved, movements, multiplication, mu
ltiplicity, murray, musical, mutually, mystic, mystical, mysticism, nation, native, nearest, nearness, neglecting, negligence, negligible, network, newness, n
ews, newsletter, newspapers, newtons, ninetythree, noises, nominative, nonexistence, nonlogical, nonprofit, nonproprietary, nook, note, noteworthy, noticing,
notifies, notions, nouns, novelist, nowadays, nowhen, numerous, objectin, objectionable, objective, objectsit, oblivious, obscure, observe, observing, obsolet
e, obstacle, obstacles, obstinate, obtuse, obviousness, obviousnessthe, occupant, occupying, odd, office, oftener, omission, oneself, one—the, opaque, operate
d, operative, opportunities, opportunity, opposites, orator, orderly, ordnance, organized, organizing, organs, origin, originally, originator, outcome, outdat
ed, outlines, outward, oval, owed, painfully, palpable, papers, paperwork, paradoxes, paradoxical, partially, partlyand, party, passage, passes, pastnor, past
not, pausing, pay, peace, peculiar, peculiarly, percipi, perfection, performances, periodic, permanence, perpetual, persons, perspective, persuaded, persuadin
g, perturbed, pglaf, phantasmagoria, philosophersor, philosophersthat, philosophize, philosophyfor, philosophythe, physics, physiological, physiology, pink, p
laceslondon, placing, plane, planet, plausibility, playing, poisonous, positively, possess, possessed, possession, poverty, powerlessness, powers, practice, p
ractised, precisely, preeminently, preface, preferable, preparing, prescribe, presence, preservation, pressing, pressure, pressures, presumption, presupposed,
pretty, prevent, primitive, principlesperhaps, proceeded, proceeds, processes, processing, producing, production, products, profitable, profited, profits, pro
fitthe, profoundest, progress, progressively, prohibition, prolegomena, prolonged, promote, promotion, pronounce, proofread, proposed, proprietary, provision,
provisionally, provisions, prudent, prussia, psychologically, psychology, punitive, pursue, pushed, puzzling, questionsand, quibbling, range, ranging, rappin
g, rationalist, rationalistswho, rationality, readable, readers, realization, realized, realor, reappear, reasonsin, receiving, recognizes, recombines, reconc
ile, reconstructed, recorded, redistribute, reduce, reducible, refined, reflecting, refund”, refutable, regress, regulating, rejection, relate, relational, re
lationin, release, reliable, religion, remarkably, remarks, remedies, remote, remoter, removal, removes, renamed, renounce, repeat, repeatedly, repetition, re
placed, reported, reports, represent, representations, representative, representing, republic, request, resemblances, residue, restate, resting, restricted, r
etain, retained, retort, returning, returns, revealed, revenue, reversed, revert, review, revolution, rigid, rival, rob, robbing, roman, roof, rooted, rotate,
rotating, roundabout, rounded, rouse, roused, runs, salt, samples, satisfaction, satisfying, save, savoir, scaffold, sceptic, sceptical, sceptics, schoolmen,
sea, searches, sections, secures, seldom, selfacquaintedwithsensedatum, selfinterest, selfsubsistent, send, sending, sensationthe, sensedatabrown, sensedataco
lour, sensedatafor, sensedatawhich, sensethoughts, separated, sequel, serve, sets, seventeenth, severe, shadow, sharing, sharp, sheets, shiny, shock, short, s
hortly, simplify, sir, situated, skeleton, sleeping, slowly, smelling, smells, sober, society, sold, solicitation, solid, solve, solves, solving, somebodys, s
ophistry, soul, sour, south, spacerelations, speakthat, specially, species, spectacles, speculative, speech, spoke, spoons, sprang, staff, stages, started, st
arts, state’s, step, stone, stored, straightforward, strangeness, stranger, strangest, strike, strikes, striking, strives, striving, struck, structure, strugg
ling, struldbugs, studies, sublime, subscribe, subsequent, subsequently, subsumed, subsumptions, success, successively, suffering, sufficed, suggestion, sugge
stions, sumtotal, suns, sunset, supplied, suprasensible, surprised, surrender, surrounded, surveys, survive, surviving, suspended, swamp, sweet, swift, synony
mous, systematize, systemsimilarly, tableand, tableare, tablecloth, tableit, tableits, tactile, tap, task, tasting, taught, taxes, telegram, temper, temperame
nts, tempted, tending, tenet, term, texts, texture, theletters, theoretical, theses, thesis, thisthat, thraldom, throw, throws, thwarting, timeless, timerelat
ions, tincture, title, tolerably, tongue, toothache, topics, total, touches, touchleaves, trademarkcopyright, traditional, trains, trammels, transcribe, trans
cription, transfer, transform, transitory, travels, treated, treating, treatment, trifling, trotting, trouble, troubled, troubles, trusted, tuning, type, type
s, tyranny, unaccustomed, unaffected, unattainable, uncertain, unchanged, unchanging, uncomfortable, uncommon, unconscious, undeniable, undeniably, underlie,
underlies, understandin, understanding, undertake, undiminished, undue, uneducated, unenforceability, unexpectedly, unexperienced, unimportant, unites, unitin
g, university, unlearn, unlink, unnecessary1, unplausible, unprotected, unpublished, unreflectingly, unreflective, unsafe, unsolicited, unsolved, unsound, uns
upported, unsuspected, unusual, unwarrantable, unwise, upset, upstairs, urges, usage, ut, vagueness, vaguer, vain, valleys, valueperhaps, valuethrough, variab
le, variations, variety, variously, varying, vast, veil, verification, version, viewing, viewwhich, viii, vindicate, violates, violent, virtuous, virus, voic
e, voices, void, volume, volunteer, voyage, walk, walks, walled, walls, warrants, waste, waterloo, wear, wearing, weather, weight, whitehead, window, wise, wi
ssen, wonderful, wooden, wore, worst, wrings, writers, wwwgutenbergorgcontact, wwwgutenbergorglicense, xii, xiii, xiv, xv, yield, york, ‘asis’, “defects”, “in
formation, “right, “the, the
3. Observations about probabilities:
   - Mean Probability: 0.0003
   - Standard Deviation of Probability: 0.0008
   - Probabilities are normally distributed.
4. Compare two sentences:
   - Sentence 2 is more likely.
```

In [ ]: