Creating the Word List

Table of Contents

Create and Run the Word Counter Python Script 1
Create the Dictionary File 2

Create and Run the Word Counter Python Script

To create the dictionary, we start with the unique words used and how many times each word is used in our sample. This Python script will give us a file formatted for easy import to a spreadsheet.

Procedure 1. To create and run the python script word_counter.py:

1. `sampleAgg.txt`
2.
3. `sampleAgg.txtword_counter.py`
4. **python word_counter.py > dictionary.csv**

```
import re
import sys
from collections import Counter

def analyze_document_frequency(filepath: str = "sampleAgg.txt"):
    """
    Analyzes a text document to count word frequency.

    This function reads a text file, converts its content to lowercase,
    extracts words between 4 and 15 characters long,
    and counts the occurrences of each extracted word. It prints
    the words sorted by frequency in descending order, then alphabetically for tie

    Args:
        filepath (str): The path to the input text file. Defaults to "sampleAgg.tx

    Returns:
        None: Prints the word frequencies to standard output. Errors are printed t
    """
    try:
        # Open and read the document, converting content to lowercase.
        # 'utf-8' encoding is specified for broad compatibility.
        with open(filepath, 'r', encoding='utf-8') as document_file:
            text_string = document_file.read().lower()
    except FileNotFoundError:
        # Handle the case where the specified input file does not exist.
        print(f"Error: The file '{filepath}' was not found.", file=sys.stderr)
```

```
            return
        except Exception as e:
            # Catch other general exceptions during file reading.
            print(f"An unexpected error occurred while reading the file: {e}", file=sy
            return

        # Use regular expression to find all words.
        # '\b' ensures whole words. '[a-z]{4,15}' matches lowercase letters between 4
        match_pattern = re.findall(r'\b[a-z]{4,15}\b', text_string)

        # Count word occurrences using collections.Counter for efficiency.
        word_counts = Counter(match_pattern)

        # Sort words: primary key is count (descending), secondary key is word (ascend
        # Lambda function creates a tuple for sorting: (-count) for descending, then w
        sorted_words = sorted(word_counts.items(), key=lambda item: (-item[1], item[0]

        print("Word Frequencies:")
        print("-----------------")
        # Iterate through the sorted words and print each word and its count, separate
        for word, count in sorted_words:
            print(f"{word} | {count}")

if __name__ == "__main__":
    # Make sure the function runs only when the script is executed directly, not w
    analyze_document_frequency()
```

Note on Code Source: This code started with the python script in *Alchemy of Tomes* [Fisher (2020)], which does not work in the latest Python versions. We used Gemini (Google AI) to update this script.

Create the Dictionary File

With three small samples of cybersecurity documentation from different organizations, the Python script gave us 4755 unique words. Your sample will be much larger. Your goal is to create a cybersecurity dictionary of approximately 1,000 words. This does not include product names or company-specific words.

Prerequisites: Run the Python script.

Effort: Use the output file from the Python script to create the dictionary file. This will require approximately ten minutes.

Procedure 2. To create the dictionary file:

1. We will use Google Sheets. You can use Microsoft Excel or similar.

   Our Python script uses a pipe (|) as a delimiter between the word and its frequency of use. When you import the file, set the pipe as the delimiter.

2. If you know that you will use a specific AI-driven tool, such as writer.com or jasper.ai, change the dictionary headers and values to work with the aquired tool.

   If you do not have a checker tool yet, change row 1 to be these headers: `Word, Count, Part of Speech, Definition, Good Example, Bad Example, Allowed?, Audience, Alt1, Alt2`.

3.

4. `all`

5. Audience

The script sorted the output by highest frequency to lowest. Start with the words most used. These will be the easiest.