Provide Context

The main principle of a controlled language is that every word is used with one definition and one part of speech (PoS). The PoS of the word in the Dictionary is important and must exist. The technical writer will review all the uses of each word in the aggregated sample, to analyze which PoS is most used, which best fits the style guide, and which is least ambiguous. The technical writer will use a search-all feature in Adobe PDF or in a feature-rich text editor, to get all instances of each word, with context, to analyze usage and make a decision. You can save the time of each lookup with a script that gets the context results for the technical writer.

This code gets the context of each word: 5 words before the word and 5 after. You can change the number of words when you run the script.

1. Save the words, the [0] item in each line of the CSV or the first column of the Dictionary file, as a file.
2. Save this code as a Python file and run it:

   python wordContext.py > words.csv aggregate.txt Context.txt 5
3. Send the results to the technical writer.

```
import re

def extract_word_context_to_file(words_file, content_file, output_file, \
 context_word_count=5):
    """
    Finds words from a list in a content file and extracts a context of
    'context_word_count' words before and after the found word.
    Writes the results to a new output file.

    Args:
        words_file (str): Path to file with words, one in each line.
        content_file (str): Path to the aggregate file.
        output_file (str): Pathname of file with results.
        context_word_count (int): Number of words before and after.
    """
    try:
        with open(words_file, 'r', encoding='utf-8') as wf:
            target_words = [word.strip() for word in wf if word.strip()]

        with open(content_file, 'r', encoding='utf-8') as cf:
            content = cf.read()
        """
        Split content into words, keep punctuation for simpler splitting.
        A more robust solution might separate punctuation.
        """
        all_words = re.findall(r'\b\w+\b|[^\w\s]', content.lower()) \
        # Lowercase for matching

        found_contexts = []

        for target_word in target_words:
            # Iterate through the content to find matches
            for i, word_in_content in enumerate(all_words):
                if word_in_content == target_word.lower(): \
                # Case-insensitive matching
```

```
                    # Determine start and end index for context
                    start_index = max(0, i - context_word_count)
                    end_index = min(len(all_words), \
                     i + context_word_count + 1)

                    # Extract the context words
                    context = all_words[start_index:end_index]

                    # Reconstruct the context string
                    context_string = ' '.join(context)

                    found_contexts.append(f"Target Word: {target_word}\n \
                     Context: {context_string.strip()}\n")

        with open(output_file, 'w', encoding='utf-8') as of:
            if found_contexts:
                for item in found_contexts:
                    of.write(item + "-----\n") \
                    # Added a delimiter for readability between entries
                print(f"Contexts successfully written to '{output_file}'.")
            else:
                of.write(f"No matching words found for words in \
                 '{words_file}' within '{content_file}'.")
                print(f"No matching words found. '{output_file}' \
                created with a note.")

    except FileNotFoundError:
        print(f"Error: Required file not found. \
        Make sure '{words_file}' and '{content_file}' exist.")
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    UWORDS_FILE = "UWords.txt"
    SAMPLE_AGG_FILE = "SampleAgg.txt"
    OUTPUT_CONTEXT_FILE = "FoundWordContexts.txt" \
    # New output file for contexts
    CONTEXT_WORDS = 5 \
    # Number of words before and after

    extract_word_context_to_file \
     (UWORDS_FILE, SAMPLE_AGG_FILE, OUTPUT_CONTEXT_FILE, CONTEXT_WORDS)
```

Script created by Gemini, Google AI.