BIS 420 PROGRAMMING FOR DATA SCIENCE

PRAJAKTA POHARE
CHAPTER 13 EXERCISE 13.4
ILLINOIS STATE UNIVERSITY

Modify the previous program to read a word list (see Section 9.1) and then print all the words in the book that are not in the word list. How many of them are typos? How many of them are common words that should be in the word list, and how many of them are really obscure?

```
import string


def load_book(book):
    with open('/Users/prajaktapohare/Library/CloudStorage/OneDrive-ILStateUniversity/BIS420/Week 13/book.txt', 'r', encoding='utf-8') as f:
        lines = f.readlines()
    start = 0
    end = len(lines)
    for i, line in enumerate(lines):
        if "*** START OF" in line:
            start = i + 1
        elif "*** END OF" in line:
            end = i
            break
    return ''.join(lines[start:end])


def clean_text(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator).lower()
```

```
def count_words(text):
    words = text.split()
    word_count = {}
    for word in words:
        word_count[word] = word_count.get(word, 0) + 1
    return word_count


def load_wordlist(wordlist):
    with open('/Users/prajaktapohare/Library/CloudStorage/OneDrive-
ILStateUniversity/BIS420/Week 13/wordlist.txt', 'r') as f:
        return set(word.strip().lower() for word in f)


def find_unknown_words(words, wordlist):
    return {word for word in words if word not in wordlist}


def categorize_unknown_words(unknown_words):
    typos = set()
    common_missing = set()
    obscure = set()

    known_common = {
        'mr', 'mrs', 'dont', 'cant', 'im', 'ive', 'youre', 'thats',
        'wont', 'doesnt', 'didnt', 'isnt', 'wasnt', 'arent', 'couldnt',
        'shouldnt', 'wouldnt', 'hell', 'shes', 'hes', 'okay', 'yeah'
    }

    for word in unknown_words:
```

```
        if len(word) <= 2 or word.isnumeric():
            typos.add(word)
        elif word in known_common or "'" in word:
            common_missing.add(word)
        else:
            obscure.add(word)

    return typos, common_missing, obscure

def analyze_book(book, wordlist):
    raw_text = load_book(book)
    cleaned_text = clean_text(raw_text)
    word_counts = count_words(cleaned_text)
    all_words = set(word_counts.keys())

    unknown_words = find_unknown_words(all_words, wordlist)
    typos, common_missing, obscure = categorize_unknown_words(unknown_words)

    print(f"\nAnalysis of {book}:")
    print(f"Total words: {sum(word_counts.values())}")
    print(f"Unique words: {len(all_words)}")
    print(f"Unknown words (not in dictionary): {len(unknown_words)}")
    print(f"  Typos (short/invalid): {len(typos)}")
    print(f"  Common but missing: {len(common_missing)}")
    print(f"  Obscure/rare words: {len(obscure)}")

    print("\nSample typos:", sorted(list(typos))[:10])
```

print("Sample common missing:", sorted(list(common_missing))[:10])

print("Sample obscure words:", sorted(list(obscure))[:10])

wordlist = load_wordlist('wordlist.txt')

book_file = 'book.txt'

analyze_book(book_file, wordlist)

```python
import string

def load_book(book):
    with open('/Users/prajaktapohare/Library/CloudStorage/OneDrive-ILStateUniversity/BIS420/Week 13/book.txt', 'r', encoding='utf-8') as f:
        lines = f.readlines()
    start = 0
    end = len(lines)
    for i, line in enumerate(lines):
        if "*** START OF" in line:
            start = i + 1
        elif "*** END OF" in line:
            end = i
            break
    return ''.join(lines[start:end])

def clean_text(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator).lower()

def count_words(text):
    words = text.split()
    word_count = {}
    for word in words:
        word_count[word] = word_count.get(word, 0) + 1
    return word_count

def load_wordlist(wordlist):
    with open('/Users/prajaktapohare/Library/CloudStorage/OneDrive-ILStateUniversity/BIS420/Week 13/wordlist.txt', 'r') as f:
        return set(word.strip().lower() for word in f)

def find_unknown_words(words, wordlist):
    return {word for word in words if word not in wordlist}

def categorize_unknown_words(unknown_words):
    typos = set()
    common_missing = set()
    obscure = set()

    known_common = {
        'mr', 'mrs', 'dont', 'cant', 'im', 'ive', 'youre', 'thats',
        'wont', 'doesnt', 'didnt', 'isnt', 'wasnt', 'arent', 'couldnt',
        'shouldnt', 'wouldnt', 'hell', 'shes', 'hes', 'okay', 'yeah'
    }

    for word in unknown_words:
        if len(word) <= 2 or word.isnumeric():
            typos.add(word)
        elif word in known_common or "'" in word:
            common_missing.add(word)
        else:
```

```
        else:
            obscure.add(word)

    return typos, common_missing, obscure

def analyze_book(book, wordlist):
    raw_text = load_book(book)
    cleaned_text = clean_text(raw_text)
    word_counts = count_words(cleaned_text)
    all_words = set(word_counts.keys())

    unknown_words = find_unknown_words(all_words, wordlist)
    typos, common_missing, obscure = categorize_unknown_words(unknown_words)

    print(f"\nAnalysis of {book}:")
    print(f"Total words: {sum(word_counts.values())}")
    print(f"Unique words: {len(all_words)}")
    print(f"Unknown words (not in dictionary): {len(unknown_words)}")
    print(f"  Typos (short/invalid): {len(typos)}")
    print(f"  Common but missing: {len(common_missing)}")
    print(f"  Obscure/rare words: {len(obscure)}")

    print("\nSample typos:", sorted(list(typos))[:10])
    print("Sample common missing:", sorted(list(common_missing))[:10])
    print("Sample obscure words:", sorted(list(obscure))[:10])



wordlist = load_wordlist('wordlist.txt')


book_file = 'book.txt'

analyze_book(book_file, wordlist)
```