Paul Tang aventador917@csu.fullerton.edu
Zhiwei Su zwsu@csu.fullerton.edu

Summary of report document

This report is mainly divided into two parts: pseudo code and screenshots of test examples.

Briefly introduce my algorithm: My algorithm uses a large number of dictionaries to optimize the efficiency of the algorithm.

I created a class Word to represent a class of words that can be combined. It has two attributes, one is the word specifically represented by it, and the other is its word frequency.

We first parse Synonyms into a list of Word classes. Then use word_dict to associate specific words with Word classes. When reading the word frequency, we can quickly get the word class corresponding to the word, and then add the word frequency to the word frequency attribute in the Word class.

Finally, we print out all the Word objects and get the final result.

Pseudocode of the algorithm

```
class Word:
    def __init__(self):
        self.words_frequencies = 0
        self.words = []

    def add_word(self, word):
        self.words.append(word)

    def add_frequencies(self, f):
        self.words_frequencies += f

def parse_string(str):
    pattern = re.compile(r'\(.*?\)')
    find_result = pattern.findall(str)
    result_list = []
    for f in find_result:
        f = f.replace("(", "").replace(")", "").replace("”  ", "").replace(" “", "").split(",")
        result_list.append(f)
    return result_list


print("Input:")
words_frequencies = input()
synonyms = input()
synonyms_list = parse_string(synonyms)
word_dict = {}
all_word = []
```

```
for s in synonyms_list:
    if s[0] in word_dict and s[1] in word_dict:
        w1 = word_dict[s[0]]
        w2 = word_dict[s[1]]
        if w1 is not w2:
            w1.words.extend(w2.words)
            for w in w2.words:
                word_dict[w] = w1
            all_word.remove(w2)

    for i in range(2):
        if s[i] not in word_dict:
            if s[1 - i] in word_dict:
                word = word_dict[s[1 - i]]
            else:
                word = Word()
                all_word.append(word)
            word.add_word(s[i])
            word_dict[s[i]] = word

words_frequencies_list = parse_string(words_frequencies)
frequencies = {}
for w in words_frequencies_list:
    frequencies[w[0]] = int(w[1])
for f in frequencies:
    word_dict[f].add_frequencies(frequencies[f])
print_result = []

for a in all_word:
    a.words.sort()
    represent_word = a.words[0]
    print_result.append((represent_word, a.words_frequencies))

result = "Output: CF[]= {"
result += "(“" + print_result[0][0] + "”," + str(print_result[0][1]) + ")"
for i in range(1, len(print_result)):
    result += ",(“" + print_result[i][0] + "”," + str(print_result[i][1]) + ")"
result += "} of size " + str(len(print_result))
print(result)
```

Sample screenshots

Sample1:

```
Input file name:
sample1
Output: CF[]= {(“feet”,17),(“day”,11),(“fear”,3),(“big”,22),(“are”,23)} of size 5
```

Sample2:

```
Input file name:
sample2
Output: CF[]= {(“herbivore”,78),(“huge”,67),(“magic”,57)} of size 3
```

Sample3:

```
Input file name:
sample3
Output: CF[]= {(“herbivore”,78),(“huge”,67),(“magic”,57)} of size 3
```