HW2 Mining Contiguous Sequential Patterns in Text

Description

Implement a contiguous sequential pattern mining algorithm and apply it to text data to mine potential phrase candidates.

Input

The provided input file ("reviews_sample.txt") consists of 10,000 online reviews from Yelp users. The reviews have been stemmed to remove the postfix of each word so words with similar semantics can have the same form and most of the punctuation has been removed. So, each line is a list of strings separated by spaces.

An example line is provided below:

`cold cheap beer good bar food good service looking great pittsburgh style fish sandwich place breading light fish plentiful good side home cut fry good grilled chicken salad steak soup day homemade lot special great place lunch bar snack beer`

Output

You need to implement an algorithm to mine contiguous sequential patterns that are frequent in the input data. A contiguous sequential pattern is a sequence of items that frequently appears as a consecutive subsequence in a database of many sequences. For example, if the database is

```
A,B,A,C
A,C,A,B,A,B
B,A,A,C,D
```

and the minimum support is 2, then patterns like "A,B,A" or "A,C" are both frequent contiguous sequential patterns, while the pattern "A,A" is not a frequent contiguous sequential pattern because in the first two sequences the two A's are not consecutive to each other. Notice that it is still a frequent sequential pattern though.

Also, notice that multiple appearances of a subsequence in a single sequence record only counts once. For example, the pattern "A,B" appears 1 time in the first sequence and 2 times in the second, but its support should be calculated as 2, as there are only 2 records containing subsequence "A,B".

When implementing the algorithm, you could use any programming language you like. We only need your resulting pattern file, not your source code file.

Please set the relative minimum support to 0.01 and run it on the given text file. In other words, you need to extract all the frequent contiguous sequential patterns that have an absolute support no smaller than 100.

Please write all the frequent contiguous sequential patterns along with their absolute supports into a text file named "patterns.txt". Every line corresponds to exactly one pattern you found and should be in the following format:

support:item_1;item_2;item_3

Make sure that you **forma** each line correctly in the output file. For instance, use a semicolon instead of other characters to separate different items in the sequence.

For example, suppose the phrase "parking lot" has an absolute support 133, then the line corresponding to this frequent contiguous sequential pattern in "patterns.txt" should be:

133:parking;lot

Notice that the **order does matter** in sequential pattern mining. That is to say, `133:lot;parking`

may be graded as incorrect, even if `133:parking;lot` is a frequent contiguous sequential pattern.

Make sure you also include all the length-1 patterns.

Notes:

A counter tool is provided to support convenient and rapid tallies.
A *Counter* is a dict subclass for counting hashable objects.
It is a collection where elements are stored as dictionary keys and their counts are stored as dictionary values.
Counter objects have a dictionary interface except that they return a zero count for missing items. Counts are allowed to be any integer value including zero or negative counts.
As a dict subclass, Counter inherited the capability to remember insertion order.
Elements are counted from an iterable or initialized from another mapping. Counter objects support additional methods beyond those available for all dictionaries:

- elements()
- most_common([n])
- subtract([iter/mapping])
- total()
- fromkeys(iter)
- update([iter/mapping])

https://docs.python.org/3/library/collections.html#collections.Counter

strip(): Remove spaces at the beginning and at the end of the string
split(): splits a string into a *list* If sep is not specified, sep is any whitespace, runs of consecutive whitespace are regarded as a single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace.

Line 16 below is creating a subsequence from a list of words in the given range [i:i + length] and joining them with a space in between. **This is commonly used in sequential pattern mining to extract contiguous subsequences from a sequence of items.**

Line 17 below increments the count of the specific subsequence in the line_counts Counter.

In line 21 below Counter().items() converts the Counter object to a list of (elem, cnt) pairs.

Line 22 splits the string "sequence" into a list of substrings using the space character (' ') as the delimiter.

In [1]:

```python
from collections import Counter

sequence_counts = Counter()
min_support = 100

input_file = "reviews_sample.txt"
output_file = "patterns.txt"

with open(input_file, 'r') as file:
    for line in file:
        row = line.strip().split() # list
        line_counts = Counter()
        for seq_len in range(1, 3): #1,2, seq_len of seq
        # if 5 row in row/list and seq_len = 2, then range = 4, range(4) = 0-3
            for i in range(len(row) - seq_len + 1):
                subsequence = " ".join(row[i:i + seq_len])
                line_counts[subsequence] += 1
        for item in line_counts:
            sequence_counts[item] += 1

for sequence, count in sequence_counts.items():
    word = sequence.split(" ")
    if count >= min_support:
        out_string = (";".join (each for each in word))
        with open(output_file, 'a') as out_file:
            out_file.writelines(str(count)+":"+out_string + "\n")
```