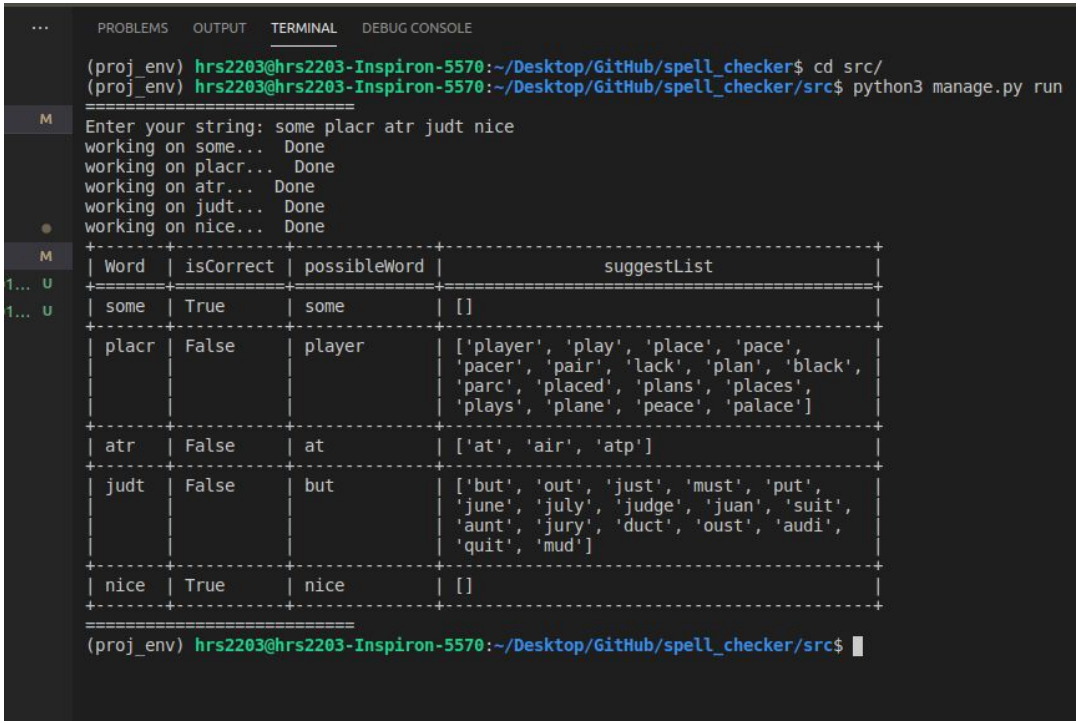
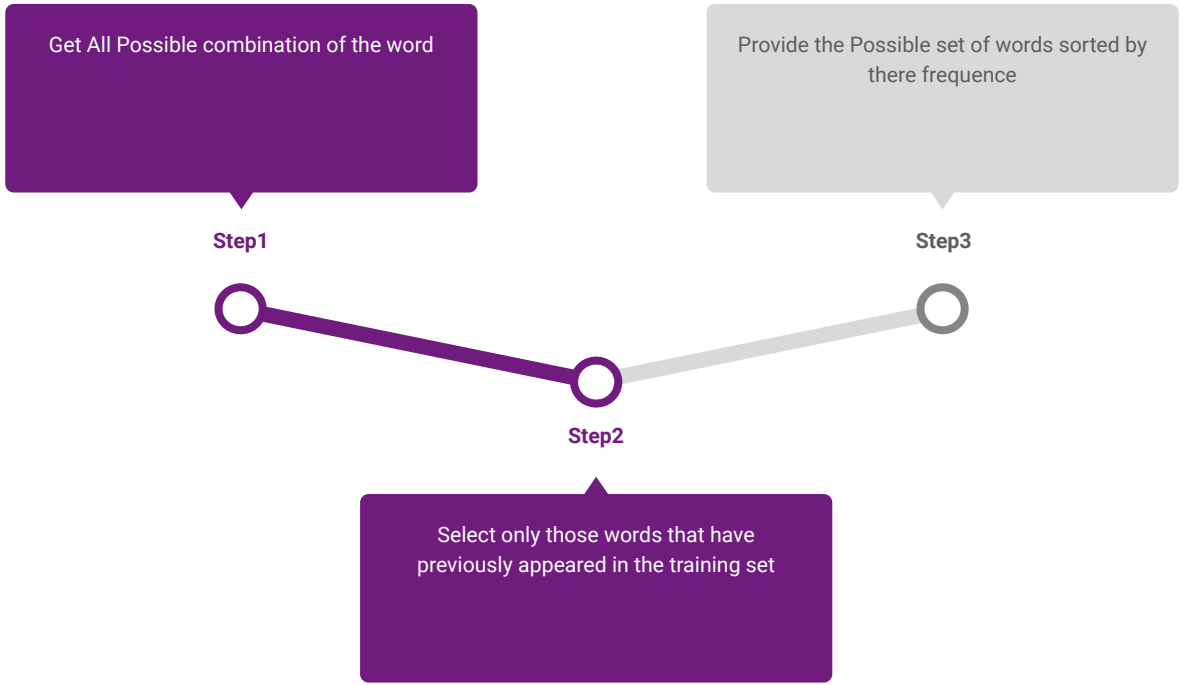
Spell Checker

By Hrishabh Pandey
S20180010064

Sample Input and Output



Word Suggestion Module



Generate Possible Combination of Word

Algorithm used: Peter norvig's Algo (http://norvig.com/spell-correct.html)

Generate word with edit distance <=2 with operation such as delete, insert, transpose, replace

Other Possible approach :

1. LingPipe (http://www.alias-i.com/lingpipe/demos/tutorial/querySpellChecker/read-me.html)
2. SymSpell (https://github.com/wolfgarbe/symspell)

Reason for not using above mentioned method:

Although the above algorithms provide significant jump is word suggestion, they require a huge data set to start with and they also have a high in memory (RAM) usage (as per my understanding.). My approach was to minimize memory consumption as much as possible and make the read/write operation from disk as much as possible making this algo much more scalable on devices with memory constraints.

Word Search Module

Word Storing pattern

1. There are total 78 files acting as our database.

2. Files are divides as a.txt, ai.txt, ar.txt and so on dividing the whole large set of words starting from a into 3 files.

3. Word storing format is `sample=10` where sample is the word stored and 10 is its occurrence frequency.

4. Words are stored in sorted format, this decrease the search time. Eg.

animal=5
air=10

Word Search method

1. File name is decided based on first 2 char of the word, eg, for word `sample` we will look into `s.txt`, for word ‘flight` we will look into `fi.txt`

2. The output from the file is `word=count` if word is found. Else its `word=0`.

3. The search time is O(n) in worst case, but since the files are divided, it is quick.

Training Module

Read File Module

1. Reads the training file, does preprocessing on the read string, and generate list of words

2. The generated word list is splitted into sublist based on the starting letter

3. The new sublist is also divided into 3 parts base on the file to which it belongs to

Eg. string : “are a airport bat” into
[
  [[a], [airport], [are]], [[bat], [ ], [ ]]
]

Write Module

1. Based on the input from splitted list the word is written down to the db set in appropriate file.

2. Writing rule
If (word is new): write(word=1)
else: write(word={previous count + 1})

3. Words are written in sorted order.

Conclusion

Additional Feature:

You can add a new word that you see fit using `python3 manage.py addWord` command

Resources: http://norvig.com/spell-correct.html

Conclusion:

As for now this program has many possible improvements and I plan to work on it afterwards. I would be very encouraged for as many remarks as possible.

Hrishabh Pandey

S20180010064