Spell Correction

We next look at the problem of correcting spelling errors in queries. For instance, we may wish to retrieve documents containing the term carrot when the user types the query carot.

We look at two steps to solving this problem: the first based on *edit distance* and the second based on *k-gram overlap*.

Levenshtein distance (LD) is a measure of the similarity between two strings, which we will refer to as the source string (s) and the target string (t). The **distance** is the number of deletions, insertions, or substitutions required to transform s into t.

Edit Distance : It is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other.

In python one package is there called autocorrect and having a method called spell. Which can correct maximum words.

METHOD

1 Find the all possible common word or common typo word else remain same.

To find this it create a lot of combinations of letters.

2 Import a large text file which contains approximately all words And we have the count of words.

3 Then it finds which word is used frequently it picks that word.

So It may not work always.

PYTHON CODE FOR SPELL CORRECTION

```
from autocorrect.nlp_parser import NLP_COUNTS, NLP_WORDS

AL = 'abcdefghijklmnopqrstuvwxyz'
class Word(object):
```

```
    def __init__(self, word):
        word_ = word.lower()
        slice_range = range(len(word_) + 1)
        self.slices = tuple((word_[:i], word_[i:])
                            for i in slice_range)
        self.word = word


    def typos(self):
        return (self.replaces() | self.inserts())

    def replaces(self):
        return {concat(a, c, b[1:])
            for a, b in self.slices[:-1]
            for c in AL}
    def inserts(self):
        return {concat(a, c, b)
            for a, b in self.slices
            for c in AL}

def common(words):
    return set(words) & NLP_WORDS

def correction(word, correction):
    if word.istitle():
        return correction.title()
    if word.isupper():
        return correction.upper()
    if len(word) > 2 and word[:2].isupper():
        return correction.title()

    return correction


def spell(word):
    w = Word(word)
    possiable_list = (common([word])or common(w.typos()) or [word])
    print(possiable_list)
    select_best = max(possiable_list,key = NLP_COUNTS.get)
    print(select_best)
```

return correction(word, select_best)

Output

```
]: spell('liste')

   {'lists', 'listen', 'listed'}
   listen

]: 'listen'
```

OR,
we can use
Autocorrect package for spell correction
Pip install autocorrect

```
from autocorrect import spell
spell('Gradiate')

'Graduate'
```