Spelling Error Detection and Correction

This program takes in a bunch of sentences. Identifies non-word spelling errors and correct them Using:

1. Bi-gram model
2. Kernighan Noisy Channel Model
3. Error Confusion Matrix
4. Damerau-Levenshtein Edit Distance

RUN

1. Unzip the code folder.
2. Go inside the src folder.
3. Run Python Main.py file.
   (If need to test different sentence please change Sentence 1 and Sentence 2.)

Help on class BiGramService in module BiGramService:

class BiGramService(__builtin__.object)

| Find the Unigram and Bigram Probability.

| Finds the Probability of a Sentence.

| Methods defined here:

| __init__(self)

| create_bigram(self, words)

| Create Unigram Model for words loaded.

| create_unigram(self, words)

| Create Unigram Model for words loaded.

| loadTrainingCorpus(self)

| Read a collection of word and build the training corpus

| probability(self, word, words='', type='uni_gram')

| Calculate the Maximum Likelihood Probability of n-Grams. With Add 1 Laplace Coefficient.

| sentenceprobability(self, sentence, type='uni_gram', form='anti-log')

| Calculate Maximum Likelihood Probability of a sentence.

| tokenize(self, text)

| ----------

| Data descriptors defined here:

| __dict__

| dictionary for instance variables (if defined)

| __weakref__

| list of weak references to the object (if defined)

| ----------

| Data and other attributes defined here:

| logger = <logging.Logger object>

Help on class WordCorrectionService in module WordCorrectionService:

class WordCorrectionService(__builtin__.object)

| Methods defined here:

| __init__(self, gram_service)

| channelProbability(self, x, y, edit)

| Method to calculate channel model probability for errors.

| createDictionaryCache(self, dict_list)

| editType(self, candidate, word)

| Calculate edit type for single edit errors.

| generateCandidates(self, word)

| Generate set of candidates for a given word using Damerau-Levenshtein Edit Distance.

| loadConfusionMatrix(self)

| Load Confusion Matrix from external data file.

| loadDictionary(self)

| Load dictionary from external data file.

| presentInDictionary(self, token)

| ----------

| Data descriptors defined here:

| __dict__

| dictionary for instance variables (if defined)

| __weakref__

| list of weak references to the object (if defined)

| --

| Data and other attributes defined here:

| logger = <logging.Logger object>

Help on class SpellingCorrectService in module SpellingCorrectService:

class SpellingCorrectService(__builtin__.object)

| Methods defined here:

| __init__(self)

| correctCase(self, sentence, correct_sentence)

| process(self, text)

| process_sentence(self, sentence)

| Get the sentence and return the corrected sentence

| --

| Data descriptors defined here:

| __dict__

| dictionary for instance variables (if defined)

| __weakref__

| list of weak references to the object (if defined)

| --

| Data and other attributes defined here:

| logger = <logging.Logger object>

Output

```
INFO:root:Loading Training Data to build word corpus ...
INFO:root:Processing Training Corpus
INFO:root:Creating Uni-gram count model ...
INFO:root:Calculated Count for Unigram Model
INFO:root:Creating Bi-gram Count Model ...
INFO:root:Calculated Count for Bigram Model
INFO:root:Loading dictionary from data file ...
INFO:__main__:Spelling Correction Service Loaded
```

```
INFO:__main__:Test 1
INFO:root:    Input Token : ['she', 'is', 'a', 'briliant', 'acress']
INFO:root:        Found Correct : she
INFO:root:        Found Correct : is
INFO:root:        Found Correct : a
INFO:root:        All Candidates for briliant : ['brilliant']
INFO:root:        Edit Type : brilliant: ('DELETION', 'l', '', 'i', 'il')
INFO:root:        All Candidates for acress : ['actress', 'acres', 'access', 'cress',
'across', 'agress']
INFO:root:        Edit Type : actress: ('DELETION', 't', '', 'c', 'ct')
INFO:root:        Edit Type : acres: ('INSERTION', '', 's', 'ss', 's')
INFO:root:        Edit Type : access: ('SUBSTITUTION', 'c', 'r', 'r', 'c')
INFO:root:        Edit Type : cress: ('INSERTION', '', 'a', '#a', '#')
INFO:root:        Edit Type : across: ('SUBSTITUTION', 'o', 'e', 'e', 'o')
INFO:root:        Edit Type : agress: ('SUBSTITUTION', 'g', 'c', 'c', 'g')
INFO:root:    Input Token : ['she', 'wno', 'tweve', 'oscar', 'awards', 'but', 'her',
'latest', 'mvie', 'is', 'bad']
INFO:root:        Found Correct : she
INFO:root:        All Candidates for wno : ['wao', 'ono', 'no', 'vno', 'wro', 'wgo',
'who', 'wnn', 'wo', 'woo', 'wpo', 'bno', 'wnt', 'wyo', 'wnx', 'wco', 'uno']
INFO:root:        Edit Type : wao: ('SUBSTITUTION', 'a', 'n', 'n', 'a')
INFO:root:        Edit Type : ono: ('SUBSTITUTION', 'o', 'w', 'w', 'o')
INFO:root:        Edit Type : no: ('INSERTION', '', 'w', '#w', '#')
INFO:root:        Edit Type : vno: ('SUBSTITUTION', 'v', 'w', 'w', 'v')
INFO:root:        Edit Type : wro: ('SUBSTITUTION', 'r', 'n', 'n', 'r')
INFO:root:        Edit Type : wgo: ('SUBSTITUTION', 'g', 'n', 'n', 'g')
INFO:root:        Edit Type : who: ('SUBSTITUTION', 'h', 'n', 'n', 'h')
INFO:root:        Edit Type : wnn: ('SUBSTITUTION', 'n', 'o', 'o', 'n')
INFO:root:        Edit Type : wo: None
INFO:root:        Edit Type : woo: ('SUBSTITUTION', 'o', 'n', 'n', 'o')
INFO:root:        Edit Type : wpo: ('SUBSTITUTION', 'p', 'n', 'n', 'p')
INFO:root:        Edit Type : bno: ('SUBSTITUTION', 'b', 'w', 'w', 'b')
INFO:root:        Edit Type : wnt: ('SUBSTITUTION', 't', 'o', 'o', 't')
INFO:root:        Edit Type : wyo: ('SUBSTITUTION', 'y', 'n', 'n', 'y')
INFO:root:        Edit Type : wnx: ('SUBSTITUTION', 'x', 'o', 'o', 'x')
INFO:root:        Edit Type : wco: ('SUBSTITUTION', 'c', 'n', 'n', 'c')
INFO:root:        Edit Type : uno: ('SUBSTITUTION', 'u', 'w', 'w', 'u')
INFO:root:        All Candidates for tweve : ['twelve', 'twere']
INFO:root:        Edit Type : twelve: ('DELETION', 'l', '', 'v', 'vl')
INFO:root:        Edit Type : twere: ('SUBSTITUTION', 'r', 'v', 'v', 'r')
INFO:root:        Found Correct : oscar
INFO:root:        Found Correct : awards
INFO:root:        Found Correct : but
INFO:root:        Found Correct : her
INFO:root:        Found Correct : latest
INFO:root:        All Candidates for mvie : ['movie', 'tvie', 'vie']
INFO:root:        Edit Type : movie: ('DELETION', 'o', '', 'm', 'mo')
INFO:root:        Edit Type : tvie: ('SUBSTITUTION', 't', 'm', 'm', 't')
INFO:root:        Edit Type : vie: ('INSERTION', '', 'm', '#m', '#')
INFO:root:        Found Correct : is
INFO:root:        Found Correct : bad
INFO:__main__:Input Text : She is a briliant acress. She wno Tweve Oscar awards but
her latest mvie is bad.
INFO:__main__:Result     : She is a brilliant actress. She wao Twelve Oscar awards but
her latest movie is bad.
INFO:__main__:            5.32300019264 Seconds to compute
INFO:__main__:Test 2
INFO:root:    Input Token : ['she', 'is', 'a', 'beaiful', 'actress']
INFO:root:        Found Correct : she
INFO:root:        Found Correct : is
INFO:root:        Found Correct : a
INFO:root:        All Candidates for beaiful : []
INFO:root:        Found Correct : actress
```

```
INFO:root:    Input Token : ['her', 'moveie', 'did', 'very', 'well', 'at', 'the',
'box', 'ofice']
INFO:root:        Found Correct : her
INFO:root:        All Candidates for moveie : ['movie']
INFO:root:        Edit Type : movie: ('INSERTION', '', 'e', 'ie', 'i')
INFO:root:        Found Correct : did
INFO:root:        Found Correct : very
INFO:root:        Found Correct : well
INFO:root:        Found Correct : at
INFO:root:        Found Correct : the
INFO:root:        Found Correct : box
INFO:root:        All Candidates for ofice : ['office']
INFO:root:        Edit Type : office: ('DELETION', 'f', '', 'o', 'of')
INFO:__main__:Input Text : She is a beaiful actress. Her moveie did very well at the
box ofice.
INFO:__main__:Result     : She is a beaiful actress. Her movie did very well at the
box office.
INFO:__main__:             0.905999898911 Seconds to compute
```