Credit Hours System

CMPN402

Machine Intelligence

Cairo University

Faculty of Engineering

Programming Assignment Report

Problem 2

Submitted to: Dr. Nevin Darwish

Submitted by: Tarek Niazy Taha

Student ID: 4180307

Problem No.: Problem 2 (22.1)

a- The attached files are:

- dataset.txt: it contains a 125k words
- nlp.py: this file contains the code

b- Relationship between the files:

Inside nlp.py file dataset.txt is opened and its data is loaded to a variable named <u>dataset</u>

c- How to run the code:

-At the end of the file (nlp.py), the text is assigned to a variable called ‘s’, and then call <u>print(Split_with_no_Spaces(s))</u>

which sends the variable s after converting its characters to lower case sends it to Split_with_no_Spaces() function which implements the logic of the Viterbi algorithm.

d- Explaining the algorithm:

- At the beginning of the cell, file dataset.txt is opened and loaded to a variable called dataset,
- Then unigram model is created, I calculated the cost of each word in the dataset by using;
  log(Rank*log(length(dataset)))
- I assumed that the most frequent word takes the lowest cost
  i.e.: the: log(1*log(125000))=0.707
- The unigram model is a dictionary which holds each word and their corresponding cost. (t is sorted in ascending order from the lowest cost to the highest one)
- Get the length of the longest word and set it to a variable called maximum_word_len

- Then Function Split_with_no_spaces(String) which takes the string that is going to be splitted; this function contains of three parts :

    1. First define a variable called historyTable which is used to save the costs of the possible combination of characters of the given string, then it takes the string as input parameter and iterate over it and gives each index of the string to a function called WordMatch(the second part),this part is responsible for getting the word that has a minimum cost among possible words .

    2. WordMatch(index) takes the index of the of the string, and then define a variable called possible words which saves inside it the index beside the past indices and check if its corresponding character matches with any word in the unigram model if not puts infinity, and then choses the word that has the minimum cost among the possible words.

    3. Backtracking; this part is responsible for backtracking over the history table and also backtrack over the string in order to get the matched minimum cost word, It is based on Conclusion that the last item in the history table is already the cost of the right matched word, So it gets the length of that word by backtrack over the string and then Subtracting it from the original length of the string and using that value as an index to get the next right matched word

that has the minimum cost then do this operation all over again till it finishes the string, and then printing the matched splitted words.

- **Test Cases:**

```
s = 'thelongestlistofthelongeststuffatthelongestdomainnameat
longlast.com'



out = ['the', 'longest', 'list', 'of', 'the', 'longest',
'stuff', 'at', 'the', 'longest', 'domain', 'name', 'at',
'long', 'last', '.', 'com'],
```

```
s = 'thumbgreenappleactiveassignmentweeklymetaphor.'

Out= ['thumb', 'green', 'apple', 'active', 'assignment',
'weekly', 'metaphor', '.']
```

```
s = 'thereismassesoftextinformationofpeoplescommentswhichisp
arsedfromhtmlbuttherearenodelimitedcharactersinthemforexampl
ethumbgreenappleactiveassignmentweeklymetaphorapparentlyther
earethumbgreenappleetcinthestringialsohavealargedictionaryto
querywhetherthewordisreasonablesowhatsthefastestwayofextract
ionthxalot.'

Out = ['there', 'is', 'masses', 'of', 'text', 'information',
'of', 'peoples', 'comments', 'which', 'is', 'parsed',
'from', 'html', 'but', 'there', 'are', 'no', 'delimited',
'characters', 'in', 'them', 'for', 'example', 'thumb',
'green', 'apple', 'active', 'assignment', 'weekly',
'metaphor', 'apparently', 'there', 'are', 'thumb', 'green',
'apple', 'etc', 'in', 'the', 'string', 'i', 'also', 'have',
'a', 'large', 'dictionary', 'to', 'query', 'whether', 'the',
'word', 'is', 'reasonable', 'so', 'whats', 'the', 'fastest',
'way', 'of', 'extraction', 'thx', 'a', 'lot', '.']
```

```
s = 'itwasadarkandstormynighttherainfellintorrentsexceptatoc
casionalintervalswhenitwascheckedbyaviolentgustofwindwhichsw
eptupthestreetsforitisinlondonthatoursceneliesrattlingalongt
hehousetopsandfiercelyagitatingthescantyflameofthelampsthats
truggledagainstthedarkness.'

Out=
['it', 'was', 'a', 'dark', 'and', 'stormy', 'night', 'the',
'rain', 'fell', 'in', 'torrents', 'except', 'at',
'occasional', 'intervals', 'when', 'it', 'was', 'checked',
'by', 'a', 'violent', 'gust', 'of', 'wind', 'which',
'swept', 'up', 'the', 'streets', 'for', 'it', 'is', 'in',
'london', 'that', 'our', 'scene', 'lies', 'rattling',
'along', 'the', 'housetops', 'and', 'fiercely', 'agitating',
'the', 'scanty', 'flame', 'of', 'the', 'lamps', 'that',
'struggled', 'against', 'the', 'darkness', '.']
```

```
s = 'Priorresearchpointstoefficientidentificationofembeddedw
ordsasakeyfactorinfacilitatingthereadingoftextprintedwithout
spacingbetweenwordsHerewefurthertestedtheprimaryroleofbottom
upwordidentificationbyalteringthisprocesswithalettertranspos
itionmanipulationIntwoexperimentsweexaminedsilentreadingandr
eadingaloudofnormalsentencesandsentencescontainingwordswithl
ettertranspositionsinbothnormallyspacedandunspacedconditions
Wepredictedthatlettertranspositionsshouldbeparticularlyharmf
ulforreadingunspacedtextInlinewithourpredictionthemajorityof
ourmeasuresofreadingfluencyshowedthatunspacedtextwithlettert
ranspositionswasdisproportionatelydifficulttoreadThesefindin
gsprovidefurthersupportfortheclaimthatreadingtextwithoutbetw
eenwordspacingreliesprincipallyonefficientbottomupprocessing
enablingaccuratewordidentificationintheabsenceofvisualcuesto
identifywordboundariesPriorresearchpointstoefficientidentifi
cationofembeddedwordsasakeyfactorinfacilitatingthereadingoft
extprintedwithoutspacingbetweenwordsHerewefurthertestedthepr
imaryroleofbottomupwordidentificationbyalteringthisprocesswi
thalettertranspositionmanipulationIntwoexperimentsweexamined
silentreadingandreadingaloudofnormalsentencesandsentencescon
tainingwordswithlettertranspositionsinbothnormallyspacedandu
```

```
nspacedconditionsWepredictedthatlettertranspositionsshouldbe
particularlyharmfulforreadingunspacedtextInlinewithourpredic
tionthemajorityofourmeasuresofreadingfluencyshowedthatunspac
edtextwithlettertranspositionswasdisproportionatelydifficult
toreadThesefindingsprovidefurthersupportfortheclaimthatreadi
ngtextwithoutbetweenwordspacingreliesprincipallyonefficientb
ottomupprocessingenablingaccuratewordidentificationintheabse
nceofvisualcuestoidentifywordboundaries.'
```

```
Out = ['prior', 'research', 'points', 'to', 'efficient',
'identification', 'of', 'embedded', 'words', 'as', 'a',
'key', 'factor', 'in', 'facilitating', 'the', 'reading',
'of', 'text', 'printed', 'without', 'spacing', 'between',
'words', 'here', 'we', 'further', 'tested', 'the',
'primary', 'role', 'of', 'bottom', 'up', 'word',
'identification', 'by', 'altering', 'this', 'process',
'with', 'a', 'letter', 'transposition', 'manipulation',
'in', 'two', 'experiments', 'we', 'examined', 'silent',
'reading', 'and', 'reading', 'aloud', 'of', 'normal',
'sentences', 'and', 'sentences', 'containing', 'words',
'with', 'letter', 'transpositions', 'in', 'both',
'normally', 'spaced', 'and', 'un', 'spaced', 'conditions',
'we', 'predicted', 'that', 'letter', 'transpositions',
'should', 'be', 'particularly', 'harmful', 'for', 'reading',
'un', 'spaced', 'text', 'inline', 'with', 'our',
'prediction', 'the', 'majority', 'of', 'our', 'measures',
'of', 'reading', 'fluency', 'showed', 'that', 'un',
'spaced', 'text', 'with', 'letter', 'transpositions', 'was',
'disproportionately', 'difficult', 'to', 'read', 'these',
'findings', 'provide', 'further', 'support', 'for', 'the',
'claim', 'that', 'reading', 'text', 'without', 'between',
'word', 'spacing', 'relies', 'principally', 'on',
'efficient', 'bottom', 'up', 'processing', 'enabling',
'accurate', 'word', 'identification', 'in', 'the',
'absence', 'of', 'visual', 'cues', 'to', 'identify', 'word',
'boundaries', 'prior', 'research', 'points', 'to',
'efficient', 'identification', 'of', 'embedded', 'words',
'as', 'a', 'key', 'factor', 'in', 'facilitating', 'the',
'reading', 'of', 'text', 'printed', 'without', 'spacing',
'between', 'words', 'here', 'we', 'further', 'tested',
'the', 'primary', 'role', 'of', 'bottom', 'up', 'word',
'identification', 'by', 'altering', 'this', 'process',
'with', 'a', 'letter', 'transposition', 'manipulation',
'in', 'two', 'experiments', 'we', 'examined', 'silent',
'reading', 'and', 'reading', 'aloud', 'of', 'normal',
'sentences', 'and', 'sentences', 'containing', 'words',
'with', 'letter', 'transpositions', 'in', 'both',
'normally', 'spaced', 'and', 'un', 'spaced', 'conditions',
'we', 'predicted', 'that', 'letter', 'transpositions',
'should', 'be', 'particularly', 'harmful', 'for', 'reading',
'un', 'spaced', 'text', 'inline', 'with', 'our',
'prediction', 'the', 'majority', 'of', 'our', 'measures',
'of', 'reading', 'fluency', 'showed', 'that', 'un',
'spaced', 'text', 'with', 'letter', 'transpositions', 'was',
'disproportionately', 'difficult', 'to', 'read', 'these',
'findings', 'provide', 'further', 'support', 'for', 'the',
'claim', 'that', 'reading', 'text', 'without', 'between',
'word', 'spacing', 'relies', 'principally', 'on',
'efficient', 'bottom', 'up', 'processing', 'enabling',
'accurate', 'word', 'identification', 'in', 'the',
'absence', 'of', 'visual', 'cues', 'to', 'identify', 'word',
'boundaries', '.']
```

- **<u>Assumptions:</u>**
    - The input string should be free from any spaces, hyphens, punctuations, and abbreviation and names.
    - I Assumed that the most frequent word has the minimum cost, So, in the Viterbi algorithms I always check for the word that gives me the minimum cost

- **<u>Performance Measure:</u>**
    - Runtime: longest runtime = 0.36 seconds
    - Its performance is related to the size of the data set

- If there is un defined word in the input string this algorithm will try to break it down to sub words that corresponding to the words already defined in the data set if non defined it will break it to its letters.
- The law that is used in calculating the cost
- I perform a pre-processing to the input text I remove any white spaces and all null characters.
- Till now the accuracy tends to be high as I handled almost most of the cases that makes the model crashes, but as I already mentioned it depends on the data contained on the data set, but it also fails when a text like it’s or let’s you’re is entered.