Assignment 1 **Supun Fernando – 219332D**

Github Link: https://github.com/SupunFernando/Assignment1_IR

Part 1: Tokenization

Used tokenizers

- Student feedback data – nltk.tokenize.word_tokenize
- Twitter data – nltk.tokenize.casual.TweetTokenizer
- Research paper data – nltk.tokenize.word_tokenize

Word_tokenize

It tokenizes a string to split off punctuation. This function in Python splits a given sentence into words using the NLTK library

*nltk.tokenize.**word_tokenize**(text, language='english', preserve_line=False)*

TweetTokenizer

Twitter-aware tokenizer, designed to be flexible and easy to adapt to new domains and tasks. In tokenizing tweets TweetTokenizer will be used prominently.

*nltk.tokenize.casual.**TweetTokenizer**(preserve_case=True, reduce_len=False, strip_handles=False)*

Discussion

Part 1: Tokenization

Tweet tokenization is been selected as the optimal tokenization type in twitter data tokenization purpose, since it considers the attributes of tweeter data such as hashtags, emojis, and tweeter related social media texts and styles. Hence, tweetTokenizer, which specifically built for the very on purpose accurately tokenize this domain compared with other alternatives.

For student feedback data and research paper data, which include punctuation, line breaks and whitespaces, word tokenize type has been selected because it's capability in breaking strings and separating meaningful sentences considering punctuations, whitespaces etc.

Twitter data

```
#read twitter.txt file
tweeterFile = open("C:\\Users\\Supun\\Desktop\\Supun Backup\\Supun\\transfer\\documents\\myfiles\\MScDA\\DS\\IR\\Assignment_1\\tw
twitterdata = tweeterFile.read()

#tokenize
tweet_tokenizer = TweetTokenizer()
token_twitter = tweet_tokenizer.tokenize(twitterdata)

print(token_twitter)
```

```
['Reminds', 'me', 'of', 'Liberal', 'Immigration', 'Fraudster', 'Monsef', 'avoiding', 'deportation', 'from', 'Canada', '.', '#cd
npoli', '#LPC', '#CPCLDR', '', '', '_', 'https://t.co/ZOZOSe1CqQ', '#immigration', '#integration', '#canada', 'https://t.co/
M5cKGyvV8F', 'We', 'want', 'controlled', 'immigration', 'that', 'contributes', 'positively', 'to', 'the', 'UK', 'economy', '.',
'Same', 'as', 'Australia', '&', 'Canada', '.', 'https://t.co/99mYliuOes', 'Is', 'the', 'new', 'Manitoba', 'immigration', 'fee',
'a', 'head', 'tax', '?', 'https://t.co/LsG7C3vLe9', 'Canada', 'immigration', 'profit', 'influence', 'modernistic', 'delhi', 'ye
t', 'abhinav', ':', 'XKofy', 'https://t.co/becgusY2i6', 'Canada', 'Immigration', 'Minister', 'to', '', '', '', 'Substantia
lly', 'Increase', 'Immigration', 'Numbers', 'https://t.co/nEFw30MRaa', 'https://t.co/cyI867PZRV', 'M', '', '', 'me', 'les',
'#USA', '=p', 'ays', "d'immigration", 'par', 'excellence', 'CONTR', '', '', 'LE', 'RIGOUREUSEMENT', "l'immigration", 'et',
```

Student feedback data

```
#read student_feedback.txt file
feedbackFile = open("C:\\Users\\Supun\\Desktop\\Supun Backup\\Supun\\transfer\\documents\\myfiles\\MScDA\\DS\\IR\\Assignment_1\\s
feedbackdata = feedbackFile.read()

#tokenize
token_feedback = nltk.word_tokenize(feedbackdata)

print(token_feedback)
```

```
['Honestly', 'last', 'seven', 'lectures', 'are', 'good', '.', 'Lectures', 'are', 'understandable', '.', 'Lecture', 'slides', 'a
re', 'very', 'useful', 'to', 'self-study', 'also', '.', 'The', 'given', 'opportunity', 'to', 'ask', 'questions', 'from', 'the',
'lecturer', 'is', 'appreciative', '.', '```', 'Good', ':', ')', '<', 'br', '/', '>', 'please', 'do', 'recap', 'at', 'class', 'st
arting', 'it', '&', '#', '039', ';', 's', 'better', 'for', 'us', '.', '<', 'br', '/', '>', 'sometimes', 'teaching', 'speed', 'i
s', 'very', 'high', '.', '<', 'br', '/', '>', '<', 'br', '/', '>', 'Thanks', '!', ':', ')', '<', 'br', '/', '>', '```', 'The',
'lectures', 'are', 'good', '..', 'but', 'a', 'bit', 'speed.A', 'in', 'class', 'working', 'activity', 'is', 'a', 'must', 'one.S
o', 'please', 'take', 'another', 'hour', 'in', 'thursdays', 'madame.', "'", '<', 'br', '/', '>', 'We', 'can', 'hear', 'your',
```

Research paper data

```
#read research_paper.txt file
researchFile = open("C:\\Users\\Supun\\Desktop\\Supun Backup\\Supun\\transfer\\documents\\myfiles\\MScDA\\DS\\IR\\Assignment_1\\r
researchdata = researchFile.read()

#tokenize
token_research = nltk.word_tokenize(researchdata)

print(token_research)
```

```
['Neural', 'network', 'models', 'have', 'shown', 'their', 'promising', 'opportunities', 'for', 'multi-task', 'learning', ',',
'which', 'focus', 'on', 'learning', 'the', 'shared', 'layers', 'to', 'extract', 'the', 'common', 'and', 'task-invariant', 'feat
ures', '.', 'However', ',', 'in', 'most', 'existing', 'approaches', ',', 'the', 'extracted', 'shared', 'features', 'are', 'pron
e', 'to', 'be', 'contaminated', 'by', 'task-specific', 'features', 'or', 'the', 'noise', 'brought', 'by', 'other', 'tasks',
'.', 'In', 'this', 'paper', ',', 'we', 'propose', 'an', 'adversarial', 'multi-task', 'learning', 'framework', ',', 'alleviatin
g', 'the', 'shared', 'and', 'private', 'latent', 'feature', 'spaces', 'from', 'interfering', 'with', 'each', 'other', '.', 'W
e', 'conduct', 'extensive', 'experiments', 'on', '16', 'different', 'text', 'classification', 'tasks', ',', 'which', 'demonstra
```

Part 2: Isolated word correction & Context sensitive word correction

Isolated word correction

In the case of word getting misspelled, the most correct and list of likely words are been suggested in this section.

Python Library: pyspellchecker

It uses a Levenshtein Distance algorithm to find permutations within an edit distance of 2 from the original word. It then compares all permutations (insertions, deletions, replacements, and transpositions) to known words in a word frequency list. Those words that are found more often in the frequency list are more likely the correct results (source: https://pyspellchecker.readthedocs.io) It supports multiple languages including English, Spanish, German, French, and Portuguese.

Context sensitive word correction

Python Library: Symspell

Classsymspellpy.symspellpy.SymSpell(max_dictionary_edit_distance=2, prefix_length=7, count_threshold=1)

Symmetric Delete spelling correction algorithm. initial_capacity from the original code is omitted since python cannot preallocate memory. compact_mask from the original code is omitted since we're not mapping suggested corrections to hash codes. Using word_segmentation method divides a string into words by inserting missing spaces at the appropriate positions misspelled words are corrected and do not affect segmentation; existing spaces are allowed and considered for optimum segmentation. (source: https://symspellpy.readthedocs.io/en/latest/api/symspellpy)

TextBlob's Spelling Correction is used also in the code but it is on sentence wise not in token terms. Hence in this study as acceptable approach Symspell based algorithm have incorporated.

Discussion

Part 2.1: Isolated word correction

In processing text in the domain of information retrieval word spelling checking matters. Inbuilt python library – pyspellchecker is used in spell checking as an algorithm.

Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question.

In the used text datasets of twitter, student_feedback and research_paper, most of the words are been spell checked and corrected. As an exceptional case it can be found that twitter dataset hashtag has been eliminated at its most cases by using pyspellchecker.

Twitter data – sample

```
: #twitter data
  wordCorrection(token_twitter)
```

```
campagne -> mostly liked answer : champagne
Mostly Likely Options : {'campagnes', 'champagne'}


know-all -> mostly liked answer : knowall
Mostly Likely Options : {'knowall'}


#cdnpoli -> mostly liked answer : #cdnpoli
Mostly Likely Options : {'#cdnpoli'}


https://t.co/4k84ee8y63 -> mostly liked answer : https://t.co/4k84ee8y63
Mostly Likely Options : {'https://t.co/4k84ee8y63'}


#immigration -> mostly liked answer : immigration
Mostly Likely Options : {'immigration'}
```

Student feedback data

```
#Student feedback data
wordCorrection(token_feedback)
```

```
madame. -> mostly liked answer : madame
Mostly Likely Options : {'madames', 'madame'}


undersatand -> mostly liked answer : understand
Mostly Likely Options : {'understand'}


examples.lectures -> mostly liked answer : examples.lectures
Mostly Likely Options : {'examples.lectures'}


speed.a -> mostly liked answer : speed
Mostly Likely Options : {'speed', 'speedee', 'speedos', 'spenda', 'speeder', 'speedo', 'speeded', 'speedway', 'speedoo', '
eed's", 'speeds', 'speedy'}


interesting.we -> mostly liked answer : interesting.we
Mostly Likely Options : {'interesting.we'}
```

Research paper data

```
#Research paper data
wordCorrection(token_research)
```

```
constraints.specifically -> mostly liked answer : constraints.specifically
Mostly Likely Options : {'constraints.specifically'}


task-dependent -> mostly liked answer : task-dependent
Mostly Likely Options : {'task-dependent'}


luong -> mostly liked answer : long
Mostly Likely Options : {'ulong', 'long', 'lung', 'loong'}


task-invariant -> mostly liked answer : task-invariant
Mostly Likely Options : {'task-invariant'}


�infantile� -> mostly liked answer : infantile
Mostly Likely Options : {'infantile'}


shared-private -> mostly liked answer : shared-private
Mostly Likely Options : {'shared-private'}


off-the-shelf -> mostly liked answer : off-the-shelf
Mostly Likely Options : {'off-the-shelf'}


2016c -> mostly liked answer : 2016c
Mostly Likely Options : {'2016c'}
```

Part 2.2: Context sensitive word correction

Symmetric Delete spelling correction algorithm - Symspell which provides much higher speed and lower memory consumption. Context-sensitive spelling correction is the task of correcting spelling errors which result in valid words. Context-sensitive spelling correction is the task of correcting spelling errors which result in valid words. Superior feature of this line is correcting the errors using the reference of a suitable corpus. Here in this study have used word.txt dictionary file as the corpus. It is a proven evidence that after tokenized text processing with Context sensitive word correction is more precise with corrected words as text tokens.

Twitter data

```
#twitter data

print(correct_tokenized_text(token_twitter))
```

```
['Reminds', 'me', 'of', 'Liberal', 'Immi', 'Frauds', 'Monsey', 'a', 'deport', 'from', 'Canada', 'a', 'cdn', 'lpc', 'cp', 'a',
'a', 'a', 'ttp', 'dimming', 'winter', 'canada', 'ttp', 'We', 'want', 'control', 'immi', 'that', 'contrib', 'positive', 'to', 't
he', 'Uk', 'economy', 'a', 'Same', 'as', 'Austral', 'a', 'Canada', 'a', 'ttp', 'Is', 'the', 'new', 'Manitoba', 'immi', 'fee',
'a', 'head', 'tax', 'a', 'ttp', 'Canada', 'immi', 'profit', 'in', 'modern', 'delhi', 'yet', 'abhinaya', 'a', 'Azofy', 'ttp', 'C
anada', 'Immi', 'Minister', 'to', 'a', 'a', 'a', 'Subst', 'In', 'Immi', 'Numbers', 'ttp', 'ttp', 'M', 'a', 'a', 'me', 'les', 'd
usa', 'up', 'ays', 'dimming', 'par', 'excel', 'Contr', 'a', 'a', 'Le', 'Rigour', 'immi', 'et', 'acc', 'a', 'a', 's', 'a', 'a',
'la', 'green', 'a', 'a', 'a', 'a', 'ttp', 'pshaw', 'what', 'changes', 'should', 'be', 'made', 'to', "Canada's", 'immi', 'laws',
'due', 'to', 'the', 'influx', 'of', 'immi', 'and', 'violent', 'a', 'L', 'a', 'a', 'immi', 'irra', 'a', 'a', 'gulix', 'a', 'a',
're', 'au', 'Canada', 'd', 'a', 'a', 'corti', 'a', 'a', 'e', 'en', '5', 'question', 'ttp', 'Immi', 'irra', 'a', 'a', 'gulix',
'a', 'a', 're', 'au', 'Canada', 'd', 'a', 'a', 'corti', 'a', 'a', 'e', 'en', '5', 'question', 'ttp', 'ttp', 'ttp', 'ttp', 'Wil
l', 'Media', 'ask', 'the', 'Liberal', 'if', 'they', 'actual', 'have', 'a', 'solid', 'plan', 'for', 'Canada', 'a', 'a', 'a',
```

Student feedback data

```
#student feedback data

print(correct_tokenized_text(token_feedback))
```

```
['Honest', 'last', 'seven', 'lecture', 'are', 'good', 'a', 'Lecture', 'are', 'undrest', 'a', 'Lecture', 'slides', 'are', 'ver
y', 'useful', 'to', 'self', 'also', 'a', 'The', 'given', 'apport', 'to', 'ask', 'question', 'from', 'the', 'lecture', 'is', 'ap
prend', 'a', 'a', 'Good', 'a', 'a', 'a', 'br', 'a', 'a', 'please', 'do', 'recap', 'at', 'class', 'starting', 'it', 'a', 'a',
'9', 'a', 's', 'better', 'for', 'us', 'a', 'a', 'br', 'a', 'a', 'some', 'teaching', 'speed', 'is', 'very', 'high', 'a', 'a', 'b
r', 'a', 'a', 'a', 'br', 'a', 'a', 'Thanks', 'a', 'a', 'a', 'a', 'br', 'a', 'a', 'a', 'The', 'lecture', 'are', 'good', 'a', 'bu
t', 'a', 'bit', 'speed', 'in', 'class', 'working', 'a', 'is', 'a', 'must', "one's", 'please', 'take', 'another', 'hour', 'in',
'thurs', "madame'", "'t", 'a', 'br', 'a', 'a', 'We', 'can', 'hear', 'your', 'voice', 'clearly', 'and', 'can', 'undrest', 'th
e', 'things', 'you', 'teach', 'a', 'Present', 'slides', 'also', 'good', 'source', 'to', 'refer', 'a', 'lf', 'you', 'can', 'do',
'more', 'example', 'question', 'within', 'the', 'class', 'and', 'it', 'will', 'help', 'us', 'to', 'undrest', 'the', 'principe',
'well', 'a', 'a', 'br', 'a', "a'", "'", 'Lecture', 'was', 'well', 'struct', 'and', 'well', 'or', 'a', 'It', 'was', 'easy', 't
```

Research paper data

```
#student feedback data

print(correct_tokenized_text(token_research))
```

```
['Neural', 'network', 'models', 'have', 'shown', 'their', 'promise', 'apport', 'for', 'multi', 'learning', 'a', 'which', 'focu
s', 'on', 'learning', 'the', 'shared', 'layers', 'to', 'extract', 'the', 'common', 'and', 'tasking', 'feature', 'a', 'However',
'a', 'in', 'most', 'existing', 'approach', 'a', 'the', 'extract', 'shared', 'feature', 'are', 'prone', 'to', 'be', 'contam', 'b
y', 'tasks', 'feature', 'or', 'the', 'noise', 'brought', 'by', 'other', 'tasks', 'a', 'In', 'this', 'paper', 'a', 'we', 'propos
e', 'an', 'adversa', 'multi', 'learning', 'frame', 'a', 'alluvia', 'the', 'shared', 'and', 'private', 'latent', 'feature', 'spa
ces', 'from', 'interne', 'with', 'each', 'other', 'a', 'We', 'conduct', 'ex', 'expert', 'on', '16', 'differ', 'text', 'classi
c', 'tasks', 'a', 'which', 'demon', 'the', 'benefit', 'of', 'our', 'approach', 'a', 'Besides', 'a', 'we', 'show', 'that', 'th
e', 'shared', 'know', 'learned', 'by', 'our', 'pro', 'model', 'can', 'be', 'regard', 'as', 'off', 'know', 'and', 'easily', 'tra
ns', 'to', 'new', 'tasks', 'a', 'Multi', 'learning', 'is', 'an', 'effect', 'approach', 'to', 'improve', 'the', 'perform', 'of',
```

Part 3: Stemmer and Lemmatize

Stemmer: PorterStemmer

It stems or removes the edges of the common endings to words so that they can be resolved to a common form.

Lemmatizer: WordNetLemmatizer

Wordnet is a large, freely and publicly available lexical database for the English language aiming to establish structured semantic relationships between words

(source: https://www.machinelearningplus.com/nlp/lemmatization-)

Discussion

Part 3: Stemmer and Lemmatize

Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors.

Lemmatization is slower than stemming because it uses a corpus to supply lemma. And to get the proper lemma a parts-of-speech have to define. And also it is harder to create a new lemmatizer for new language than a stemming algorithm. In the meantime Stemming precision is low and recall is higher because it gets all documents which contain root words, but those all do not contain correct value.

Stemmer

Twitter data

```
#Stemmatizer
```

```
import nltk

from nltk.stem import PorterStemmer
```

```
twitter_words= token_twitter
ps =PorterStemmer()
for w in twitter_words:
    rootWord=ps.stem(w)
    print(w + ": " +rootWord)
```

```
Reminds: remind
me: me
of: of
Liberal: liber
Immigration: immigr
Fraudster: fraudster
Monsef: monsef
avoiding: avoid
deportation: deport
from: from
Canada: canada
```

Student feedback data

```
feedback_words= token_feedback
ps =PorterStemmer()
for w in feedback_words:
    rootWord=ps.stem(w)
    print(w + ": " +rootWord)
```

```
Honestly: honestli
last: last
seven: seven
lectures: lectur
are: are
good: good
.: .
Lectures: lectur
are: are
understandable: understand
 .
```

Research paper data

```
research_words= token_research
ps =PorterStemmer()
for w in research_words:
    rootWord=ps.stem(w)
    print(w + ": " +rootWord)
```

```
Neural: neural
network: network
models: model
have: have
shown: shown
their: their
promising: promis
opportunities: opportun
for: for
multi-task: multi-task
learning: learn
```

Lemmatize

Twitter data

```
#Lemmatizer
```

```
from nltk.stem import WordNetLemmatizer
```

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Supun\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.
```

```
True
```

```
wordnet_lemmatizer = WordNetLemmatizer()

tokenization = nltk.word_tokenize(twitterdata)
for w in tokenization:
    print("Lemma for {} : {}".format(w, wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma for Reminds : Reminds
Lemma for me : me
Lemma for of : of
Lemma for Liberal : Liberal
Lemma for Immigration : Immigration
Lemma for Fraudster : Fraudster
Lemma for Monsef : Monsef
Lemma for avoiding : avoiding
Lemma for deportation : deportation
Lemma for from : from
Lemma for Canada : Canada
```

Student feedback data

```
wordnet_lemmatizer = WordNetLemmatizer()

tokenization = nltk.word_tokenize(feedbackdata)
for w in tokenization:
    print("Lemma for {} : {}".format(w, wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma for Honestly : Honestly
Lemma for last : last
Lemma for seven : seven
Lemma for lectures : lecture
Lemma for are : are
Lemma for good : good
Lemma for . : .
Lemma for Lectures : Lectures
Lemma for are : are
Lemma for understandable : understandable
Lemma for . : .
Lemma for Lecture : Lecture
```

Research paper data

```
wordnet_lemmatizer = WordNetLemmatizer()

tokenization = nltk.word_tokenize(researchdata)
for w in tokenization:
    print("Lemma for {} : {}".format(w, wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma for Neural : Neural
Lemma for network : network
Lemma for models : model
Lemma for have : have
Lemma for shown : shown
Lemma for their : their
Lemma for promising : promising
Lemma for opportunities : opportunity
Lemma for for : for
Lemma for multi-task : multi-task
Lemma for learning : learning
Lemma for , : ,
Lemma for which : which
```