Computer Engineering Department

Natural Language Processing

Assignment 1

Ali Sedaghi
97521378

Table of contents

Theoretical questions **1**
- Q1- Natural language processing tools 1
  - Natural Language ToolKit (NLTK) 1
  - wordninja 1
  - scikit-learn 1
  - spaCy 1
- **Q2- Regular expression** **1**
  - Phone number 1
  - Date in format dd-mm-yyyy 2
  - URLs of .ir and .org 2
  - Vehicle registration plates 2
- **Q3- Maximum matching word segmentation** **2**

Practical questions **3**
- Q1- Regex 3
- Q2- NLTK intro 3
- Q3- Normalization 4
  - Using NLTK and re 5
  - Using pySpellChecker (Bonus) 5
- Q4- Word tokenization 6
  - Part A 6
  - Part B 6
  - Part C: TreebankWordTokenizer 7
  - Part D: RegexpTokenizer 8
  - Part E: WhitespaceTokenizer 8
  - Part F: WordPunctTokenizer 9
- Q5- Stemming 9
  - Part A: Stemmers in NLTK 9
  - Part B: Porter vs Lancaster 10
  - Part C: Lemmatization 11
  - Part D 11
- Q6- Data preprocessing 11
  - Part A: Word cloud 14
  - Part B: Top-10 trends chart 15

Theoretical questions

Q1- Natural language processing tools

1. Natural Language ToolKit (NLTK)

Is one of the most popular tools in NLP. It's contained with valuable corpus, text processors, helper functions, etc. It is also open source and people can contribute to source code. Some helpful modules inside NLTK are: parsers, tokenizers, stemming tools, tagging, etc. NLTK requires Python version 3.7, 3.8, 3.9 or 3.10.
https://www.nltk.org

2. wordninja

Probabilistically split concatenated words using NLP based on English Wikipedia uni-gram frequencies. It is one of the best choices for N-gram. This tool is also open source.
https://github.com/keredson/wordninja

3. scikit-learn

It is a machine learning tool. There are many functionalities for working with text data inside this tool. For example you can load datasets such as newsgroups, build your sentiment analysis pipeline, train a classifier, extract features from text data, identify language, etc.
https://scikit-learn.org/

4. spaCy

spaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython. The library is published under the MIT license and its main developers are Matthew Honnibal and Ines Montani, the founders of the software company Explosion[1].
https://spacy.io/

Q2- Regular expression

A. Phone number

```
^(09|00989|\+989)[0-9]{1,16}$
```

[1] https://en.wikipedia.org/wiki/SpaCy

B. Date in format dd-mm-yyyy

^(0[1-9]|[1-2][0-9]|3[0-1])-(0[1-9]|1[0-2])-([0-1][0-9]{3}|20[0-1][0-9]|202[0-2])$

C. URLs of .ir and .org

^(http://|https://)?www\.[a-zA-Z]([a-zA-Z0-9]*)\.(org|ir)(/?)$

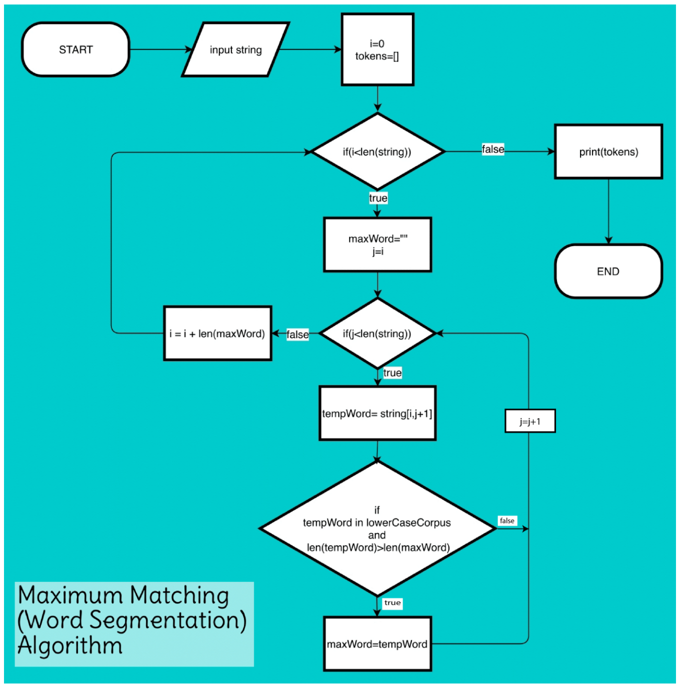
D. Vehicle registration plates

^([0-9][0-9])[A-Z]([0-9][0-9]{2})([A-Z][A-Z])([0-9][0-9])$

Q3- Maximum matching word segmentation[2]

We use this algorithm for separating words in a text which does not have any delimiter such as space or comma.
thisisatextwithnospace -> this is a text with no space
Algorithm:

1. Start with first letter: c
2. Find longest word that starts with c
3. If there is a match then mark a boundary and separate that word. Else set that single letter as a word.
4. Go to the next character then go to step 2 until the text is finished.

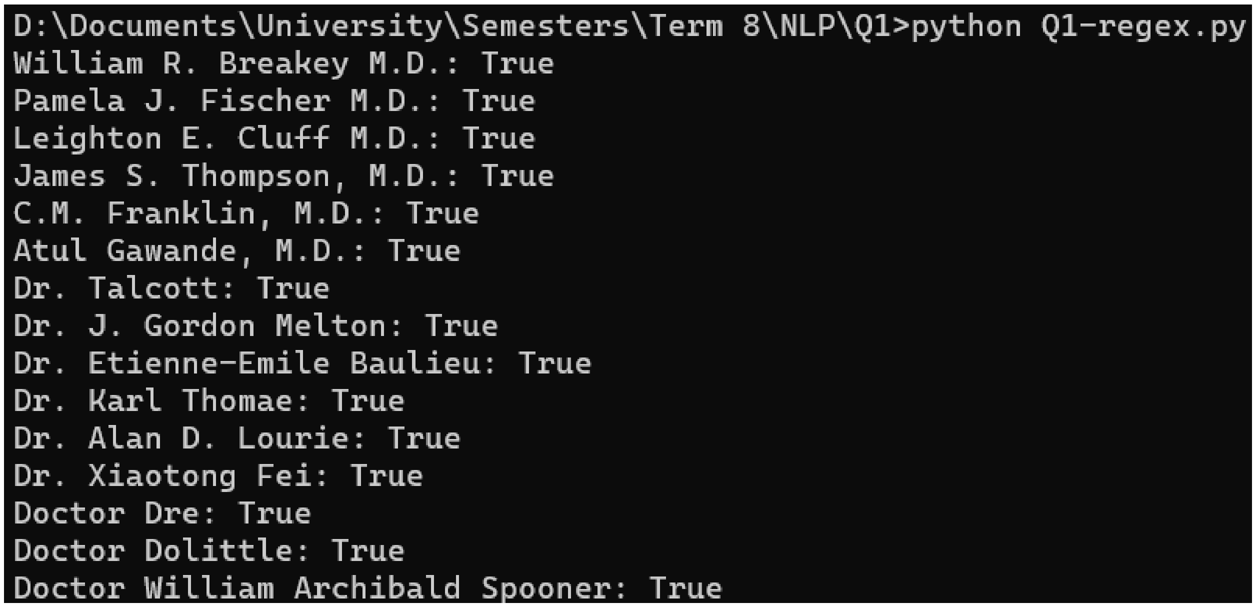


2 https://medium.com/@anshul16/maximum-matching-word-segmentation-algorithm-python-code-3444fe4bd6f9

Practical questions

Q1- Regex

Source code and input files are inside the Q1 folder.
I used python built-in library for regex i.e re library. I used the match function to check whether a string is accepted by a pattern or not.
All samples are started with Dr. or Doctor or finished with M.D.

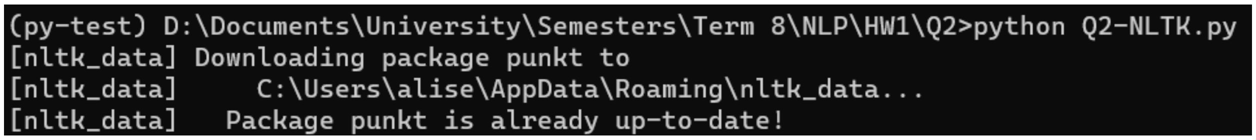


```
D:\Documents\University\Semesters\Term 8\NLP\Q1>python Q1-regex.py
William R. Breakey M.D.: True
Pamela J. Fischer M.D.: True
Leighton E. Cluff M.D.: True
James S. Thompson, M.D.: True
C.M. Franklin, M.D.: True
Atul Gawande, M.D.: True
Dr. Talcott: True
Dr. J. Gordon Melton: True
Dr. Etienne-Emile Baulieu: True
Dr. Karl Thomae: True
Dr. Alan D. Lourie: True
Dr. Xiaotong Fei: True
Doctor Dre: True
Doctor Dolittle: True
Doctor William Archibald Spooner: True
```

Q2- NLTK intro

Source code and input files are inside the Q2 folder.
First we install the nltk library with the command below. We also need to download punkt

pip install nltk
nltk.download('punkt')

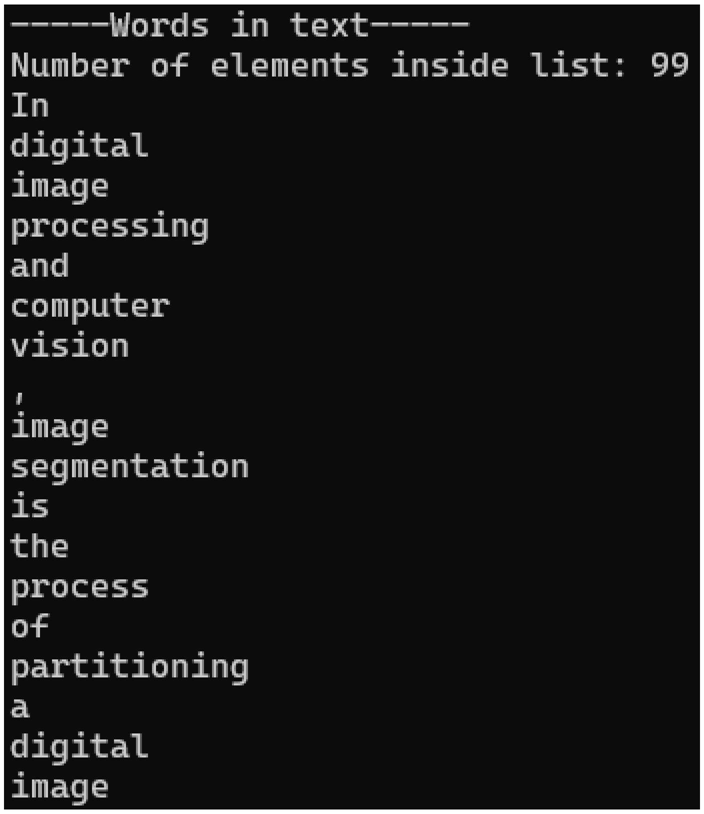


```
(py-test) D:\Documents\University\Semesters\Term 8\NLP\HW1\Q2>python Q2-NLTK.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\alise\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

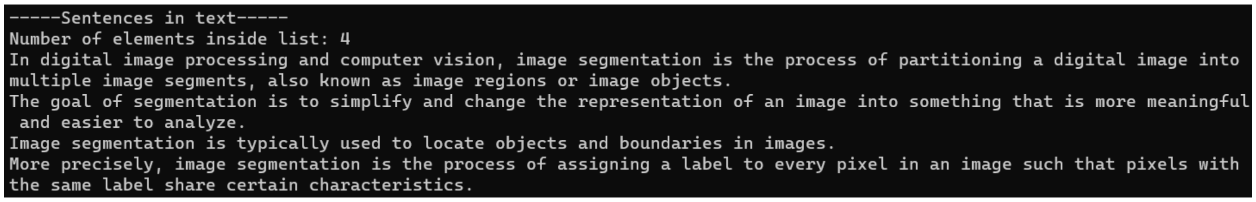
Then we use word_tokenize and sent_tokenize as tokenizers. Tokenizers divide strings into lists of substrings. For example, tokenizers can be used to find the words and punctuation in a string[3] or find sentences in a text.
word_tokenize: We use the word_tokenize() method to split a sentence into tokens or words.
sent_tokenize: We use the sent_tokenize() method to split a document or paragraph into sentences.[4]

[3] https://www.nltk.org/api/nltk.tokenize.html
[4] https://www.analyticsvidhya.com/blog/2019/07/how-get-started-nlp-6-unique-ways-perform-tokenization

Corpus:[5]
In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple image segments, also known as image regions or image objects.
The goal of segmentation is to simplify and change the representation of an image into something that is more meaningful and easier to analyze.
Image segmentation is typically used to locate objects and boundaries in images.
More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.



```
-----Words in text-----
Number of elements inside list: 99
In
digital
image
processing
and
computer
vision
,
image
segmentation
is
the
process
of
partitioning
a
digital
image
```



```
-----Sentences in text-----
Number of elements inside list: 4
In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into
multiple image segments, also known as image regions or image objects.
The goal of segmentation is to simplify and change the representation of an image into something that is more meaningful
 and easier to analyze.
Image segmentation is typically used to locate objects and boundaries in images.
More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with
the same label share certain characteristics.
```

Q3- Normalization

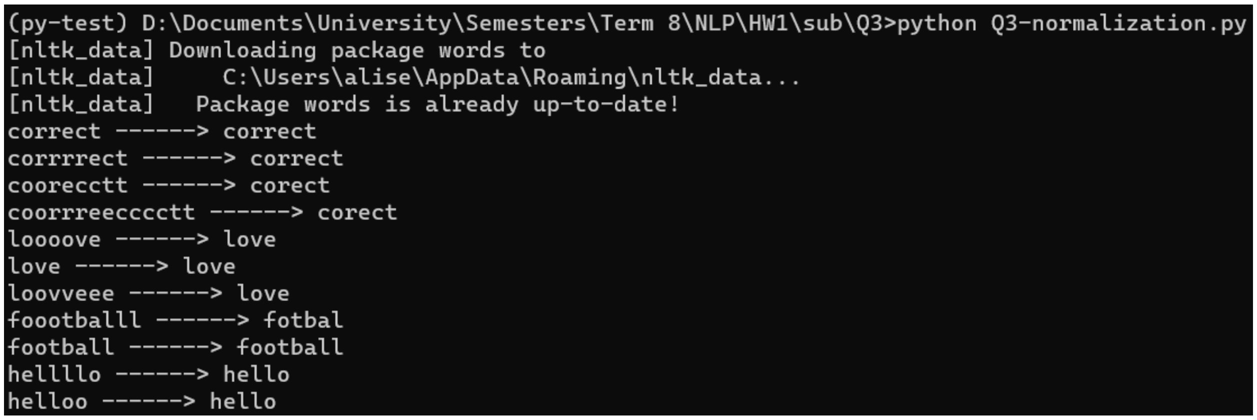
Source code and input files are inside the Q3 folder.

[5] https://en.wikipedia.org/wiki/Image_segmentation

Using NLTK and re

NLTK words dictionary is used to check if a word is valid or not.
Python re library and compile function is used to reduce a pattern of repeated characters. For example if we a character k times we will consider k, k-1, …, k occurrences of that character recursively.

re.compile(r'(\w*)(\w)\2(\w*)').sub(r'\1\2\3', word)



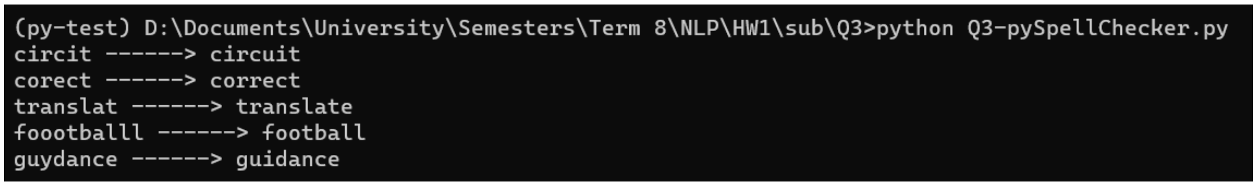
```
(py-test) D:\Documents\University\Semesters\Term 8\NLP\HW1\sub\Q3>python Q3-normalization.py
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\alise\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
correct ------> correct
corrrrect ------> correct
cooreccctt ------> corect
coorrreeccccctt ------> corect
loooove ------> love
love ------> love
loovveee ------> love
foootballl ------> fotbal
football ------> football
hellllo ------> hello
helloo ------> hello
```

There are some interesting results. Both forms of corect[6] and correct are inside words in the dictionary. Also fotbal and football.

Using pySpellChecker (Bonus)

There is a python package named pyspellchecker[7] which corrects misspelled words.

pip install pyspellchecker



```
(py-test) D:\Documents\University\Semesters\Term 8\NLP\HW1\sub\Q3>python Q3-pySpellChecker.py
circit ------> circuit
corect ------> correct
translat ------> translate
foootballl ------> football
guydance ------> guidance
```

As you can see the results are very good. This library is also more powerful than our NLTK code with re package. It's able to correct misspelled words like guydance to guidance or circit to circuit.

[6] https://en.wiktionary.org/wiki/corect
[7] https://pypi.org/project/pyspellchecker

Q4- Word tokenization

Source code, input and output files are inside the Q4 folder.

Part A

First we create a dictionary containing file paths.

```
tokenize = {
    'AlbertEinstein.txt': None,
    'Shahnameh.txt': None,
    'ShortSampleEnglish.txt': None,
    'ShortSamplePersian.txt': None,
}
```

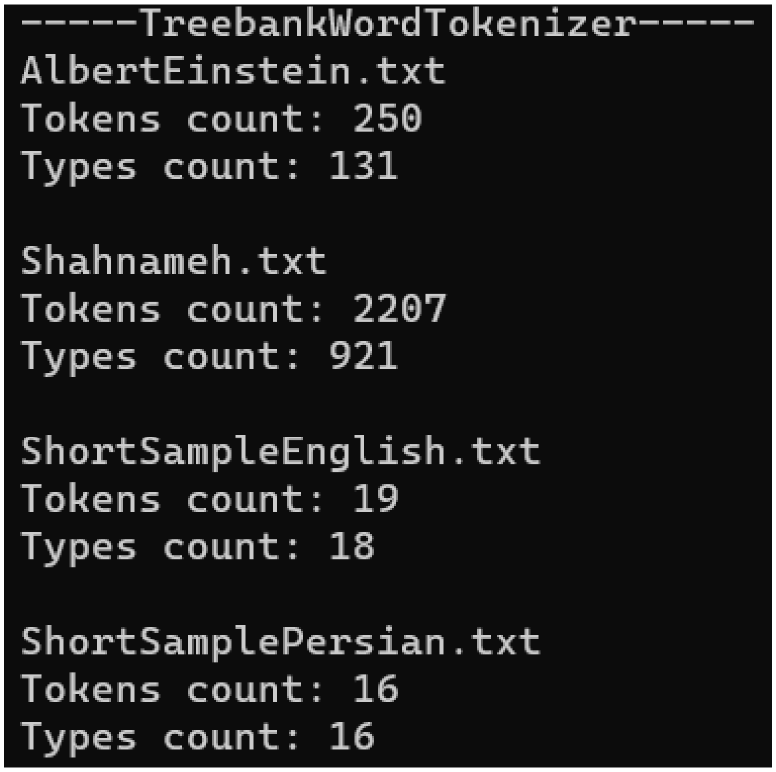
Part B

Set each entry’s value inside the dictionary to a Tokenizer class.

```
class Tokenize:
    def __init__(self, path):
        self.text = open(path, mode='r', encoding='utf-8').read()
    def treebank_word_tokenizer(self):
        return TreebankWordTokenizer().tokenize(self.text)
    def regexp_tokenizer(self, pattern, gaps=True):
        return RegexpTokenizer(pattern, gaps=gaps).tokenize(self.text)
    def whitespace_tokenizer(self):
        return WhitespaceTokenizer().tokenize(self.text)
    def word_punct_tokenizer(self):
        return WordPunctTokenizer().tokenize(self.text)

# Init Tokenize classes
for path in tokenize.keys():
    tokenize[path] = Tokenize(path)
```

Part C: TreebankWordTokenizer

```python
print("-----TreebankWordTokenizer-----")
paths = tokenize.keys()
for path in paths:
    tokens = tokenize[path].treebank_word_tokenizer()
    print(path)
    print(f"Tokens count: {len(tokens)}")
    print(f"Types count: {get_types_count(tokens)}")
    print()
```



```
-----TreebankWordTokenizer-----
AlbertEinstein.txt
Tokens count: 250
Types count: 131

Shahnameh.txt
Tokens count: 2207
Types count: 921

ShortSampleEnglish.txt
Tokens count: 19
Types count: 18

ShortSamplePersian.txt
Tokens count: 16
Types count: 16
```

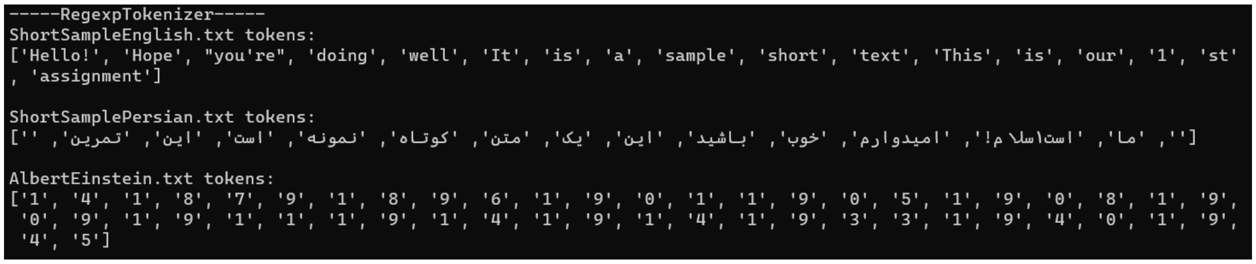
Part D: RegexpTokenizer

RegexpTokenizer(pattern, gaps=gaps)

Pattern: regular expression term. ‘\s+|\.’ and ‘[0-9]’

Gaps: if true, find patterns between words. Else, find patterns inside words.

```
print("-----RegexpTokenizer-----")
path = 'ShortSampleEnglish.txt'
tokens = tokenize[path].regexp_tokenizer(pattern='\s+|\.')
print(f"{path} tokens:")
print(tokens)
path = 'ShortSamplePersian.txt'
tokens = tokenize[path].regexp_tokenizer(pattern='\s+|\.')
print(f"\n{path} tokens:")
print(tokens)
path = 'AlbertEinstein.txt'
tokens = tokenize[path].regexp_tokenizer(pattern='[0-9]', gaps=False)
print(f"\n{path} tokens:")
print(tokens)
```



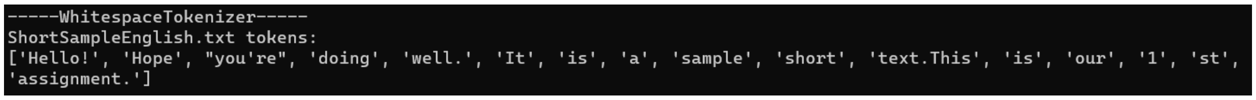
```
-----RegexpTokenizer-----
ShortSampleEnglish.txt tokens:
['Hello!', 'Hope', "you're", 'doing', 'well', 'It', 'is', 'a', 'sample', 'short', 'text', 'This', 'is', 'our', '1', 'st'
, 'assignment']

ShortSamplePersian.txt tokens:
['', 'ما', 'استاسلا م!', 'امیدوارم', 'خوب', 'باشید', 'این', 'یک', 'متن', 'کوتاه', 'نمونه', 'است', 'این', 'تمرین', '']

AlbertEinstein.txt tokens:
['1', '4', '1', '8', '7', '9', '1', '8', '9', '6', '1', '9', '0', '1', '1', '9', '0', '5', '1', '9', '0', '8', '1', '9',
 '0', '9', '1', '9', '1', '1', '1', '9', '1', '4', '1', '9', '1', '4', '1', '9', '3', '3', '1', '9', '4', '0', '1', '9',
 '4', '5']
```

Part E: WhitespaceTokenizer

It simply splits words with \n \t \s.

RegexpTokenizer: pattern=”\n|\t|\s” gaps=True

```
print("\n-----WhitespaceTokenizer-----")
path = 'ShortSampleEnglish.txt'
tokens = tokenize[path].whitespace_tokenizer()
print(f"{path} tokens:")
print(tokens)
```

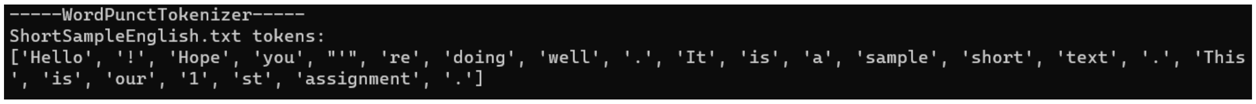


```
-----WhitespaceTokenizer-----
ShortSampleEnglish.txt tokens:
['Hello!', 'Hope', "you're", 'doing', 'well.', 'It', 'is', 'a', 'sample', 'short', 'text.This', 'is', 'our', '1', 'st',
'assignment.']
```

Part F: WordPunctTokenizer

Tokenizes based on punctuations.
. ! ? “ ‘

```
print("\n-----WordPunctTokenizer-----")
path = 'ShortSampleEnglish.txt'
tokens = tokenize[path].word_punct_tokenizer()
print(f"{path} tokens:")
print(tokens)
```

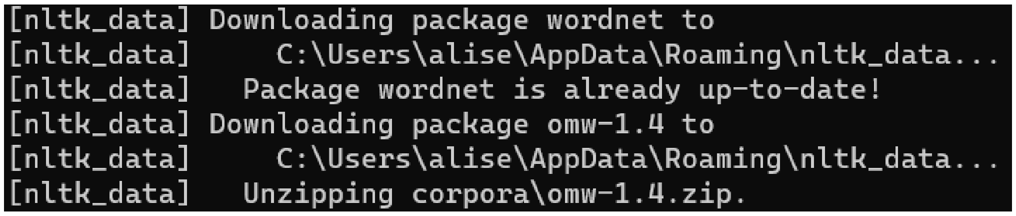


```
-----WordPunctTokenizer-----
ShortSampleEnglish.txt tokens:
['Hello', '!', 'Hope', 'you', "'", 're', 'doing', 'well', '.', 'It', 'is', 'a', 'sample', 'short', 'text', '.', 'This
', 'is', 'our', '1', 'st', 'assignment', '.']
```

Q5- Stemming

Source code, input and output files are inside the Q4 folder.
We need to download wordnet and omw-1.4

nltk.download('wordnet')
nltk.download('omw-1.4')



```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\alise\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\alise\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\omw-1.4.zip.
```

Part A: Stemmers in NLTK

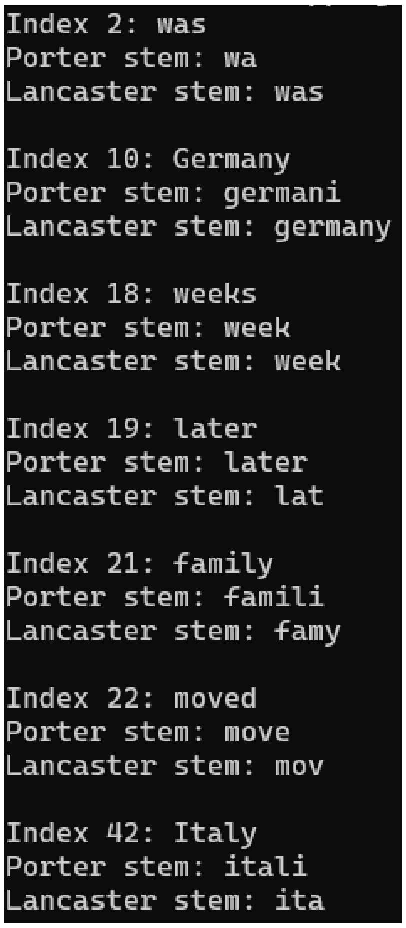
In NLTK there are some APIs for stemming. These are inside nltk.stem. We imported PorterStemmer and LancasterStemmer. In order to use these stemmers we need to create an instance of that stemmer.

```
porter = PorterStemmer()
lancaster = LancasterStemmer()
```

Then we should call the stem(word) function of those instances.

Part B: Porter vs Lancaster

```
print("-----Stemming-----")
porter = PorterStemmer()
lancaster = LancasterStemmer()
text = open('../Q4/AlbertEinstein.txt', mode='r', encoding='utf-8').read()
tokens = TreebankWordTokenizer().tokenize(text)
for i in [2, 10, 18, 19, 21, 22, 42]:
    w = tokens[i]
    print(f"Index {i}: {w}")
    print(f"Porter stem: {porter.stem(w)}")
    print(f"Lancaster stem: {lancaster.stem(w)}")
    print()
```

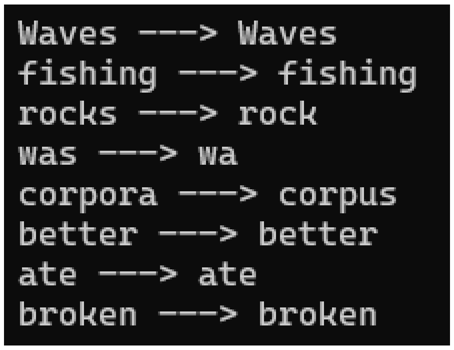


```
Index 2: was
Porter stem: wa
Lancaster stem: was

Index 10: Germany
Porter stem: germani
Lancaster stem: germany

Index 18: weeks
Porter stem: week
Lancaster stem: week

Index 19: later
Porter stem: later
Lancaster stem: lat

Index 21: family
Porter stem: famili
Lancaster stem: famy

Index 22: moved
Porter stem: move
Lancaster stem: mov

Index 42: Italy
Porter stem: itali
Lancaster stem: ita
```

Lancaster stemmings are not very accurate. In some cases the words are changed to an unrelated word. It usually removes the end part of the word. Although it has better results in “was” and “germany”..
Porter converts “y” to “i” but this conversion is not always right. Porter stemmed “was” to “wa” which is wrong.

Part C: Lemmatization

```
print("-----Lemmatization-----")
lemmatizer = WordNetLemmatizer()
words = ["Waves", "fishing", "rocks", "was", "corpora", "better", "ate",
"broken"]
for w in words:
    print(f"{w} ---> {lemmatizer.lemmatize(w)}")
```



```
Waves ---> Waves
fishing ---> fishing
rocks ---> rock
was ---> wa
corpora ---> corpus
better ---> better
ate ---> ate
broken ---> broken
```

Part D

Lemmatization for nouns is in a good state. But it has not changed any verbs. We should set the “pos” argument as “v” in order to lemmatize verbs in a better way. But now we don’t have good results in nouns. We should set that argument based on each word to get the best results.
In order to find the type of a word we can use NLTK tagging functions.

Q6- Data preprocessing

Source code, input and output files are inside the Q6 folder.
Step 1: Remove white spaces
White spaces are redundant and useless to our language model and we must remove them.

Before	After
RT @GOPChairwoman: The economic boom continues! 273K jobs added in February 7M+ jobs added since @realDonaldTrump was elected Unemploymâ€¦	RT @GOPChairwoman: The economic boom continues! 273K jobs added in February 7M+ jobs added since @realDonaldTrump was elected Unemploymâ€¦
THANK YOU OHIO! #VOTE https://t.co/5EJjFZVLHZ	THANK YOU OHIO! #VOTE https://t.co/5EJjFZVLHZ
RT @RyanAFournier: Look at the size of this crowd down Pennsylvania Ave. More than a MILLION people showed up to support this President.â€¦	RT @RyanAFournier: Look at the size of this crowd down Pennsylvania Ave. More than a MILLION people showed up to support this President.â€¦

Step 2: Lower casing

Our language model must not differ between Hello and hello. Because they are the same.

Before	After
Republicans and Democrats have both created our economic problems.	republicans and democrats have both created our economic problems.
I was thrilled to be back in the Great city of Charlotte, North Carolina with thousands of hardworking American Patriots who love our Country, cherish our values, respect our laws, and always put AMERICA FIRST! Thank you for a wonderful evening!! #KAG2020 https://t.co/dNJZfRsl9y	i was thrilled to be back in the great city of charlotte, north carolina with thousands of hardworking american patriots who love our country, cherish our values, respect our laws, and always put america first! thank you for a wonderful evening!! #kag2020 https://t.co/dnjzfrsl9y
RT @CBS_Herridge: READ: Letter to surveillance court obtained by CBS News questions where there will be further disciplinary action and choâ€¦	rt @cbs_herridge: read: letter to surveillance court obtained by cbs news questions where there will be further disciplinary action and choâ€¦

Step 3: Remove handles

Usernames are special words and they are meaningless. For example my Telegram username is @nikomadol but this word has no meaning. We should remove them.

Before	After
rt @mzhemingway: very friendly telling of events here about comey's apparent leaking to compliant media. if you read those articles and thoâ€¦	rt very friendly telling of events here about comey's apparent leaking to compliant media. if you read those articles and thoâ€¦
rt @erictrump: https://t.co/ncrndosfiv	rt https://t.co/ncrndosfiv
rt @gopchairwoman: the economic boom continues! 273k jobs added in february 7m+ jobs added since @realdonaldtrump was elected unemploymâ€¦	rt the economic boom continues! 273k jobs added in february 7m+ jobs added since was elected unemploymâ€¦

Step 4: Remove punctuations and special characters

These characters are not words. So they don't have meaning in the language model. Because of this reason we remove them.

Before	After
rt very friendly telling of events here about comey's apparent leaking to compliant media. if you read those articles and thoâ€¦	rt very friendly telling of events here about comeys apparent leaking to compliant media if you read those articles and tho
rt https://t.co/ncrndosfiv	rt httpstconcrndosfiv
rt the economic boom continues! 273k jobs added in february 7m+ jobs added since was elected unemploymâ€¦	rt the economic boom continues k jobs added in february m jobs added since was elected unemploym

Step 5: Tokenization

We used TreebankWordTokenizer for tokenizing. It uses some patterns to get tokens inside text.

Before	After
rt very friendly telling of events here about comeys apparent leaking to compliant media if you read those articles and tho	['rt', 'very', 'friendly', 'telling', 'of', 'events', 'here', 'about', 'comeys', 'apparent', 'leaking', 'to', 'compliant', 'media', 'if', 'you', 'read', 'those', 'articles', 'and', 'tho']
rt httpstconcrndosfiv	['rt', 'httpstconcrndosfiv']
rt the economic boom continues k jobs added in february m jobs added since was elected unemploym	['rt', 'the', 'economic', 'boom', 'continues', 'k', 'jobs', 'added', 'in', 'february', 'm', 'jobs', 'added', 'since', 'was', 'elected', 'unemploym']

Step 6: Remove stop words

We used NLTK stop words. Stop words are words which have little value to us. They are words such as “a”, “and”, “the”, etc. These words don’t give us much information about corpus so we remove them.

Before	After
rt very friendly telling of events here about comeys apparent leaking to compliant media if you read those articles and tho	rt friendly telling events comeys apparent leaking compliant media read articles tho
rt httpstconcrndosfiv	rt httpstconcrndosfiv
rt the economic boom continues k jobs added in february m jobs added since was elected unemploym	rt economic boom continues k jobs added february jobs added since elected unemploym

Step 7: Remove short words

Words that are less than 3 characters are usually invalid words. They are usually created by mistakes inside the tokenizer. For example words such as “rt”, “k”, “m”, etc don’t give us any useful information. So we remove them.

Before	After
rt friendly telling events comeys apparent leaking compliant media read articles tho	friendly telling events comeys apparent leaking compliant media read articles tho
rt httpstconcrndosfiv	httpstconcrndosfiv
rt economic boom continues k jobs added february jobs added since elected unemploym	economic boom continues jobs added february jobs added since elected unemploym

Step 8: Stemming

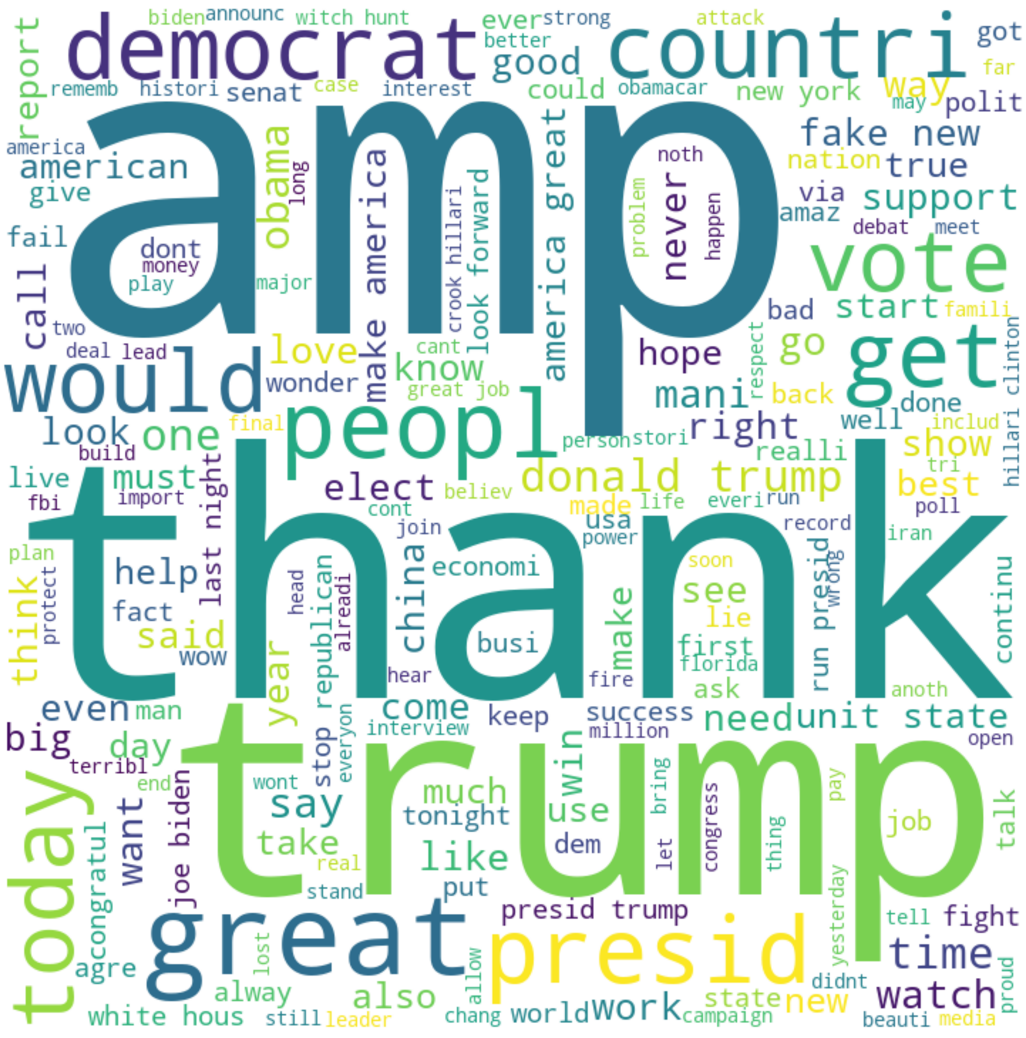
Lancaster stemmings are not very accurate. In some cases the words are changed to an unrelated word. It usually removes the end part of the word. Although it has better results in “was” and “germany”..

Porter converts “y” to “i” but this conversion is not always right. Porter stemmed “was” to “wa” which is wrong.

Before	After
friendly telling events comeys apparent leaking compliant media read articles tho	friendli tell event comey appar leak compliant media read articl tho
httpstconcrndosfiv	httpstconcrndosfiv
economic boom continues jobs added february jobs added since elected unemploym	econom boom continu job ad februari job ad sinc elect unemploym

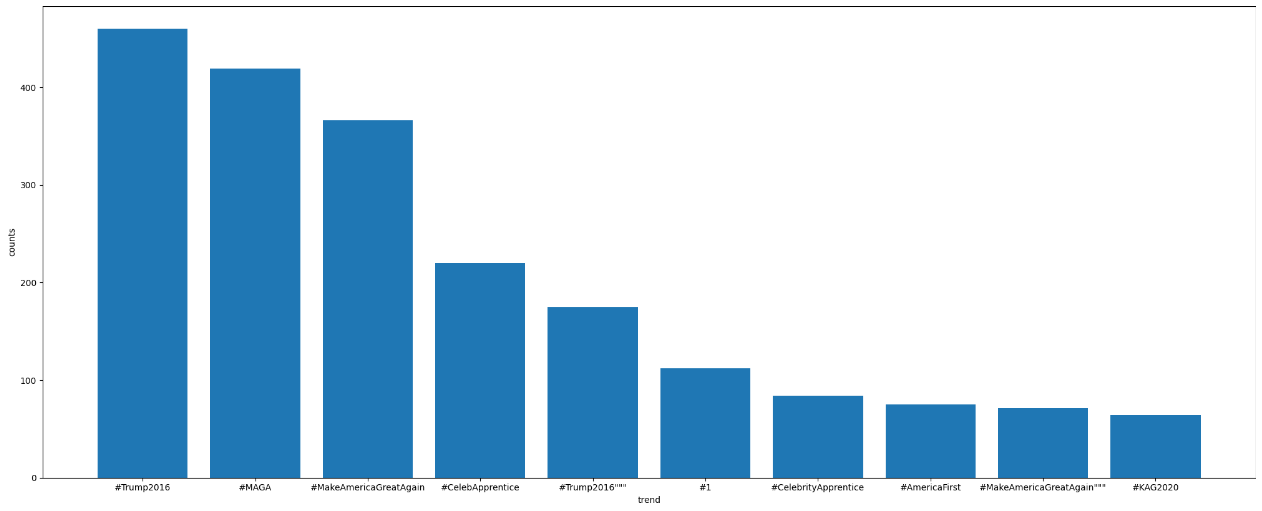
Part A: Word cloud

Amp, Thank, trump, democrat, etc



Part B: Top-10 trends chart

#Trump2016
#Maga
#MakeAmericaGreatAgain


counts
#Trump2016
#MAGA
#MakeAmericaGreatAgain
#CelebApprentice
#Trump2016"""
#1
#CelebrityApprentice
#AmericaFirst
#MakeAmericaGreatAgain"""
#KAG2020
trend