CE314/887 – NATURAL LANGUAGE ENGINEERING

ASSIGNMENT 1: PROBABILITIES, REGULAR EXPRESSIONS AND LANGUAGE MODELS

Submitted by

ISHWAR VENUGOPAL

(REGISTRATION NUMBER: 1906084)

MSc Artificial Intelligence

November 2019

Part 1: Tokenization, Part-of-Speech Tagging

Program file name: Part_1.py

Q1. Create a program that reads the text from the following website and identify all the types and tokens before and after lowercasing and lemmatization. Use NLTK functions to perform these tasks, from the url reader to the tokenizer and lemmatizer.

https://www.theguardian.com/music/2018/oct/19/while-my-guitar-gently-weeps-beatles-george-harrison

Solution:

Sample Output:

```
**********

*** Lowercasing ***

The number of tokens before lowercase: 768
The number of types before lowercase 400


The number of tokens after lowercase: 768
The number of types after lowercase 375


**********

*** Lemmatization ***

The number of tokens before lemmatization: 768
The number of types before lemmatization: 375
The number of tokens after lemmatization: 768
The number of types after lemmatization: 366


**********
```

Explanation:

I have used the BeautifulSoup package in python to extract the text from the specified url [5]. Even after reading the html file with the package, it was found that the text still had the following extra elements:

i) Certain HTML tags that were used during the creation of the webpage [3]
ii) A large number of blank lines
iii) Long Headings spread over multiple lines with unnecessary spaces in between.

All these unwanted elements were first removed from the extracted text [2]. For the tokenization and lemmatization of the text, I used some in-built functions in the NLTK package like word_tokenize() and WordNetLemmatizer() [4].

Q2 (a) Assign part-of-speech (POS) tags to all tokens in the text used above. Use one of the implemented POS taggers in NLTK to do this.

Solution:

Sample Output:

The text after POS tagging:

[('unheard', 'JJ'), ('version', 'NN'), ('of', 'IN'), ('the', 'DT'), ('beatles', 'NNS'), ('while', 'IN'), ('my', 'PRP$'), ('guitar', 'NN'), ('gently', 'RB'), ('weeps', 'VBZ'), ('released', 'VBN'), ('music', 'NN'), ('the', 'DT'), ('guardian', 'JJ'), ('skip', 'NN'), ('to', 'TO'), ('main', 'JJ'), ('content', 'NN'), ('the', 'DT'), ('guardian', 'JJ'), ('back', 'NN'), ('to', 'TO'), ('home', 'NN'), ('support', 'NN'), ('the', 'DT'), ('guardian', 'NN'), ('available', 'JJ'), ('for', 'IN'), ('everyone', 'NN'), ('funded', 'VBN'), ('by', 'IN'), ('reader', 'NN'), ('contribute', 'NN'), ('subscribe', 'NN'), ('subscribe', 'NN'), ('search', 'NN'), ('job', 'NN'), ('dating', 'VBG'), ('sign', 'NN'), ('in', 'IN'), ('my', 'PRP$'), ('account', 'NN'), ('comment', 'NN'), ('reply', 'VBZ'), ('public', 'JJ'), ('profile', 'NN'), ('account', 'NN'), ('detail', 'NN'), ('email', 'NN'), ('marketing', 'NN'), ('membership', 'NN'), ('contribution', 'NN'), ('subscription', 'NN'), ('sign', 'NN'), ('out', 'IN'), ('search', 'NN'), ('switch', 'NN'), ('to', 'TO'), ('the', 'DT'), ('uk', 'JJ'), ('edition', 'NN'), ('switch', 'NN'), ('to', 'TO'), ('the', 'DT'), ('u', 'JJ'), ('edition', 'NN'), ('switch', 'NN'), ('to', 'TO'), ('the', 'DT'), ('australia', 'JJ'), ('edition', 'NN'), ('switch', 'NN'), ('to', 'TO'), ('the', 'DT'), ('international', 'JJ'), ('edition', 'NN'), ('current', 'JJ'), ('edition', 'NN'), ('uk', 'JJ'), ('edition', 'NN'), ('news', 'NN'), ('opinion', 'NN'), ('sport', 'NN'), ('culture', 'NN'), ('lifestyle', 'VBZ'), ('show', 'VBP'), ('more', 'RBR'), ('news', 'NN'), ('uk', 'JJ'), ('news', 'NN'), ('election', 'NN'), ('2019', 'CD'), ('world', 'NN'), ('news', 'NN'), ('business', 'NN'), ('football', 'NN'), ('environment', 'NN'), ('uk', 'JJ'), ('politics', 'NNS'), ('education', 'NN'), ('society', 'NN'), ('science', 'NN'), ('tech', 'NN'), ('global', 'JJ'), ('development', 'NN'), ('city', 'NN'), ('obituary', 'JJ'), ('opinion', 'NN'), ('the', 'DT'), ('guardian', 'JJ'), ('view', 'NN'), ('columnist', 'NN'), ('cartoon', 'NN'), ('opinion', 'NN'), ('video', 'NN'), ('letter', 'NN'), ('sport', 'NN'), ('football', 'NN'), ('cricket', 'NN'), ('rugby', 'NN'), ('union', 'NN'), ('tennis', 'NN'), ('cycling', 'VBG'), ('f1', 'JJ'), ('golf', 'NN'), ('boxing', 'VBG'), ('rugby', 'JJ'), ('league', 'NN'), ('racing', 'VBG'), ('u', 'JJ'), ('sport', 'JJ'), ('culture', 'NN'), ('film', 'NN'), ('music', 'NN'), ('tv', 'NN'), ('radio', 'NN'), ('book', 'NN'), ('art', 'NN'), ('design', 'NN'), ('stage', 'NN'), ('game', 'NN'), ('classical', 'JJ'), ('lifestyle', 'JJ'), ('fashion', 'NN'), ('food', 'NN'), ('recipe', 'VBP'), ('travel', 'NN'), ('health', 'NN'), ('fitness', 'NN'), ('woman', 'NN'), ('men', 'NNS'), ('love', 'VBP'), ('sex', 'NN'), ('beauty', 'NN'), ('home', 'NN'), ('garden', 'NN'), ('money', 'NN'), ('car', 'NN'), ('what', 'WP'), ('term', 'NN'), ('do', 'VBP'), ('you', 'PRP'), ('want', 'VB'), ('to', 'TO'), ('search', 'VB'), ('search', 'NN'), ('with', 'IN'), ('google', 'NNS'), ('make', 'VBP'), ('a', 'DT'), ('contribution', 'NN'), ('subscribe', 'NN'), ('uk', 'JJ'), ('edition', 'NN'), ('switch', 'NN'), ('to', 'TO'), ('the', 'DT'), ('u', 'JJ'), ('edition', 'NN'), ('switch', 'NN'), ('to', 'TO'), ('the', 'DT'), ('australia', 'JJ'), ('edition', 'NN'), ('switch', 'NN'), ('to', 'TO'), ('the', 'DT'), ('international', 'JJ'), ('edition', 'NN'), ('search', 'NN'), ('job', 'NN'), ('dating', 'VBG'), ('holiday', 'JJ'), ('live', 'JJ'), ('event', 'NN'), ('masterclasses', 'NNS'), ('digital', 'JJ'), ('archive', 'JJ'), ('guardian', 'JJ'), ('print', 'NN'), ('shop', 'NN'), ('patron', 'NN'), ('discount', 'NN'), ('code', 'VBD'), ('the', 'DT'), ('guardian', 'JJ'), ('app', 'NN'), ('video', 'NN'), ('podcasts', 'VBZ'), ('picture', 'NN'), ('newsletter', 'NN'), ('today', 'NN'), ("'s", 'POS'), ('paper', 'NN'), ('inside', 'IN'), ('the', 'DT'), ('guardian', 'JJ'), ('the', 'DT'), ('observer', 'NN'), ('guardian', 'JJ'), ('weekly', 'JJ'), ('professional', 'NN'), ('network', 'NN'), ('crossword', 'NN'), ('facebook', 'NN'), ('twitter', 'NN'), ('search', 'NN'), ('job', 'NN'), ('dating', 'VBG'), ('holiday', 'JJ'), ('live', 'JJ'), ('event', 'NN'), ('masterclasses', 'NNS'), ('digital', 'JJ'), ('archive', 'JJ'), ('guardian', 'JJ'), ('print', 'NN'), ('shop', 'NN'), ('patron', 'NN'), ('discount', 'NN'), ('code', 'NN'), ('film', 'NN'), ('music', 'NN'), ('tv', 'NN'), ('radio', 'NN'), ('book', 'NN'), ('art', 'NN'), ('design', 'NN'), ('stage', 'NN'), ('game', 'NN'), ('classical', 'JJ'), ('more', 'JJR'), ('the', 'DT'), ('beatles', 'NNS'), ('this', 'DT'), ('article', 'NN'), ('is', 'VBZ'), ('more', 'JJR'), ('than', 'IN'), ('1', 'CD'), ('year', 'NN'), ('old', 'JJ'), ('unheard', 'JJ'), ('version', 'NN'), ('of', 'IN'), ('the', 'DT'), ('beatles', 'NNS'), ('while', 'IN'), ('my', 'PRP$'), ('guitar', 'NN'), ('gently', 'RB'), ('weeps', 'VBZ'), ('released', 'VBN'), ('this', 'DT'), ('article', 'NN'), ('is', 'VBZ'), ('more', 'JJR'), ('than', 'IN'), ('1', 'CD'), ('year', 'NN'), ('old', 'JJ'), ('acoustic', 'JJ'), ('demo', 'NN'), ('of', 'IN'), ('the', 'DT'), ('song', 'NN'), ('regarded', 'VBD'), ('a', 'DT'), ('one', 'CD'), ('of', 'IN'), ('george', 'NN'), ('harrison', 'NN'), ('s', 'NN'), ('best', 'JJS'), ('composition', 'NN'), ('to', 'TO'), ('be', 'VB'), ('included', 'VBN'), ('in', 'IN'), ('remastered', 'JJ'), ('white', 'JJ'), ('album', 'NN'), ('set', 'VBN'), ('ben', 'JJ'), ('beaumont-thomas', 'JJ'), ('ben_bt', 'NN'), ('fri', 'NN'), ('19', 'CD'), ('oct', 'NN'), ('2018', 'CD'), ('12.09', 'CD'), ('bst', 'NN'), ('last', 'JJ'), ('modified', 'VBN'), ('on', 'IN'), ('fri', 'NN'), ('19', 'CD'), ('oct', 'NN'), ('2018', 'CD'), ('17.30', 'CD'), ('bst', 'NN'), ('share', 'NN'), ('on', 'IN'), ('facebook', 'NN'), ('share', 'NN'), ('on', 'IN'), ('twitter', 'NN'), ('share', 'NN'), ('via', 'IN'), ('email', 'NN'), ('the', 'DT'), ('beatles', 'NNS'), ('in', 'IN'), ('1968', 'CD'), ('with',

'IN'), ('george', 'NN'), ('harrison', 'NN'), ('front', 'JJ'), ('photograph', 'NN'), ('rex', 'NN'), ('features/blackbrow', 'VBP'), ('three', 'CD'), ('unheard', 'JJ'), ('version', 'NN'), ('of', 'IN'), ('while', 'IN'), ('my', 'PRP$'), ('guitar', 'NN'), ('gently', 'RB'), ('weeps', 'VBZ'), ('regarded', 'VBN'), ('by', 'IN'), ('many', 'JJ'), ('a', 'DT'), ('george', 'NN'), ('harrison', 'NN'), ('s', 'JJ'), ('greatest', 'JJS'), ('contribution', 'NN'), ('to', 'TO'), ('the', 'DT'), ('beatles', 'NNS'), ('have', 'VBP'), ('been', 'VBN'), ('released', 'VBN'), ('online', 'IN'), ('the', 'DT'), ('song', 'NN'), ('wa', 'NN'), ('written', 'VBN'), ('by', 'IN'), ('harrison', 'NN'), ('in', 'IN'), ('1968', 'CD'), ('after', 'IN'), ('he', 'PRP'), ('had', 'VBD'), ('studied', 'VBN'), ('transcendental', 'JJ'), ('meditation', 'NN'), ('with', 'IN'), ('the', 'DT'), ('maharishi', 'NN'), ('mahesh', 'JJ'), ('yogi', 'NN'), ('in', 'IN'), ('india', 'NN'), ('there', 'EX'), ('are', 'VBP'), ('two', 'CD'), ('completely', 'RB'), ('unheard', 'JJ'), ('version', 'NN'), ('the', 'DT'), ('esher', 'JJR'), ('demo', 'NN'), ('is', 'VBZ'), ('an', 'DT'), ('acoustic', 'JJ'), ('version', 'NN'), ('complete', 'JJ'), ('with', 'IN'), ('beautiful', 'JJ'), ('multitracked', 'JJ'), ('vocal', 'NN'), ('recorded', 'VBD'), ('at', 'IN'), ('harrison', 'NN'), ('s', 'NN'), ('house', 'NN'), ('in', 'IN'), ('surrey', 'NN'), ('in', 'IN'), ('may', 'MD'), ('1968', 'CD'), ('in', 'IN'), ('preparation', 'NN'), ('for', 'IN'), ('the', 'DT'), ('studio', 'NN'), ('recording', 'VBG'), ('that', 'WDT'), ('began', 'VBD'), ('later', 'RB'), ('that', 'IN'), ('month', 'NN'), ('it', 'PRP'), ('is', 'VBZ'), ('one', 'CD'), ('of', 'IN'), ('27', 'CD'), ('demo', 'NN'), ('recorded', 'VBN'), ('at', 'IN'), ('the', 'DT'), ('house', 'NN'), ('included', 'VBD'), ('on', 'IN'), ('a', 'DT'), ('forthcoming', 'NN'), ('remastered', 'JJ'), ('version', 'NN'), ('of', 'IN'), ('the', 'DT'), ('white', 'JJ'), ('album', 'NN'), ('facebook', 'NN'), ('twitter', 'NN'), ('pinterest', 'VBP'), ('another', 'DT'), ('titled', 'VBN'), ('acoustic', 'JJ'), ('version', 'NN'), ('take', 'VB'), ('2', 'CD'), ('is', 'VBZ'), ('a', 'DT'), ('raw', 'JJ'), ('shaky', 'NN'), ('but', 'CC'), ('endearing', 'VBG'), ('alternative', 'JJ'), ('to', 'TO'), ('the', 'DT'), ('solo', 'NN'), ('harrison', 'NN'), ('demo', 'NN'), ('played', 'VBD'), ('on', 'IN'), ('acoustic', 'JJ'), ('guitar', 'NN'), ('and', 'CC'), ('harmonium', 'NN'), ('that', 'WDT'), ('wa', 'VBZ'), ('previously', 'RB'), ('released', 'VBN'), ('a', 'DT'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('anthology', 'NN'), ('3', 'CD'), ('compilation', 'NN'), ('in', 'IN'), ('1996', 'CD'), ('at', 'IN'), ('one', 'CD'), ('point', 'NN'), ('harrison', 'NN'), ('complains', 'VBZ'), ('yeah', 'CC'), ('maybe', 'RB'), ('you', 'PRP'), ('ll', 'VBP'), ('have', 'VBP'), ('to', 'TO'), ('give', 'VB'), ('him', 'PRP'), ('his', 'PRP$'), ('own', 'JJ'), ('mic', 'NN'), ('to', 'TO'), ('someone', 'NN'), ('in', 'IN'), ('the', 'DT'), ('studio', 'NN'), ('a', 'DT'), ('the', 'DT'), ('harmonium', 'NN'), ('drift', 'NN'), ('into', 'IN'), ('discordance', 'NN'), ('facebook', 'NN'), ('twitter', 'NN'), ('pinterest', 'VBP'), ('the', 'DT'), ('final', 'JJ'), ('unheard', 'JJ'), ('version', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('newly', 'RB'), ('remastered', 'VBN'), ('mix', 'NN'), ('of', 'IN'), ('the', 'DT'), ('album', 'JJ'), ('track', 'NN'), ('like', 'IN'), ('the', 'DT'), ('original', 'JJ'), ('it', 'PRP'), ('feature', 'VBD'), ('an', 'DT'), ('electric', 'JJ'), ('guitar', 'NN'), ('part', 'NN'), ('from', 'IN'), ('eric', 'JJ'), ('clapton', 'NN'), ('he', 'PRP'), ('once', 'RB'), ('said', 'VBD'), ('he', 'PRP'), ('felt', 'VBD'), ('the', 'DT'), ('song', 'NN'), ('wa', 'NN'), ('about', 'IN'), ('harrison', 'NN'), ('s', 'JJ'), ('dislocation', 'NN'), ('from', 'IN'), ('the', 'DT'), ('group', 'NN'), ('who', 'WP'), ('didn', 'VBZ'), ('t', 'NN'), ('share', 'NN'), ('harrison', 'JJ'), ('s', 'NN'), ('zeal', 'NN'), ('for', 'IN'), ('eastern', 'JJ'), ('mysticism', 'NN'), ('following', 'VBG'), ('their', 'PRP$'), ('trip', 'NN'), ('to', 'TO'), ('india', 'VB'), ('ringo', 'NN'), ('starr', 'NN'), ('infamously', 'RB'), ('brought', 'VBD'), ('a', 'DT'), ('suitcase', 'NN'), ('full', 'JJ'), ('of', 'IN'), ('heinz', 'JJ'), ('bean', 'NN'), ('on', 'IN'), ('the', 'DT'), ('trip', 'NN'), ('because', 'IN'), ('he', 'PRP'), ('didn', 'VBZ'), ('t', 'NN'), ('want', 'NN'), ('to', 'TO'), ('eat', 'VB'), ('spicy', 'NN'), ('food', 'NN'), ('and', 'CC'), ('left', 'VBD'), ('after', 'IN'), ('two', 'CD'), ('week', 'NN'), ('the', 'DT'), ('newly', 'RB'), ('remastered', 'VBN'), ('white', 'JJ'), ('album', 'NN'), ('is', 'VBZ'), ('released', 'VBN'), ('9', 'CD'), ('november', 'NNS'), ('and', 'CC'), ('a', 'DT'), ('well', 'RB'), ('a', 'DT'), ('the', 'DT'), ('esher', 'NN'), ('demo', 'NN'), ('also', 'RB'), ('feature', 'VBD'), ('50', 'CD'), ('mostly', 'RB'), ('unreleased', 'JJ'), ('additional', 'JJ'), ('recording', 'NN'), ('made', 'VBD'), ('during', 'IN'), ('the', 'DT'), ('studio', 'NN'), ('session', 'NN'), ('topic', 'IN'), ('the', 'DT'), ('beatles', 'NNS'), ('george', 'VBP'), ('harrison', 'JJ'), ('pop', 'NN'), ('and', 'CC'), ('rock', 'NN'), ('news', 'NN'), ('share', 'NN'), ('on', 'IN'), ('facebook', 'NN'), ('share', 'NN'), ('on', 'IN'), ('twitter', 'NN'), ('share', 'NN'), ('via', 'IN'), ('email', 'NN'), ('share', 'NN'), ('on', 'IN'), ('linkedin', 'NNS'), ('share', 'NN'), ('on', 'IN'), ('pinterest', 'JJS'), ('share', 'NN'), ('on', 'IN'), ('whatsapp', 'NNS'), ('share', 'NN'), ('on', 'IN'), ('messenger', 'NN'), ('reuse', 'NN'), ('this', 'DT'), ('content', 'JJ'), ('view', 'NN'), ('all', 'DT'), ('comment', 'NN'), ('order', 'NN'), ('by', 'IN'), ('newest', 'JJS'), ('oldest', 'JJS'), ('recommendation', 'NN'), ('show', 'NN'), ('25', 'CD'), ('25', 'CD'), ('50', 'CD'), ('100', 'CD'), ('all', 'DT'), ('thread', 'NN'), ('collapsed', 'VBD'), ('expanded', 'VBN'), ('unthreaded', 'JJ'), ('loading', 'VBG'), ('comments…', 'NN'), ('trouble', 'NN'), ('loading', 'VBG'), ('view', 'NN'), ('more', 'RBR'), ('comment', 'NN'), ('most', 'RBS'), ('popular', 'JJ'), ('film', 'NN'), ('music', 'NN'), ('tv', 'NN'), ('radio', 'NN'), ('book', 'NN'), ('art', 'NN'), ('design', 'NN'), ('stage', 'NN'), ('game', 'NN'), ('classical', 'JJ'), ('news', 'NN'), ('opinion', 'NN'), ('sport', 'NN'), ('culture', 'NN'), ('lifestyle', 'VBZ'), ('about', 'IN'), ('u', 'JJ'), ('contact', 'NN'), ('u', 'JJ'), ('complaint', 'NN'), ('correction', 'NN'), ('securedrop', 'NN'), ('work', 'NN'), ('for', 'IN'), ('u', 'JJ'), ('privacy', 'NN'), ('policy', 'NN'), ('cookie', 'NN'), ('policy', 'NN'), ('term', 'NN'), ('condition', 'NN'), ('help', 'VBP'), ('all', 'DT'), ('topic', 'NN'), ('all', 'DT'), ('writer', 'NN'), ('modern', 'JJ'), ('slavery', 'NN'), ('act', 'NN'), ('digital', 'JJ'), ('newspaper', 'NN'), ('archive', 'JJ'), ('facebook', 'NN'), ('twitter', 'NN'), ('advertise', 'NN'), ('with', 'IN'), ('u', 'JJ'), ('guardian', 'JJ'), ('lab', 'NN'), ('search', 'NN'), ('job', 'NN'), ('dating', 'VBG'), ('patron', 'NN'), ('discount', 'NN'), ('code', 'NN'), ('support', 'NN'), ('the', 'DT'), ('guardian', 'NN'), ('available', 'JJ'), ('for', 'IN'), ('everyone', 'NN'), ('funded', 'VBN'),

('by', 'IN'), ('reader', 'NN'), ('contribute', 'NN'), ('subscribe', 'NN'), ('back', 'RB'), ('to', 'TO'), ('top', 'VB'), ('2019', 'CD'), ('guardian', 'JJ'), ('news', 'NN'), ('medium', 'NN'), ('limited', 'VBD'), ('or', 'CC'), ('it', 'PRP'), ('affiliated', 'VBD'), ('company', 'NN'), ('all', 'RB'), ('right', 'RB'), ('reserved', 'VBN'), ('close', 'RB')]

In order to do the Part-of-Speech tagging, I used the pos_tag() function which is in-built in the NLTK package [1].

Q2 (b) POS taggers do not always assign correct tags to words. Identify tagging errors in the sentences above and briefly explain why these errors may have been caused.

Solution:

Sample Output:

```
*** UNDERSTANDING TAGGING ERRORS ***

POS tags from the default pos_tag() method:
  Squeezed text (52 lines).
POS tags from the Bigram tagger in nltk:
  Squeezed text (51 lines).
DATA OBTAINED FROM THE CONFUSION MATRIX (i.e the tagging errors resulting from the two techniques):

Total Number of True Positives: 490


False Negatives: 278


False Positives: 278
```

Explanation:

In order to identify the tagging errors associated with the pos_tag() technique used, I chose to compare it with the results obtained from another tagging technique under NLTK. The other tagging technique that I used here was the Bigram Tagger combined with a Unigram tagger [7,8,11]. The tags obtained from both the techniques were analysed and a Confusion matrix[9] was created. This would facilitate us in identifying words that have been wrongly or ambiguously tagged by either of the tagging techniques. The number of true positives, false positives and false negatives are displayed on the screen after successfully running the program.

Here I am using the tags obtained from the Bigram tagger as the reference (true) tags. True positives refer to those tags which are correctly tagged by the pos_tag(). False positives are those words which are tagged as positive to one class, but which doesn't belong to that class in reality. False negatives, on the other hand, are those classified as negative in class but actually did belong to that class. Hence, the confusion matrix is a good way to understand how well our POS tagger has performed and where have the mistakes been made.

Errors in POS tagging are mainly caused due to the difference in the way each tagger deals with unknown words. The extent to which the tagger has been trained upon determines the number of words it encounters and classifies as 'UNK'. The number of errors in tagging is determined mainly by this reason.

Part 2: Regular Expressions, FSAs and FSTs

Program file: Part_2.py

Q3: Write a regular expression that can find all telephone numbers in a text. Your expression should be able to deal with different formats, for example +55 51 33083838, 1206 872020, 01206 872020 and 05679401945 as well as +44 5679401945 and 0044 5679401945. For full marks: include the output of a Python program that applies your regular expression to any url specified by the user, reads it and finds the telephone numbers. The output should clearly identify what the telephone number is.

Solution:

Sample Output:

```
Enter a valid url: Demonstrating invalid url
Enter a valid url: https://www.essex.ac.uk/about/contact

Warning (from warnings module):
  File "C:\Users\USER\Downloads\NLE\NLE\Final\Part_2\Part_
    soup = BeautifulSoup(html)
UserWarning: No parser was explicitly specified, so I'm usi
her system, or in a different virtual environment, it may u

The code that caused this warning is on line 32 of the file
.parser"' to the BeautifulSoup constructor.

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\USER\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\USER\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
List of telephone numbers:
+44 0 1206 873333
+44 0 1206 873666
+44 0 1206 872719
+44 0 1206 872400
+44 0 1206 874321
+44 0 1702 328200
+44 0 1206 873333
```

Explanation:
The program first checks if the url provided by the user is a valid url or not[12]. To demonstrate this, the first url entered in the sample output shown is intentionally not a valid url. The regular expression used in this program to search for telephone numbers can identify telephone numbers of the following formats:

1) +55 51 33083838
2) 1206 872020
3) 01206 872020
4) 05679401945
5) +44 5679401945
6) 0044 5679401945

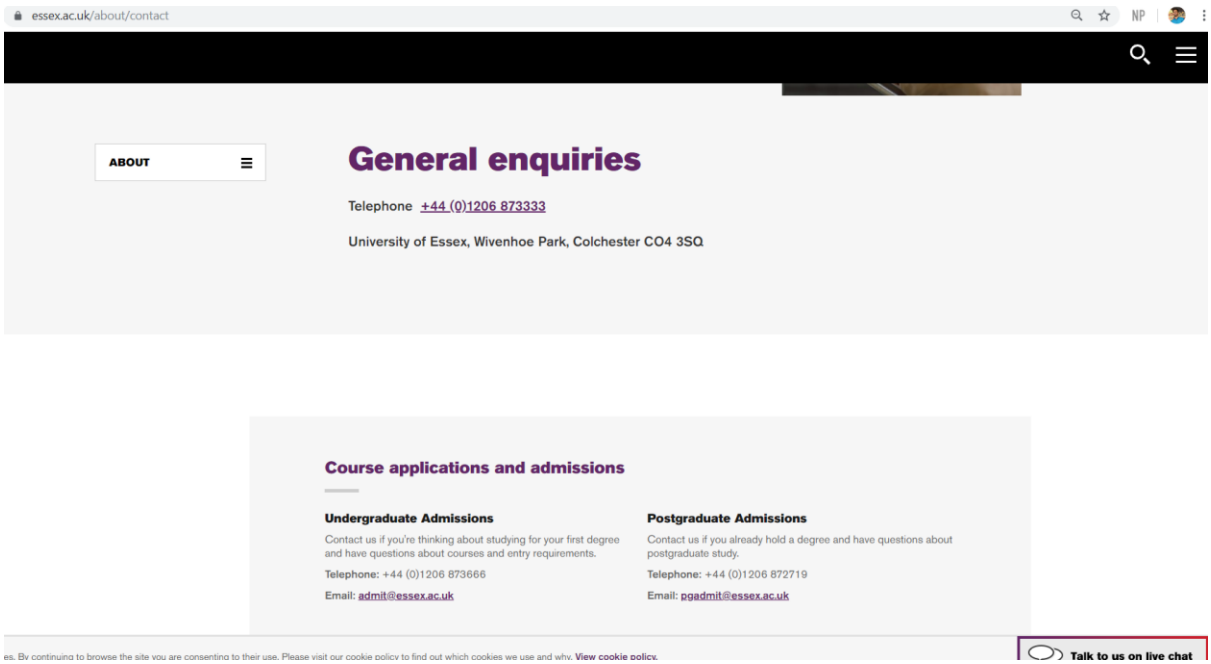
7) +44 0 1206 873333

This still does not cover all the possible formats of telephone numbers because each country has multiple different formats to depict landline and mobile phone numbers[16]. Hence, it gives rise to a numerous amount of possibilities.

[+]?0?0?[0-9]{2}[\s]?[0-9]{1,3}[\s]?[0-9]{4}[\s]?[0-9]{3,6}

This Regular Expression above (used in the program) basically looks for a '+','0','00' which usually appear in-front of telephone numbers. And checks for spaces between the numbers according to the 7 formats listed above[15].

The webpage from which the telephone numbers are extracted (for the sample output shown above) actually looks like this:



It is a webpage showing the contact details for the University of Essex. The program correctly extracts all the telephone numbers on the webpage.

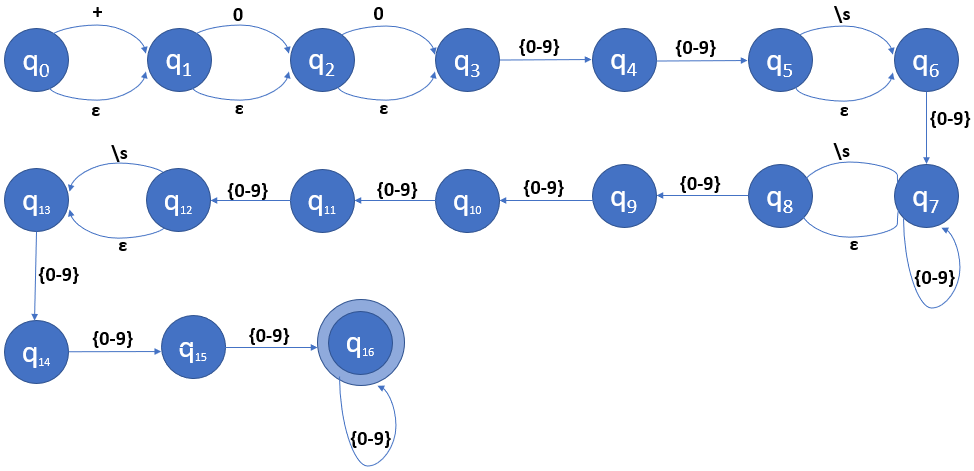
Q4. Write a FSA equivalent to the regular expression you just wrote. You can either use a drawing program or write down a transition table.

Solution:

The regular expression used in the program is:

[+]?0?0?[0-9]{2}[\s]?[0-9]{1,3}[\s]?[0-9]{4}[\s]?[0-9]{3,6}

The FSA equivalent for this Regular Expression looks like the following:



Part 3: N-Gram Models

Q5. Computing a unigram model

Use the Toy dataset. The vocabulary is the words in the sampledata.vocab.txt plus the UNK token. Do not include <s> and </s> in the vocabulary.

a) Compute the probabilities in a unigram language model without smoothing. Show your work for P(a), P(c), P(UNK). Create a table and list all the probabilities in the unigram model.

Solution:

Sample Output:

```
- Unsmoothed -

UNK 0.0
<s> 0.14285714285714285
a 0.19047619047619047
b 0.23809523809523808
c 0.2857142857142857
</s> 0.14285714285714285
```

Explanation:
The steps followed were as follows [6]:

1) Count the total number of tokens in the training data (N)
2) Get the total vocabulary for the unigram model (V)
3) Calculate the frequency of each unigram
4) Computer the corresponding probabilities of each unigram using the following equation:

$$P(w_i) = \frac{count(w_i)}{N}$$

b) Smooth the model using Laplace smoothing. Show your work for P(a), P(c), P(UNK). Show all the smoothed probabilities in a table.

Solution:

Sample Output:

```
- Smoothed -

UNK 0.04
<s> 0.16
a 0.2
b 0.24
c 0.28
</s> 0.16
```

Explanation:
The probabilities obtained in the previous step were smoothed using the following equation for Laplace smoothing:

$$P(w_i) = \frac{count(w_i) + 1}{N + V}$$

Q6. Computing a bigram model

Use the Toy dataset. The vocabulary is the words in sampledata.vocab.txt, plus the UNK token and </s> symbols. <s> should be included only in the context or history. </s> should not be included in the history but only as the following word. The table below should clarify for you.

a) Compute the probabilities in a bigram language model without smoothing. Show your work for P(b|a), P(UNK|<s>), P(UNK|UNK). Create a table and list all of the probabilities in the model.

Solution:

Sample Output:

```
- Unsmoothed -

       UNK       <s>     a    b         c
UNK    0.0  0.000000  0.00  0.0  0.000000
</s>   0.0  0.000000  0.00  0.2  0.333333
a      0.0  0.666667  0.25  0.0  0.166667
b      0.0  0.333333  0.50  0.2  0.166667
c      0.0  0.000000  0.25  0.6  0.333333
```

Explanation:
The steps followed were as follows:

1) Calculating the total vocabulary for the bigram model
2) Counting the frequency of each bigram in the training data

3) Using the frequencies to find the corresponding probabilities with the following formula:

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1},w_i)}{count(w_{i-1})}$$

To display the output as a table, I used the pandas library in python [13,14]. The dictionary used to save the probabilities was converted into a dataframe.

b) Smooth the model using Laplace smoothing. Show your work for P(b|a), P(UNK|<s>), P(UNK|UNK). Show all the smoothed probabilities in a table.

Solution:

Sample Output:

```
- Smoothed -

            UNK       <s>    a         b         c
UNK    0.166667  0.111111  0.1  0.090909  0.083333
</s>   0.166667  0.111111  0.1  0.181818  0.250000
a      0.166667  0.333333  0.2  0.090909  0.166667
b      0.166667  0.222222  0.3  0.181818  0.166667
c      0.166667  0.111111  0.2  0.363636  0.250000
```

Explanation:
The probabilities calculated earlier was smoothed using the Laplace smoothing. The following equation was applied:

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1},w_i) + 1}{count(w_{i-1}) + V}$$

Q7 Computing sentence probabilities

Use the Toy dataset. There are 5 sentences in sampletest.txt. Using the smoothed models above, compute the probability of each sentence. For unigram probability, you should ignore the <s> and </s> symbols.
a) Show your work for sentence numbers 3, 4, 5, for each model: unigram and bigram.
b) Fill in the probabilities of all the sentences in a table.

Solution:

Sample Output:

```
=== SENTENCE PROBABILITIES ===

                        Sent 1    Sent 2    Sent 3    Sent 4    Sent 5
Unigram probability:  0.000344  0.000023  0.000344  0.000014  0.000002
Bigram Probability:   0.009091  0.000413  0.000168  0.000505  0.000168
```

Explanation:
The table shown above gives the sentence probabilities for all the five sentences in the testing data, with the unigram model and the bigram model. We can observe that the probabilities seem to slightly increase when computed with the bigram model.

REFERENCES

[1] Gianpaul Rachiele, 2018, *Tokenisation and Parts of Speech (POS) Tagging in Python's NLTK Library*, Medium,<https://medium.com/@gianpaul.r/tokenization-and-parts-of-speech-pos-tagging-in-pythons-nltk-library-2d30f70af13b>

[2] 2008, *Extracting text from HTML file using Python,* StackOverflow, <https://stackoverflow.com/questions/328356/extracting-text-from-html-file-using-python>

[3] *HTML Head*, w3schools.com,<https://www.w3schools.com/html/html_head.asp>

[4] *Python: Lemmatization with NLTK,*GeeksforGeeks, <https://www.geeksforgeeks.org/python-lemmatization-with-nltk/>

[5] *Processing Raw Text,* NLTK.org, <https://www.nltk.org/book/ch03.html>

[6] *Language Processing and Python*, NLTK.org, <https://www.nltk.org/book/ch01.html#chap-introduction>

[7] *Categorizing and tagging words,* NLTK.org, <https://www.nltk.org/book/ch05.html>

[8] Pastebin, <https://pastebin.com/EC8fFqLU>

[9] 2014, *How to use the confusion matrix module in NLTK,* StackOverflow, <https://stackoverflow.com/questions/23704361/how-to-use-the-confusion-matrix-module-in-nltk>

[10] *nltk.tag package,* NLTK 3.4.5 documentation, <http://www.nltk.org/api/nltk.tag.html?highlight=pos_tag>

[11] 2016, *NLTK BigramTagger does not tag half of the sentence,* StackOverflow, <https://stackoverflow.com/questions/39167671/nltk-bigramtagger-does-not-tag-half-of-the-sentence>

[12] 2014, *Validating URLs in Python,* StackOverflow, <https://stackoverflow.com/questions/22238090/validating-urls-in-python?lq=1>

[13] 2012, *Printing Lists as Tabular Data,* StackOverflow, <https://stackoverflow.com/questions/9535954/printing-lists-as-tabular-data>

[14] 2013, *Building a table from Python nested dictionaries with missing values,* StackOverflow, <https://stackoverflow.com/questions/18746278/building-a-table-from-python-nested-dictionaries-with-missing-values>

[15] A.M Kuchling, *Regular Expression HOWTO,* Python.org, <https://docs.python.org/2/howto/regex.html>

[16] 2008, *National Conventions for writing telephone numbers*, Wikipedia, <https://en.wikipedia.org/wiki/National_conventions_for_writing_telephone_numbers>