Antibiotics Report (Work in progress)

Natural Language Processing / Text Analysis Project

This jupyter notebook / github project repository details the process of text analytics. In particular, it shows the process of extracting and cleaning text from a PDF. It also shows how to pass text through an NLP pipeline while using graphs to present the results.

Install: conda install -c conda-forge pypdf2 conda install -c conda-forge spacy conda install -c anaconda numpy conda install -c anaconda pandas conda install -c conda-forge matplotlib conda install seaborn conda install -c conda-forge geopandas conda install -c conda-forge descartes

In [1]:

```
import PyPDF2
import spacy
import numpy
import pandas
import en_core_web_sm
```

In [2]:

```
reader = PyPDF2.PdfFileReader("2019-ar-threats-report-508.pdf")
full_text = ""
```

In [3]:

```
pdf_page_number = 7
```

In [4]:

```
full_text = full_text.replace("  ", " ").replace(
    "  ", " ").replace("  ", " ")

with open("transcript_clean.txt", "w", encoding="utf-8") as temp_file:
    temp_file.write(full_text)
```

In [5]:

```
corpus = open("transcript_clean.txt", "r", encoding="utf-8").read()
nlp = spacy.load("en_core_web_sm")
```

In [6]:

```
nlp.max_length = len(corpus)
doc = nlp(corpus)
```

In [7]:

```
data_list = [["text", "text_lower", "lemma", "lemma_lower",
                "part_of_speech", "is_alphabet", "is_stopword"]]

for token in doc:
    data_list.append([token.text, token.lower_, token.lemma_, token.lemma_.lower(), tok
en.pos_, token.is_alpha, token.is_stop])
```

In [8]:

```
import csv
from textblob_de import TextBlobDE as TextBlob
```

In [9]:

```
csv.writer(open("./tokens.csv", "w", encoding="utf-8",
                newline="")).writerows(data_list)
```

In [10]:

```
data_list = [["text", "text_lower", "label"]]

for ent in doc.ents:
    data_list.append([ent.text, ent.lower_, ent.label_])

csv.writer(open("./entities.csv", "w", encoding="utf-8",
                newline="")).writerows(data_list)
```

Negative Words in English TXT Source https://gist.github.com/mkulakowski2/4289441 (https://gist.github.com/mkulakowski2/4289441)

In [11]:

```
with open("positivewords.txt", "r", encoding="utf-8") as temp_file:
    positivewords = temp_file.read().splitlines()

with open("negativewords.txt", "r", encoding="utf-8") as temp_file:
    negativewords = temp_file.read().splitlines()
```

In [12]:

```
data_list = [["text", "score"]]

for sent in doc.sents:

    # Only take into account real sentences.
    if len(sent.text) > 10:

        score = 0

        # Start scoring the sentence.
        for word in sent:

            if word.lower_ in positive_words:
                score += 1

            if word.lower_ in negative_words:
                score -= 1

        data_list.append([sent.text, score])

csv.writer(open("./sentences.csv", "w", encoding="utf-8",
                newline="")).writerows(data_list)
```

In [13]:

```
import seaborn as sns
import numpy as np
import pandas as pd
```

In [14]:

```
sns.set(style="ticks",
    rc={
        "figure.figsize": [12, 7],
        "text.color": "white",
        "axes.labelcolor": "white",
        "axes.edgecolor": "white",
        "xtick.color": "white",
        "ytick.color": "white",
        "axes.facecolor": "#5C0E10",
        "figure.facecolor": "#5C0E10"}
    )
```

In [15]:

```
"""
Visit this website for additional details regarding Spacy: https://spacy.io/usage/spacy
-101

This script extracts features from the transcript txt file and saves them to .csv files
so they can be used in any toolkkit.
"""

import csv
import spacy


def main():
    """Loads the model and processes it.

    The model used can be installed by running this command on your CMD/Terminal:
    python -m spacy download es_core_news_md

    """

    corpus = open("transcript_clean.txt", "r", encoding="utf-8").read()
    nlp = spacy.load("en_core_web_sm")

    # Our corpus is bigger than the default limit, we will set
    # a new limit equal to its length.
    nlp.max_length = len(corpus)

    doc = nlp(corpus)

    get_tokens(doc)
    get_entities(doc)
    get_sentences(doc)


def get_tokens(doc):
    """Get the tokens and save them to .csv
    Parameters
    ----------
    doc : spacy.doc
        A doc object.
    """

    data_list = [["text", "text_lower", "lemma", "lemma_lower",
                  "part_of_speech", "is_alphabet", "is_stopword"]]

    for token in doc:
        data_list.append([
            token.text, token.lower_, token.lemma_, token.lemma_.lower(),
            token.pos_, token.is_alpha, token.is_stop
        ])

    with open("./tokens.csv", "w", encoding="utf-8", newline="") as tokens_file:
        csv.writer(tokens_file).writerows(data_list)


def get_entities(doc):
    """After getting the entitites they are saved to a .csv
    using the codes below...
    Code Parameters
    ----------
    doc : spacy.doc
        A doc object.
    """

    data_list = [["text", "text_lower", "label"]]

    for ent in doc.ents:
        data_list.append([ent.text, ent.lower_, ent.label_])

    with open("./entities.csv", "w", encoding="utf-8", newline="") as entities_file:
        csv.writer(entities_file).writerows(data_list)


def get_sentences(doc):
    """Get the sentences, score and save them to .csv
    You will require to download the dataset (zip) from the following url:

    You will need to download the txt for both positive and negative
    words in English, save to your computer, and upload to Jupyter Notebook
    or whatever interface you are coding in Python with.
    negativewords.txt
    positivewords.txt
    Parameters
    ----------
    doc : spacy.doc
        A doc object.
    """

    # Load positive and negative words into lists.
    with open("positivewords.txt", "r", encoding="utf-8") as temp_file:
        positivewords = temp_file.read().splitlines()

    with open("negativewords.txt", "r", encoding="utf-8") as temp_file:
        negativewords = temp_file.read().splitlines()

    data_list = [["text", "score"]]

    for sent in doc.sents:

        # Only take into account real sentences.
        if len(sent.text) > 10:

            score = 0

            # Start scoring the sentence.
            for word in sent:

                if word.lower_ in positive_words:
                    score += 1

                if word.lower_ in negative_words:
                    score -= 1

            data_list.append([sent.text, score])


    with open("./sentences.csv", "w", encoding="utf-8", newline="") as sentences_file:
        csv.writer(sentences_file).writerows(data_list)
```

```python
import sys

if __name__ == "__main__":

    main()
```

In [16]:

```python
df = pd.read_csv("./tokens.csv")
```

In [17]:

```python
df.loc[df["lemma_lower"] == "programa", "lemma_lower"] = "programar"
```

In [18]:

```python
words = df[(df["is_alphabet"] == True) & (df["is_stopword"] == False) & (
    df["lemma_lower"].str.len() > 1)]["lemma_lower"].value_counts()[:20]
```

In [19]:

```python
import PyPDF2
import re
pdfFileObj = open('2019-ar-threats-report-508.pdf', 'rb')
pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
pdfReader.numPages
```

Out[19]:

```
150
```

In [20]:

```python
pageObj = pdfReader.getPage(0)
pageObj.extractText()
```

Out[20]:

```
'ANTIBIOTIC RESISTANCE THREATS\n IN THE UNITED STATES\n2019\n Revised Dec.
2019\n'
```

In [21]:

```python
object = PyPDF2.PdfFileReader("2019-ar-threats-report-508.pdf")
```

In [22]:

```python
# Page number code - Python
NumPages = object.getNumPages()
```

In [23]:

```python
String = "antibiotic"
```

In [24]:

```
for i in range(0, NumPages):
    PageObj = object.getPage(i)
    print("this is page " + str(i))
    Text = PageObj.extractText()
    # print(Text)
    ResSearch = re.search(String, Text)
    print(ResSearch)
```

```
this is page 0
None
this is page 1
<re.Match object; span=(86, 96), match='antibiotic'>
this is page 2
None
this is page 3
None
this is page 4
<re.Match object; span=(367, 377), match='antibiotic'>
this is page 5
<re.Match object; span=(10, 20), match='antibiotic'>
this is page 6
<re.Match object; span=(210, 220), match='antibiotic'>
this is page 7
<re.Match object; span=(525, 535), match='antibiotic'>
this is page 8
<re.Match object; span=(20, 30), match='antibiotic'>
this is page 9
<re.Match object; span=(112, 122), match='antibiotic'>
this is page 10
None
this is page 11
<re.Match object; span=(136, 146), match='antibiotic'>
this is page 12
<re.Match object; span=(496, 506), match='antibiotic'>
this is page 13
None
this is page 14
<re.Match object; span=(38, 48), match='antibiotic'>
this is page 15
<re.Match object; span=(145, 155), match='antibiotic'>
this is page 16
<re.Match object; span=(604, 614), match='antibiotic'>
this is page 17
<re.Match object; span=(222, 232), match='antibiotic'>
this is page 18
<re.Match object; span=(218, 228), match='antibiotic'>
this is page 19
None
this is page 20
<re.Match object; span=(180, 190), match='antibiotic'>
this is page 21
None
this is page 22
<re.Match object; span=(172, 182), match='antibiotic'>
this is page 23
<re.Match object; span=(555, 565), match='antibiotic'>
this is page 24
<re.Match object; span=(68, 78), match='antibiotic'>
this is page 25
None
this is page 26
<re.Match object; span=(2959, 2969), match='antibiotic'>
this is page 27
<re.Match object; span=(94, 104), match='antibiotic'>
this is page 28
<re.Match object; span=(541, 551), match='antibiotic'>
this is page 29
None
this is page 30
```

```
<re.Match object; span=(67, 77), match='antibiotic'>
this is page 31
None
this is page 32
<re.Match object; span=(59, 69), match='antibiotic'>
this is page 33
<re.Match object; span=(87, 97), match='antibiotic'>
this is page 34
None
this is page 35
<re.Match object; span=(396, 406), match='antibiotic'>
this is page 36
<re.Match object; span=(87, 97), match='antibiotic'>
this is page 37
<re.Match object; span=(123, 133), match='antibiotic'>
this is page 38
<re.Match object; span=(200, 210), match='antibiotic'>
this is page 39
<re.Match object; span=(83, 93), match='antibiotic'>
this is page 40
<re.Match object; span=(122, 132), match='antibiotic'>
this is page 41
<re.Match object; span=(153, 163), match='antibiotic'>
this is page 42
None
this is page 43
<re.Match object; span=(214, 224), match='antibiotic'>
this is page 44
<re.Match object; span=(239, 249), match='antibiotic'>
this is page 45
<re.Match object; span=(252, 262), match='antibiotic'>
this is page 46
<re.Match object; span=(120, 130), match='antibiotic'>
this is page 47
<re.Match object; span=(85, 95), match='antibiotic'>
this is page 48
None
this is page 49
<re.Match object; span=(1954, 1964), match='antibiotic'>
this is page 50
<re.Match object; span=(143, 153), match='antibiotic'>
this is page 51
<re.Match object; span=(359, 369), match='antibiotic'>
this is page 52
<re.Match object; span=(129, 139), match='antibiotic'>
this is page 53
<re.Match object; span=(879, 889), match='antibiotic'>
this is page 54
<re.Match object; span=(129, 139), match='antibiotic'>
this is page 55
<re.Match object; span=(114, 124), match='antibiotic'>
this is page 56
<re.Match object; span=(487, 497), match='antibiotic'>
this is page 57
<re.Match object; span=(96, 106), match='antibiotic'>
this is page 58
<re.Match object; span=(707, 717), match='antibiotic'>
this is page 59
None
this is page 60
None
```

```
this is page 61
<re.Match object; span=(155, 165), match='antibiotic'>
this is page 62
<re.Match object; span=(118, 128), match='antibiotic'>
this is page 63
<re.Match object; span=(144, 154), match='antibiotic'>
this is page 64
<re.Match object; span=(148, 158), match='antibiotic'>
this is page 65
<re.Match object; span=(148, 158), match='antibiotic'>
this is page 66
None
this is page 67
<re.Match object; span=(244, 254), match='antibiotic'>
this is page 68
<re.Match object; span=(179, 189), match='antibiotic'>
this is page 69
<re.Match object; span=(103, 113), match='antibiotic'>
this is page 70
<re.Match object; span=(137, 147), match='antibiotic'>
this is page 71
<re.Match object; span=(156, 166), match='antibiotic'>
this is page 72
None
this is page 73
None
this is page 74
None
this is page 75
None
this is page 76
<re.Match object; span=(780, 790), match='antibiotic'>
this is page 77
<re.Match object; span=(424, 434), match='antibiotic'>
this is page 78
None
this is page 79
None
this is page 80
<re.Match object; span=(330, 340), match='antibiotic'>
this is page 81
<re.Match object; span=(137, 147), match='antibiotic'>
this is page 82
<re.Match object; span=(373, 383), match='antibiotic'>
this is page 83
<re.Match object; span=(1202, 1212), match='antibiotic'>
this is page 84
<re.Match object; span=(511, 521), match='antibiotic'>
this is page 85
<re.Match object; span=(456, 466), match='antibiotic'>
this is page 86
None
this is page 87
None
this is page 88
<re.Match object; span=(601, 611), match='antibiotic'>
this is page 89
<re.Match object; span=(676, 686), match='antibiotic'>
this is page 90
None
this is page 91
```

```
None
this is page 92
<re.Match object; span=(525, 535), match='antibiotic'>
this is page 93
<re.Match object; span=(790, 800), match='antibiotic'>
this is page 94
<re.Match object; span=(842, 852), match='antibiotic'>
this is page 95
<re.Match object; span=(748, 758), match='antibiotic'>
this is page 96
<re.Match object; span=(699, 709), match='antibiotic'>
this is page 97
<re.Match object; span=(1192, 1202), match='antibiotic'>
this is page 98
None
this is page 99
<re.Match object; span=(1410, 1420), match='antibiotic'>
this is page 100
<re.Match object; span=(599, 609), match='antibiotic'>
this is page 101
<re.Match object; span=(178, 188), match='antibiotic'>
this is page 102
None
this is page 103
<re.Match object; span=(1238, 1248), match='antibiotic'>
this is page 104
<re.Match object; span=(444, 454), match='antibiotic'>
this is page 105
None
this is page 106
<re.Match object; span=(631, 641), match='antibiotic'>
this is page 107
<re.Match object; span=(716, 726), match='antibiotic'>
this is page 108
<re.Match object; span=(260, 270), match='antibiotic'>
this is page 109
<re.Match object; span=(311, 321), match='antibiotic'>
this is page 110
None
this is page 111
None
this is page 112
None
this is page 113
<re.Match object; span=(398, 408), match='antibiotic'>
this is page 114
<re.Match object; span=(729, 739), match='antibiotic'>
this is page 115
<re.Match object; span=(666, 676), match='antibiotic'>
this is page 116
<re.Match object; span=(1985, 1995), match='antibiotic'>
this is page 117
None
this is page 118
None
this is page 119
None
this is page 120
None
this is page 121
<re.Match object; span=(387, 397), match='antibiotic'>
```

```
this is page 122
<re.Match object; span=(91, 101), match='antibiotic'>
this is page 123
None
this is page 124
None
this is page 125
<re.Match object; span=(1020, 1030), match='antibiotic'>
this is page 126
None
this is page 127
<re.Match object; span=(685, 695), match='antibiotic'>
this is page 128
None
this is page 129
<re.Match object; span=(1311, 1321), match='antibiotic'>
this is page 130
None
this is page 131
None
this is page 132
<re.Match object; span=(760, 770), match='antibiotic'>
this is page 133
<re.Match object; span=(684, 694), match='antibiotic'>
this is page 134
<re.Match object; span=(757, 767), match='antibiotic'>
this is page 135
None
this is page 136
<re.Match object; span=(1973, 1983), match='antibiotic'>
this is page 137
<re.Match object; span=(937, 947), match='antibiotic'>
this is page 138
<re.Match object; span=(277, 287), match='antibiotic'>
this is page 139
<re.Match object; span=(614, 624), match='antibiotic'>
this is page 140
<re.Match object; span=(928, 938), match='antibiotic'>
this is page 141
<re.Match object; span=(72, 82), match='antibiotic'>
this is page 142
<re.Match object; span=(552, 562), match='antibiotic'>
this is page 143
None
this is page 144
None
this is page 145
None
this is page 146
None
this is page 147
None
this is page 148
None
this is page 149
None
```

In [25]:

```
import os
```

In [26]:

```
from os import path
from wordcloud import WordCloud
```

In [27]:

```
d = path.dirname(__file__) if "__file__" in locals() else os.getcwd()
```

pdfReader = PyPDF2.PdfFileReader(open('2019-ar-threats-report-508.pdf', 'rb')) pageData = '' for page in pdfReader.pages: pageData += page.extractText() print(pageData)

The above code was too lengthy so I changed the cell to markdown. However, please feel free to run in the cell in your own analysis. It's long!

In [29]:

```
reader = PyPDF2.PdfFileReader("2019-ar-threats-report-508.pdf")
full_text = ""
```

In [30]:

```
pdf_page_number = 7
```

In [31]:

```
for i in range(1, 150):
    # This block is used to remove the page number at the start of
    # each page. The first if removes page numbers with one digit.
    # The second if removes page numbers with 2 digits and the else
    # statement removes page numbers with 3 digits.
    if pdf_page_number <= 9:
        page_text = reader.getPage(i).extractText().strip()[1:]
    elif pdf_page_number >= 10 and pdf_page_number <= 99:
        page_text = reader.getPage(i).extractText().strip()[2:]
    else:
        page_text = reader.getPage(i).extractText().strip()[3:]

    full_text += page_text.replace("\n", "")
    pdf_page_number += 1
```

In [ ]:

```

```

In [33]:

```
# This looks weird but that's the most practical way to remove double to quad white spa
ces.
full_text = full_text.replace("  ", " ").replace(
    "  ", " ").replace("  ", " ")

with open("transcript_clean.txt", "w", encoding="utf-8") as temp_file:
    temp_file.write(full_text)
```

In [34]:

```
corpus = open("transcript_clean.txt", "r", encoding="utf-8").read()
nlp = spacy.load("en_core_web_sm")

# Our corpus is bigger than the default limit, we will set
# a new limit equal to its length.
nlp.max_length = len(corpus)

doc = nlp(corpus)
```

In [35]:

```
data_list = [["text", "text_lower", "lemma", "lemma_lower",
                "part_of_speech", "is_alphabet", "is_stopword"]]

for token in doc:
    data_list.append([token.text, token.lower_, token.lemma_, token.lemma_.lower(), tok
en.pos_, token.is_alpha, token.is_stop])

csv.writer(open("./tokens.csv", "w", encoding="utf-8",
                newline="")).writerows(data_list)
```

In [36]:

```
data_list = [["text", "text_lower", "label"]]

for ent in doc.ents:
    data_list.append([ent.text, ent.lower_, ent.label_])

csv.writer(open("./entities.csv", "w", encoding="utf-8",
                newline="")).writerows(data_list)
```

from wordcloud import WordCloud, STOPWORDS import matplotlib.pyplot as plt import pandas as pd df = pd.read_csv(r"2019-ar-threats-report-508.csv", encoding ="latin-1") comment_words = '' stopwords = set(STOPWORDS) import csv from wordcloud import WordCloud your_list = [] with open('2019-ar-threats-report-508.csv', 'rb') as f: reader = csv.reader(f) your_list = '\t'.join([i[0] for i in reader]) wordcloud = WordCloud().generate(your_list)

In [37]:

```
reader = PyPDF2.PdfFileReader('2019-ar-threats-report-508.pdf')
```

In [38]:

```
from io import StringIO

from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfdocument import PDFDocument
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.pdfpage import PDFPage
from pdfminer.pdfparser import PDFParser
```

In [39]:

```
def convert_pdf_to_string(file_path):

        output_string = StringIO()
        with open(file_path, 'rb') as in_file:
            parser = PDFParser(in_file)
            doc = PDFDocument(parser)
            rsrcmgr = PDFResourceManager()
            device = TextConverter(rsrcmgr, output_string, laparams=LAParams())
            interpreter = PDFPageInterpreter(rsrcmgr, device)
            for page in PDFPage.create_pages(doc):
                interpreter.process_page(page)

        return(output_string.getvalue())
```

In [40]:

```
def convert_title_to_filename(title):
    filename = title.lower()
    filename = filename.replace(' ', '_')
    return filename
```

In [41]:

```
def split_to_title_and_pagenum(table_of_contents_entry):
    title_and_pagenum = table_of_contents_entry.strip()

    title = None
    pagenum = None

    if len(title_and_pagenum) > 0:
        if title_and_pagenum[-1].isdigit():
            i = -2
            while title_and_pagenum[i].isdigit():
                i -= 1

            title = title_and_pagenum[:i].strip()
            pagenum = int(title_and_pagenum[i:].strip())

    return title, pagenum
```

Below are more works in progress...

writer = PyPDF2.PdfFileWriter()

for page in range(2,4):

```
    writer.addPage(reader.getPage(page))
```

output_filename = './data/original/table_of_contents.pdf'

with open(output_filename, 'wb') as output: writer.write(output)

text = data_func.convert_pdf_to_string('./data/original/table_of_contents.pdf')

In [44]:

```
sns.set(style="ticks",
    rc={
        "figure.figsize": [12, 7],
        "text.color": "white",
        "axes.labelcolor": "white",
        "axes.edgecolor": "white",
        "xtick.color": "white",
        "ytick.color": "white",
        "axes.facecolor": "#5C0E10",
        "figure.facecolor": "#5C0E10"}
    )
```

In [45]:

```
df = pd.read_csv("2019-ar-threats-report-508.csv")
```

In [46]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import base64
import string
import re
from collections import Counter
from nltk.corpus import stopwords
stopwords = stopwords.words('english')
df = pd.read_csv('2019-ar-threats-report-508.csv')
df.head()
```

Out[46]:

	ANTIBIOTIC RESISTANCE THREATS
0	IN THE UNITED STATES
1	2019
2	Revised Dec. 2019
3	This report is dedicated to the 48,700 families
4	who lose a loved one each year to antibiotic r...

In [47]:

```
df.isnull().sum()
```

Out[47]:

```
ANTIBIOTIC RESISTANCE THREATS    25
dtype: int64
```

In [48]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.base import TransformerMixin
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC
from sklearn.feature_extraction.stop_words import ENGLISH_STOP_WORDS
from sklearn.metrics import accuracy_score
from nltk.corpus import stopwords
import string
import re
import spacy
from spacy.lang.en import English
parser = English()
```

```
FutureWarning: The sklearn.feature_extraction.stop_words module is  deprec
ated in version 0.22 and will be removed in version 0.24. The correspondin
g classes / functions should instead be imported from sklearn.feature_extr
action.text. Anything that cannot be imported from sklearn.feature_extract
ion.text is now part of the private API. [deprecation.py:144]
```

In [49]:

```
STOPLIST = set(stopwords.words('english') + list(ENGLISH_STOP_WORDS))
SYMBOLS = " ".join(string.punctuation).split(" ") + ["-", "...", "“", "”"]
class CleanTextTransformer(TransformerMixin):
   def transform(self, X, **transform_params):
        return [cleanText(text) for text in X]
   def fit(self, X, y=None, **fit_params):
        return self
def get_params(self, deep=True):
        return {}

def cleanText(text):
    text = text.strip().replace("\n", " ").replace("\r", " ")
    text = text.lower()
    return text
def tokenizeText(sample):
    tokens = parser(sample)
    lemmas = []
    for tok in tokens:
        lemmas.append(tok.lemma_.lower().strip() if tok.lemma_ != "-PRON-" else tok.low
er_)
    tokens = lemmas
    tokens = [tok for tok in tokens if tok not in STOPLIST]
    tokens = [tok for tok in tokens if tok not in SYMBOLS]
    return tokens
```

Please let me know if there are any questions and/or concerns. This is a work in progress project...