EDA-part-1-Cleaning-Tokenization-Lammatization

June 30, 2022

1 Introduction

1.1 Data Sources

Data that we use are from the following links:

1) Fake and real news dataset. The data sets we use are:
   - Fake.csv.zip
   - True.csv.zip
2) Source based Fake News Classification. The data sets we use are:
   - news_articles.csv.zip
3) REAL and FAKE news dataset. The data sets we use are:
   - news.csv
4) GitHub Repo. The data sets we use are:
   - politifact_fake.csv
   - politifact_real.csv
   - gossipcop_fake.csv
   - gossipcop_real.csv

2 Importing Libraries

Libraries that we will use are:

[1]:
```
import pandas as pd
import numpy as np

import re
import string

import spacy
nlp = spacy.load('en_core_web_sm')#,parse=True,tag=True, entity=True)


import nltk
from nltk.stem.snowball import SnowballStemmer
```

```
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')



import unicodedata              ## to remove accented and special chracters
from textblob import TextBlob   ## to calculate Polarity and Subjectivity␣
 ↪(Sentiment)
# pip install textblob

from sklearn.utils import shuffle
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/miladshirani/nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

3 Functions We Use

We will define several functions to extract web addresses, phone numbers, date, time and any digits in the following. At the end, we will have a function to perform all the work at the same time.

3.1 Dropping text with no words

This function takes the data, and will drop the rows where specific column has less than a minimum number of tokens.

[1]:
```
def empty_text(data, column, min_num_word):
    data["number_of_word"] = data[column].str.split().str.len()

    index = data.loc[data["number_of_word"] < min_num_word].index
    data  = data.drop(index = index, axis = 0)
    data  = data.drop("number_of_word", axis = 1)
    return data
```

3.2 Feature extraction function

This function is compilation of all the other cleaning functions which will be introduced later.

[3]:
```
def find_it(data, keyword, replace):


    if keyword == "link":
        return link_finder(data, replace)
```

```
    elif keyword == "id":
        return id_finder(data, replace)
    elif keyword == "char":
        return char_finder(data, replace)
    elif keyword == "digit":
        return all_digit_finder(data, replace)
    elif keyword == "acc":
        return remove_accented(data)
    elif keyword == "email":
        return email_finder(data, replace)
    elif keyword == "Reuters":
        return reuters_finder(data)
```

3.2.1 Finding Links

This function will find links in a text, calculate the number of links and will replace them with a white space. The pattern used in the following function is from here

[4]:
```
def link_finder(data, replace):
    pattern = '(?:(?:https?|ftp):\/\/)?[\w/\-?=%.]+\.[\w/\-&?=%.]+'

    if replace == False:
        return len(re.findall(pattern, data))
    else:
        return re.sub(pattern," ",data)
```

[5]:
```
### Testing the function

link_finder_data = """Hello www.google.com World http://yahoo.com
and www.berkeley.edu and https://s123rd.edu/12332100kashh123jsa/dhsas?dsajsd
also this one @DanPatrick. pic.twitter.com/mUbKCIWGxB  Shannon Watts␣
 ↪(@shannonrwatts)
"""

link_finder(link_finder_data, True)
```

[5]:
```
'Hello   World   \nand   and  \nalso this one @DanPatrick.    Shannon Watts
(@shannonrwatts) \n'
```

3.2.2 Finding IDs

This function will find and replace the ids in a text.

[6]:
```
def id_finder(data, replace):
    pattern = '@[a-zA-Z0-9]+'

    if replace == False:
        return len(re.findall(pattern, data))
    else:
        return re.sub(pattern," ",data)
```

[7]:
```
### Testing the function

link_finder_data = """Hello www.google.com World http://yahoo.com
and www.berkeley.edu and https://s123rd.edu/12332100kashh123jsa/dhsas?dsajsd
also this one @DanPatrick. pic.twitter.com/mUbKCIWGxB  Shannon Watts␣
 ↪(@shannonrwatts)
"""

no_id = id_finder(link_finder_data, True)

link_finder(no_id, True)
```

[7]:
```
'Hello   World   \nand   and  \nalso this one  .    Shannon Watts ( ) \n'
```

3.2.3 Finding characters

This function will find any non alphabetic and numeric characters in a text and will replace them with a whitespace.

[8]:
```
def char_finder(data, replace):
    pattern = r'[^a-zA-z0-9]'
    if replace == False:
        return len(re.findall(pattern, data))
    else:
        return re.sub(pattern," ",data)
```

[9]:
```
test_data = """
@DanPatrick. pic.twitter.com/mUbKCIWGxB  Shannon Watts (@shannonrwatts)
"""


char_finder(test_data, replace = True)
```

[9]:
```
'  DanPatrick  pic twitter com mUbKCIWGxB  Shannon Watts   shannonrwatts   '
```

3.2.4 Finding accented characters

This function will replace any accented characters with an english alphabets.

[10]:
```
def remove_accented(data):
    new_data = unicodedata.normalize('NFKD', data)\
                          .encode('ascii', 'ignore')\
                          .decode('utf-8', 'ignore')
    return new_data
```

3.2.5 Finding numbers

This function will find any numeric data such as numbers, phone numbers and dates and will replace them with whitespace.

[11]:
```
def all_digit_finder(data, replace):
    """
    the pattern

    "\+?\s?\d+.?\d+.?\d+.?\d+/\(\d+\) \d+.?\d+"

    can be used to get the digits, dates and phone numbers
    of the form

    12.2345
    0.0123
    9876

    dates of the form:

    12/21/2020
    2020-01-22
    2020/02/23

    phone numbers of the form:

    (911) 820 2230
    (911) 820-223
    911-820-2230
    +1-814-929-2533
    0018149292533

    time of the form:

    12:00

    """
    """
    To get phone numbers of the form
    (XXX) XXX-XXX or XXX-XXX-XXXX or +X-XXX-XXX-XXXX
```

```
    we can use the following pattern
    "\+?\d+.\d+.\d+.\d+.\d+/\(\d+\) \d+.\d+"

    to get numbers of the form:
    XXX or XX.XXX
    we can use the following pattern
    "(\s\d+\s/\s\d+\.\d+\s)"

    """

    pattern = "\d+|\s\+?\s?\d+.?\d+.?\d+.?\d+|\(\d+\) \d+.?\d+|\s+\d+"

#     pattern = "\+?\d+.\d+.\d+.\d+.\d+/\(\d+\) \d+.\d+"

    if replace == False:
        return len(re.findall(pattern, data))
    else:
        return re.sub(pattern, " ", data)
```

[12]:
```
date_finder_data = """3

Hello https://s123rd.edu/12332100kashh123jsa/dhsas?dsajsd
and 1st digit: 12.2345 or 2nd digit: 0.0123 or 3rd digit: 9876

and 1st date: 12/21/2020 or 2nd date: 2020-01-22 or 3rd date: 2020/02/23

call 1st number: (911) 820-2230 or 2nd number: 911-820-2230 or
3rd number: (911) 820 2230 or call 4th number: +1-814-929-2533

or call 5th number: 0018149292533 at 1st time: 12:00

"""

test_2 = "4 December 31, 2017The"

print(all_digit_finder(test_2, replace = True))
```

```
 December , The
```

3.2.6 Email Address Finder

This function will find any email address in the text.

[13]:
```
def email_finder(data, replace):

    pattern = "[a-zA-Z0-9]+.?[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]+"

    if replace == False:
```

```
        return len(re.findall(pattern, data))

    else:
        return re.sub(pattern, "emailaddress",data)
```

[14]:
```
email = """

please email ab23_bts@gmail.com or 123asB.asd@as12.com

"""

email_finder(email, True)
```

[14]:
```
'\n\nplease email emailaddress or emailaddress\n\n'
```

3.2.7 Removing Reuters

In the text of the data, we realized that there are a lot of text that are reported by the Reuters and we decided to remove this word from the data.

[15]:
```
def reuters_finder(data):

    pattern = "Reuters"

    return re.sub(pattern, " ", data)
```

[16]:
```
test_data = "WASHINGTON (Reuters) said"


reuters_finder(test_data)
```

[16]:
```
'WASHINGTON ( ) said'
```

3.3 Cleaning Data:

This function will clean the data, tokenize and lemmatize the data. It uses the "find_it" and "tokenizing_lemmatizing" functions.

[17]:
```
def cleaning(data):
    pre_cleaning_list = [ "Reuters", "email", "id", "link", "digit", "char",␣
 ↪"acc"]

    for item in pre_cleaning_list:
        data = find_it(data, item, replace = True)

    tokenized_data = tokenizing_lemmatizing(data)
```

```
    return tokenized_data
```

3.3.1 Tokenizing and Stemming

This function takes the cleaned text and will tokenize and stem the text and if the token is in the stopwords, it will remove them from the text.

[2]:
```
def tokenizing_stemming(data):

    stemmer = SnowballStemmer(language="english")
    basic_token_pattern = r"(?u)\b\w\w+\b"
    tokenizer = RegexpTokenizer(basic_token_pattern)
    token_list = tokenizer.tokenize(data)
    return [stemmer.stem(token.lower()) for token in token_list if token.
 ↪lower() not in stop_words]
#     return [token.lower() for token in token_list if token.lower() not in␣
 ↪stop_words]
```

3.3.2 Tokenizing and Lemmatizing

This function takes the cleaned text and will tokenize and lemmatize the text and if the token is in the stopwords, it will remove them from the text.

[19]:
```
def tokenizing_lemmatizing(data):

    lemmatizer = WordNetLemmatizer()
    basic_token_pattern = r"(?u)\b\w\w+\b"
    tokenizer = RegexpTokenizer(basic_token_pattern)
    token_list = tokenizer.tokenize(data)
#     return [stemmer.stem(token.lower()) for token in token_list if token.
 ↪lower() not in stop_words]
    return [lemmatizer.lemmatize(token.lower(), pos = "v") for token in␣
 ↪token_list
            if token.lower() not in stop_words]
```

3.4 For GloVe

This function will clean the text and will return a cleaned text while it has the stopwords because we are going to use it in the notebook, "Modeling_GloVe" in the modeling section.

[20]:
```
def for_glove(data):
    p_link = '(?:(?:https?|ftp):\/\/)?[\w/\-?=%.]+\.[\w/\-&?=%.]+'
    p_id = '@[a-zA-Z0-9]+'
    p_char = r'[^a-zA-z0-9]'
    p_num = "\d+|\s\+?\s?\d+.?\d+.?\d+.?\d+|\(\d+\) \d+.?\d+|\s+\d+"
    p_email = "[a-zA-Z0-9]+.?[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]+"
```

```
    p_reut = "[a-zA-Z]+\s+reuters\s+"

    d_email   = re.sub(p_email, " ", data)
    d_id      = re.sub(p_id   , " ", d_email)
    d_link    = re.sub(p_link, " ", d_id)
    d_num     = re.sub(p_num, " ", d_link)
    d_char    = re.sub(p_char, " ", d_num)
    d_reuters = re.sub(p_reut, " ", d_char)

    cleaned_data = unicodedata.normalize('NFKD', d_reuters)\
                                .encode('ascii', 'ignore')\
                                .decode('utf-8', 'ignore')

    return word_tokenize(cleaned_data)
```

3.5 Test the Functions

In this subsection we want to check and test the functions we introduced.

[21]:
```
to_test = """2 @Flatiron

Hello www.google.com World http://yahoo.com
and www.berkeley.edu and https://s123rd.edu/12332100kashh123jsa/dhsas?dsajsd
also this one @DanPatrick. pic.twitter.com/mUbKCIWGxB  Shannon Watts␣
 ↪(@shannonrwatts)

would could might

@DanPatrick. pic.twitter.com/mUbKCIWGxB  Shannon Watts (@shannonrwatts)


Hello https://s123rd.edu/12332100kashh123jsa/dhsas?dsajsd
and 1st digit: 12.2345 or 2nd digit: 0.0123 or 3rd digit: 9876

and 1st date: 12/21/2020 or 2nd date: 2020-01-22 or 3rd date: 2020/02/23

call 1st number: (911) 820-2230 or 2nd number: 911-820-2230 or
3rd number: (911) 820 2230 or call 4th number: +1-814-929-2533

or call 5th Number: 0018149292533 at 1st time: 12:00

please email ab23_bts@gmail.com or 123asB.asd@as12.com

"""

# print(cleaning(to_test))
print(for_glove(to_test))
```

```
['Hello', 'World', 'and', 'and', 'also', 'this', 'one', 'Shannon', 'Watts',
'would', 'could', 'might', 'Shannon', 'Watts', 'Hello', 'and', 'st', 'digit',
'or', 'nd', 'digit', 'or', 'rd', 'digit', 'and', 'st', 'date', 'or', 'nd',
'date', 'or', 'rd', 'date', 'call', 'st', 'number', 'or', 'nd', 'number', 'or',
'rd', 'number', 'or', 'call', 'th', 'number', 'or', 'call', 'th', 'Number',
'at', 'st', 'time', 'please', 'email', 'or']
```

4 Importing Data and EDA

In this part we import the data.

[22]:
```
path = "../EDA/Raw_Data/"
## DataSet 1
fake = pd.read_csv(path + "Fake.csv")
true = pd.read_csv(path + "True.csv")

## DataSet 2
g_fake = pd.read_csv(path + "gossipcop_fake.csv")
g_real = pd.read_csv(path + "gossipcop_real.csv")
p_fake = pd.read_csv(path + "politifact_fake.csv")
p_real = pd.read_csv(path + "politifact_real.csv")

## DataSet 3
articles = pd.read_csv(path + "news_articles.csv")


## DataSet 4
news = pd.read_csv(path + "news-II.csv")


real_fake = {"Real" : "True", "REAL": "True" , 1:"True",
             "FAKE": "Fake", "Fake": "Fake", 0:"Fake"}
```

5 Data Cleaning of Each DataFrame

In this section, we clean the text data in each dataframe separately.

5.1 Fake_true

We first create a column in each dataframe called “label” to assign a label to the data, then we concatenate them to each other. After that, we drop the columns `title`, `date` and `subject`. Since `fake_true` dataframe is relatively big, we divide it into 12 dataframes and then we do the cleaning process to each dataframe. After cleaning each dataframe, we concatenate them again to make a one cleaned dataframe and then we save it into the “cleaned” folder.

[23]:
```
fake["label"] = "Fake"
true["label"] = "True"
```

```
fake_true = pd.concat([fake, true], axis = 0)
fake_true.drop(['title', "date", "subject"], inplace = True, axis = 1)

fake = empty_text(fake, "text", 5)
true = empty_text(true, "text", 5)

len(fake_true)
```

[23]:
```
44898
```

[24]:
```
ft_1  = fake_true.iloc[:4000, :].copy()
ft_2  = fake_true.iloc[4000:8000, :].copy()
ft_3  = fake_true.iloc[8000:12000, :].copy()
ft_4  = fake_true.iloc[12000:16000, :].copy()
ft_5  = fake_true.iloc[16000:20000, :].copy()
ft_6  = fake_true.iloc[20000:24000, :].copy()
ft_7  = fake_true.iloc[24000:28000, :].copy()
ft_8  = fake_true.iloc[28000:32000, :].copy()
ft_9  = fake_true.iloc[32000:34000, :].copy()
ft_10 = fake_true.iloc[34000:36000, :].copy()
ft_11 = fake_true.iloc[36000:40000, :].copy()
ft_12 = fake_true.iloc[40000:, :].copy()

list_of_df = [ft_1, ft_2, ft_3, ft_4, ft_5, ft_6, ft_7,
              ft_8, ft_9, ft_10, ft_11, ft_12]
```

[25]:
```
sid = SentimentIntensityAnalyzer()
sentiment_values = ['neg', 'neu', 'pos', 'compound']

for i, item in enumerate(list_of_df):
    print(i+1)

    item["cleaned"]  = item["text"].apply(lambda x: cleaning(x.lower()))
    print("\tCleaning Done")

    item["for_glove"]  = item["text"].apply(lambda x: for_glove(x))
    print("\tFor GloVe Done")

    item["num_urls"] = item["text"].apply(lambda x: link_finder(x, False))
    print("\tLink Done")

    item[sentiment_values] = item["text"].apply(sid.polarity_scores).apply(pd.
 ↪Series)
    print("\tSentiment Done")
```

```
1
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
2
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
3
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
4
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
5
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
6
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
7
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
8
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
9
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
10
        Cleaning Done
        For GloVe Done
```

```
        Link Done
        Sentiment Done
11
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
12
        Cleaning Done
        For GloVe Done
        Link Done
        Sentiment Done
```

[26]:
```
fake_true_cleaned = ft_1.copy()
to_concat = [ft_2, ft_3, ft_4, ft_5, ft_6, ft_7,
             ft_8, ft_9, ft_10, ft_11, ft_12]
for item in to_concat:
    fake_true_cleaned = pd.concat([fake_true_cleaned, item], axis = 0)
fake_true_cleaned.to_csv("../EDA/cleaned/fake_true.csv")
print("Saved to Directory")
```

```
Saved to Directory
```

5.2 gossip

We first create a column in each dataframe called "label" to assign a label to the data, then we concatenate them to each other. After that, we drop the columns `id`, `news_url`, and `tweet_ids`. Since `gossip` dataframe is not relatively big, we do the cleaning process to it without dividing it to different dataframes. After cleaning each dataframe, we concatenate them again to make a one cleaned dataframe and then we save it into the "cleaned" folder.

[27]:
```
g_fake["label"] = "Fake"
g_real["label"] = "True"
p_fake["label"] = "Fake"
p_real["label"] = "True"

gossip = pd.concat([g_fake,g_real, p_fake, p_real], axis = 0)
gossip.drop(["id", "news_url", 'tweet_ids'], inplace = True, axis = 1)
gossip.rename(columns = {"title": "text"}, inplace = True)


gossip = empty_text(gossip, "text",5)

gossip.isnull().sum()
```

[27]:
```
text     0
label    0
dtype: int64
```

[28]:
```
sid = SentimentIntensityAnalyzer()
sentiment_values = ['neg', 'neu', 'pos', 'compound']


gossip["cleaned"] = gossip["text"].apply(lambda x: cleaning(x))
print("Cleaning Done")

gossip["for_glove"]  = gossip["text"].apply(lambda x: for_glove(x))
print("For GloVe Done")

gossip["num_urls"]= gossip["text"].apply(lambda x: link_finder(x, False))
print("Link Done")

gossip[sentiment_values] = gossip["text"].apply(sid.polarity_scores).apply(pd.
  ↪Series)
print("Sentiment Done")

gossip.to_csv("../EDA/cleaned/gossip.csv")
print("Saved to Directory")
```

```
Cleaning Done
For GloVe Done
Link Done
Sentiment Done
Saved to Directory
```

5.3 articles_en

We first map the values in the column `lable` to `Fake` and `True`, then we drop several columns. Since `article_en` dataframe is not relatively big, we do the cleaning process to it without dividing it to different dataframes. After cleaning each dataframe, we concatenate them again to make a one cleaned dataframe and then we save it into the "cleaned" folder.

[29]:
```
articles["label"] = articles["label"].map(real_fake)

to_drop = ['author', 'published','title','site_url','main_img_url', 'type',
           'title_without_stopwords','text_without_stopwords', 'hasImage']

articles.drop(columns = to_drop, inplace = True, axis = 1)

articles_en = articles[articles["language"] == "english"][["text", "label"]]

articles_en.isna().sum()
articles_en.dropna(inplace = True, axis = 0)
articles_en.isna().sum()
articles_en = empty_text(articles_en, "text", 5)
len(articles_en)
```

[29]: 1943

[30]:
```
sid = SentimentIntensityAnalyzer()
sentiment_values = ['neg', 'neu', 'pos', 'compound']

articles_en["cleaned"] = articles_en["text"].apply(lambda x: cleaning(x))
print("Cleaning Done")

articles_en["for_glove"]  = articles_en["text"].apply(lambda x: for_glove(x))
print("For GloVe Done")

articles_en["num_urls"]= articles_en["text"].apply(lambda x: link_finder(x,␣
 ↪False))
print("Link Done")

articles_en[sentiment_values] = articles_en["text"]\
.apply(sid.polarity_scores).apply(pd.Series)
print("Sentiment Done")
articles_en.to_csv("../EDA/cleaned/articles_en.csv")
print("Saved to Directory")
```

```
Cleaning Done
For GloVe Done
Link Done
Sentiment Done
Saved to Directory
```

5.4 news

We first map the values in the column `lable` to `Fake` and `True`, then we drop columns `Unnamed: 0`, `title`. Since `gossip` dataframe is not relatively big, we do the cleaning process to it without dividing it to different dataframes. After cleaning each dataframe, we concatenate them again to make a one cleaned dataframe and then we save it into the "cleaned" folder.

[31]:
```
news["label"] = news["label"].map(real_fake)
news.drop(['Unnamed: 0', 'title'], axis = 1, inplace = True)

news = empty_text(news, "text", 5)
```

[32]:
```
news.isna().sum()
```

[32]:
```
text     0
label    0
dtype: int64
```

[33]:
```
sid = SentimentIntensityAnalyzer()
sentiment_values = ['neg', 'neu', 'pos', 'compound']
```

```
news["cleaned"] = news["text"].apply(lambda x: cleaning(x))
print("Cleaning Done")

news["for_glove"]  = news["text"].apply(lambda x: for_glove(x))
print("For GloVe Done")

news["num_urls"]= news["text"].apply(lambda x: link_finder(x, False))
print("Cleaning Done")

news[sentiment_values] = news["text"].apply(sid.polarity_scores).apply(pd.
  ↪Series)
print("Sentiment Done")

news.to_csv("../EDA/cleaned/news.csv")
print("Saved to Directory")
```

```
Cleaning Done
For GloVe Done
Cleaning Done
Sentiment Done
Saved to Directory
```

6 Next

At the end of this notebook, we have cleaned data and for visualizations, we will go to the next EDA notebook