Practice 2

Data cleaning and preprocessing

Data cleansing is a very important step in any machine learning model, but more so for NLP. Without a cleanup process, the data set is often a collection of words that the computer does not understand. Typically, the following procedures are used for text preprocessing:
Tokenization – splitting sentences to words.
Normalization – lowercasing, lemmatization or stemming.
Lemmatization put forms of each word into a common base or root taking into account the morphological analysis of words.
Stemming is the process of reducing inflection in words to their root forms by removing the suffixes or prefixes used with a word.
Noise entities removal – punctuation, stopwords, etc. deletion.

Let's analyze the main punctuation marks in real and fake news:

```
from collections import defaultdict
# define a function to get list of words according to the target class
def create_words(target):
   words = []
   for x in df[df['target']==target]['text'].str.split():
       for i in x:
           words.append(i)
   return words
```

Define a function counts most occurring punctuation symbols:

```
import string
# create a list with punctuation symbols
punctuation_list=list(string.punctuation)
value_list=[]
# define a function counts most occuring punct. symbols
# according to the target class
# with arguments: dataframe, function_form_words_list, list_of_targets
def most_occuring(dataset,fun,target):
   d=defaultdict(int)
   for j in range(0,len(target)):      # for every type of target
       words=fun(target[j])       # call function create_words()
       for i in words:
```

```
            if i in punctuation_list: # if word is punctuation symbol
                d[i]+=1               # count it
        # sort to 10 symbols by their count in descending order
        top=sorted(d.items(),key=lambda x: x[1],reverse=True)[:10]
        # separate punctuation symbols from their quantification
        x_items,y_counts=zip(*top)
        # append symbols and their quantification to value_list
        value_list.append(x_items)
        value_list.append(y_counts)

    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(15,5))

    ax1.bar(value_list[0],value_list[1],color="lightcoral",
    edgecolor='black', linewidth=1.2)
    ax1.set_title("Real")

    ax2.bar(value_list[2],value_list[3],color="purple",
    edgecolor='black', linewidth=1.2)
    ax2.set_title("Fake")

    plt.suptitle("Punctuations in text")
    plt.show()

most_occuring(df,create_words,[1,0])
```

Task 1: What is the difference between the object `dict` and `defaultdict`?
Task 2: Why do we need to explore punctuation in text?

Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK (Natural Language Toolkit) in python has a list of stop words stored in 16 different languages.

Print the list of stop words in English:

```
import nltk
from nltk.corpus import stopwords
print (stopwords.words('english')
```

So, in order to make the text clearer for further analysis, we need to remove punctuation marks and stop words.

Stemming and Lemmatization

Words in a sentence can have different forms, which can also complicate further analysis. Thus, procedures such as Stemming and Lemmatization are used to overcome this.

Stemming and **Lemmatization** are text normalization (or sometimes called word Normalization) techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing. Stemming and Lemmatization have been studied, and algorithms have been developed in Computer Science since the 1960's.

Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language. Stems are created by removing the suffixes or prefixes used with a word.

Lemmatization put forms of each word into a common base or root taking into account the morphological analysis of the words. In this example we will use WordNetLemmatizer (on the link you can find some other approaches to lemmatization https://www.machinelearningplus.com/nlp/lemmatization-examples-python/)

Let's look on stemming and lemmatization on example.

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer

#A list of words to be stemmed
word_list = ["friend", "friendship", "friends",
"friendships","stabil","destabilize","misunderstanding","railroad","mo
onlight","football"]
print("{0:20}{1:20}{2:20}".format("Word","Porter Stemmer","lancaster
Stemmer"))
print(word_list)

ps = PorterStemmer()
ls = LancasterStemmer()

for word in word_list:

print("{0:20}{1:20}{2:20}".format(word,ps.stem(word),ls.stem(word)))
```

The first column is initial word, the second - the word after stemming, the third - after lemmatization.

Task 3: Output the result of words stemming and lemmatization for the first clean True-False text.

Text preprocessing using universal text cleaning:

The **re** (Regular expression operations) module provides a set of powerful regular expression facilities, which allows you to quickly check whether a given string *matches* a given pattern (using the match function), or *contains* such a pattern (using the search function).

```
import re
```

```
import string
from nltk.corpus import stopwords

def clean_text(text):
    lemmatizer = WordNetLemmatizer()
    stopwords_english = stopwords.words('english')
    text= re.sub('\[[^]]*\]', '', text)
    # remove stock market tickers like $GE
    text = re.sub(r'\$\w*', '', text)
    #removal of html tags
    review =re.sub(r'<.*?>',' ',text)
    # remove old style retweet text "RT"
    text = re.sub(r'^RT[\s]+', '', text)
    # remove hyperlinks
    text = re.sub(r'https?:\/\/.*[\r\n]*', '', text)
    # remove hashtags
    # only removing the hash # sign from the word
    text = re.sub(r'#', '', text)
    text = re.sub("["
                  u"\U0001F600-\U0001F64F"  # removal of emoticons
                  u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                  u"\U0001F680-\U0001F6FF"  # transport & map symbols
                  u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                  u"\U00002702-\U000027B0"
                  u"\U000024C2-\U0001F251"
                  "]+",' ',text)
    text = re.sub('[^a-zA-Z]',' ',text)
    text = text.lower()
    text_tokens =word_tokenize(text)

    text_clean = []
    for word in  text_tokens:
        if (word not in stopwords_english and  # remove stopwords
                word not in string.punctuation):  # remove punctuation
            lem_word =lemmatizer.lemmatize(word)  # lemmatizing word
            text_clean.append(lem_word)
    text_mod=[i for i in text_clean if len(i)>2]
    text_clean=' '.join(text_mod)
    return  text_clean
```

Task 4: give a detailed verbal explanation to the function `clean_text()`

Apply cleaning function to dataset:

```
df['clean_text']=df['text'].apply(lambda x: clean_text(x))
```

Look at the top words of the dataset:

```
df['clean_temp']=df['clean_text'].apply(lambda x: str(x).split())
top=Counter([word for li in df['clean_temp'] for word in li])
temp_2=pd.DataFrame(top.most_common(20))
temp_2.columns=["common_words",'frequency']
temp_2.style.background_gradient(cmap='Blues')
```

Visualize the results:

```
data_1=df[df['target']==1]
pd.Series(' '.join([i for i in
data_1.clean_text]).split()).value_counts()[:50].plot
(kind='bar',figsize=(20,8),color='yellow'
                 ,edgecolor='black',title='Real')
plt.show()

data_0=df[df['target']==0]
pd.Series(' '.join([i for i in
data_0.clean_text]).split()).value_counts()[:50].plot
(kind='bar',figsize=(20,8),color='green'
                                                  ,edgecolor='blac
k',title='Fake')
plt.show()
```

Task 5: output top 50 most frequently occurring words after stemming (`PorterStemmer`) without stopwords punctuation symbols etc. for real and fake news separately.

Task 6: plot the results with plotly. Analyze the results.

Task 7: describe the difference between stemming and lemmatization. List the advantages and disadvantages of both methods.

Questions for Practice 2:

1. What procedures are used for text preprocessing?
2. Give the definition to tokenization, lemmatization, stemming, noise entities removal?
3. Why is it necessary to perform the above procedures?
4. List the libraries that are useful for text analysis.
5. Give the definition to stop-words.