Natural Language Processing (NLP) is concerned with the interaction between natural language and the computer. It is one of the major components of Artificial Intelligence (AI) and computational linguistics. It provides a seamless interaction between computers and human beings and gives computers the ability to understand human speech with the help of machine learning. The fundamental data type used to represent the contents of a file or a document in programming languages (for example, C, C++, JAVA, Python, and so on) is known as string. In this toolkit, we will explore various operations that can be performed on strings that will be useful to accomplish various NLP tasks.

In this tutorial, we will cover the following tasks on the example of the text classification problem:

1. Exploratory data analysis, Data cleaning and preprocessing.
2. Vectorization methods.
3. Simple models for classification.
4. Transformers for classification.

A great platform for beginner Data scientists is https://www.kaggle.com/ Here you can find lots of free datasets and code examples for learning. And when you feel more confident, you can try to take part in competitions. So the first task is to sign up on Kaggle.
The second useful tool for DS is colab.research.google.com. To your own taste, you can use colab or a standard local jupyter notebook (in this case you need to install all dependencies by yourself).

During the practical work you need to rewrite, understand and launch all the code presented in this book. When you see a paragraph starting with **'Task:', you need to write a definition, describe your opinion, or write a program in a corresponding cell type.**

During practical work you are going to apply different algorithms for text classification on 2 classes: true and fake news. So it would be better to write all practices in one file. Some algorithms are given on simple examples. It would be better if you implement these examples in separate ipynb file, and after that apply it to your text classification task in main ipynb file with true-false data analysis.

Good luck!

Practice 1

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is the process by which the data analyst becomes acquainted with their data to drive intuition and begin to formulate testable hypotheses. This process typically makes use of descriptive statistics and visualizations.

To visualize text data, generally, we use the word cloud but there are some other techniques also, which we can try to visualize the data such as Frequency Distribution and Frequency Graph. Further we will try to construct such distributions.

Here, we will go over steps done in a typical machine learning text pipeline to clean data.

One of the basic tasks in NLP is the classification task. The dataset: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset is perfect for learning the basics. Please, download a dataset for practicing by your own.

Import libraries

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Numpy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

Loading The Dataset:

```
fake = pd.read_csv('/content/Fake.csv', delimiter = ',')
true = pd.read_csv('/content/True.csv', delimiter = ',')
```

Observing the top 5 rows of fake and true data:

```
fake.head()
```

```
true.head()
```

Now we need to merge these two datasets to make further processing easier.
Also we need to add an extra column as 'temp' to distinguish between news as 1 – true_news and 0 – fake_news:

```
fake['target'] = 0
true['target'] = 1

df = pd.DataFrame()
df = true.append(fake)
```

Observing the top 5 rows of dataset:

```
df.head()
```

Checking the shape of loaded data frames:

```
print(df.shape)
```

Exploring the data:

```
df.info()
```

Shuffle the data to improve further results:

```
from sklearn.utils import shuffle
# Shuffle the data
df = shuffle(df).reset_index(drop=True)
df.head()
```

Check the distribution of the target class:
The method `countplot` from the `seaborn` library shows the counts of observations in each categorical bin using bars.

```
sns.countplot(x='target',data=df,palette=['orange','purple'])
```

Check if there are null values in the sample by summing all null values:

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

Check data types of the columns:

```
df.dtypes
```

Compare subject vs target distribution:
This shows whether there is any dependence on the distribution of real and fake news by topic.

```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15,5))
sns.countplot(x='subject',data=df,hue='target')
```

Task 1: describe the type of style `'fivethirtyeight'` used by pyplot library.
Task 2: comment on the distribution results.

Combine text and title columns:

```
df['text']=df['text']+" "+df['title']
```

Delete insignificant columns from the dataframe:

```
df.drop(['title','subject','date'],axis=1,inplace=True)
df.head()
```

`df.drop` removes rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names. If **inplace==False**, return a copy. Otherwise, do operation **`inplace`** and return None.

Calculate and plot the word number distribution in real & fake news:

Number of words in real news are lying in the range of 0 to 1500 whereas in the case of fake news it lies in the range of 0 to 2000.This shows that number of words in the fake news are usually higher than that of real news.

Count and plot average length of words in real & fake news.

```
avg_len_word=df[df['target']==1].text.str.split().map(lambda
x:np.mean([len(word) for word in x]))
avg_len_word.plot(kind='hist',bins = 25,
edgecolor='black',color='lightcoral',title='avg length of words in
true')
plt.show()
avg_len_word=df[df['target']==0].text.str.split().map(lambda
x:np.mean([len(word) for word in x]))
avg_len_word.plot(kind='hist',bins = 25,
edgecolor='black',color='lightyellow',title='avg length of words in
false')
plt.show()
```

Task 3: comment the results, shown in plots. Try different number of bins.

Analyze the top stop words in the real and fake news using **`Counter`**:
`Counter` is a container that will hold the count of each of the elements present in the container. The counter is a sub-class available inside the dictionary class.

```
from collections import Counter
# add a column ‘temp_list’ with separate words to dataframe
df['temp_list']=df['text'].apply(lambda x: str(x).split())
# count the frequency of words occurrence
top=Counter([word for li in df['temp_list'] for word in li])
# form the top 20 most frequently occurring words
temp_1=pd.DataFrame(top.most_common(20))
# set names to columns in temp_1 dataframe and display it in color
temp_1.columns=["most_common_words","frequency"]
temp_1.style.background_gradient(cmap='Blues')
```

Plotting the result with plotly:

```
import plotly.express as pe
import plotly.figure_factory as ff
fig = pe.bar(temp_1, x="frequency", y="most_common_words",
title='Commmon Words in Text', orientation='h',
             width=700, height=700, color='most_common_words')
fig.show()
```

Task 4: analyze the resulting plots, do you think such feature as most_common_words is useful for the further text classification?
Task 5: guess, how we can modify this feature to get more useful results in text data description.

Questions for Practice 1:
1. Give the definition to Natural Language Processing (NLP) field.
2. What do we need to do during Exploratory Data Analysis (EDA)?
3. Describe the classification task in the context of text analysis.
4. List the main purposes of the following libraries: Pandas, NumPy, Matplotlib, Seaborn, NLTK, Scikit-learn.
5. Why is it necessary to shuffle the data before analysis?
6. Describe the dataframe object.