بسمه تعالی

دانشکده مهندسی کامپیوتر

بهار **1401**

پردازش زبان های طبیعی

نام استاد: دکتر مینایی

دانشجو: محمد صدرا خاموشی‌فر - رضا علیدوست

فاز اول پروژه

Data Source

We use New York Times data set which is very popular and complete. You can check it from this link. This source has a huge amount of news about a wide variety of sections including movies, videos, arts, sports, ….
For each section, there are tons of data that you can easily fetch and work with it.

Data Collection

Method

We did some research for scrapping or crawling data from different websites but we faced some small problems. The main problem with crawling data was the amount of it. We had some limitations in fetching data from NYTimes since it didn't allow us to fetch much data. So, we tried to use NYTimes APIs which were provided for developers but first you must create an account on their site.

Steps

1. Create an account on nytimes.com
2. Get your api_key
3. Send get requests to the NYTimes API for each section to get some news.
4. Extract the heading and abstract of each news in returned response and build our data-set ;)

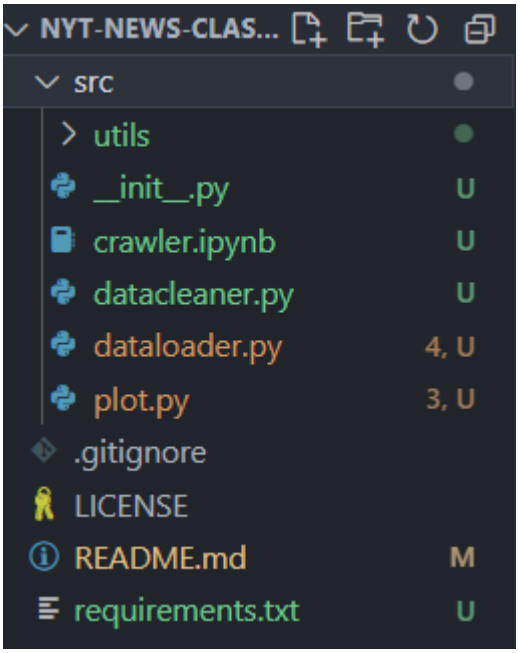
Tool and Technology

Using python requests module for sending requests to the nytimes and due to some reasons, we use notebook Colab for running our scripts. The code is provided in local and also in jupyter notebook.

Data Formats

The Format of our data is kind of a .csv file. It has over 15000 rows and in each row, we have 2 columns. The first column is for exact news content and the second row is the name of the category.

You can see the structure of the whole project in the below image;



- datacleaner.py: file for cleaning the data we some helper functions.
- plot.py: file which draws and saves the bar charts and word cloud with helper functions
- dataloader.py: file for fetching data from nytimes website and saving them
- utils.py: contains some helper functions
- crawler.ipynb: notebook file contains the above code.

Data Preprocessing

We did some preprocessing on our raw data to help our model learn better and in a much easier way.

Here are some techniques we used in our preprocessing task:

- To lower case: It's one of the most popular techniques in NLP tasks for preprocessing since our model should not be case-sensitive. So we convert all words in the news to the lower case.
- Stemming: It is another technique for preprocessing data before training the model. Different forms of words are used in news and we should reduce them to the normal form to prevent our model from being bias. Stemming tries to reduce affixes from the words for instance by stemming we can convert studies to **'studi'**. As you can see, by stemming we usually can not get the correct form of the word, and sometimes it's better to not use it in some classification tasks but we consider it in our preprocessing we'll find out whether it is good for our model or not after building and training our model. In this task, we use PorterStemmer from the NLTK module in Python.
- Lemmatization: lemmatization is another type of reducing the word for getting the normal form of the word. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. In lemmatization, we try to get the core of the word, for example, the word 'was' after lemmatizing converted to 'be' or 'better' converted to 'well'. Sometimes it is preferred over the stemming in classification tasks since it gives better results. We use the NLTK module for python to lemmatize the words. One of our big challenges in using a lemmatizer was that usually, words stayed unchanged after lemmatizing. We find out a proper solution for this problem. The solution is that we should pass the Part of Speech of each word to the lemmatizer. This task is called POS Tagging(**part-of-speech tagging).**
- Remove Punctuation: Another general task in preprocessing is removing punctuation. Since they do not have any information for our task. We remove them with help of regular expression.
- Remove Special characters and numbers: Numbers and special characters do not give us much information about the text and by eliminating them we won't lose any

important information also our model can perform better after removing them. We again use regular expressions for this task.

- Remove extra whitespaces: Removing extra spaces and whitespaces help our model to perform better and we also have an easier job in tokenization.
- Removing words with lengths less than 3: As we work on English words, we know that words with lengths of 2 or 1 can not put much information on a sentence so we decided to eliminate them from the sentences.
- Removing stopwords: Same as numbers and special characters, stopwords do not give us much information about a text so to reduce our vocabulary size and make the classification task much easier we decided to remove stopwords from the news. We can do this with the help of the NLTK module in python.

Sentences tokenization

We tokenize the sentences using nltk module provided for NLP tasks. We did it with help of the **sent_tokenize** method. The sent_tokenize function uses an instance of PunktSentenceTokenizer from the nltk. tokenize. Punkt module, which is already been trained and thus very well knows to mark the end and beginning of sentence at what characters and punctuation. **An obvious question in your mind would be why sentence tokenization is needed when we have the option of word tokenization**. Imagine you need to count average words per sentence, how you will calculate? For accomplishing such a task, you need both NLTK sentence tokenizer as well as NLTK word tokenizer to calculate the ratio. We can separate sentences in a news before tokenizing them into words. You can check out the number of sentences per news section in the attachment files. These sentence tokens help in understanding the context or developing the model for the NLP tasks.

Word tokenization

One of the most important parts of NLP tasks is the tokenization of the word. Tokenization **allows machines to read texts.** We need to tokenize sentences in order to do some preprocessing mentioned above. The NLTK has different types of word tokenizers but for this task we choose TreebankWordTokenizer. In this task, we used some tokenizer but it doesn’t make much difference cause the data was not too complicated and can be

tokenized easily. The Treebank tokenizer uses regular expressions to tokenize text as in Penn Treebank. This implementation is a port of the tokenizer sed script written by Robert McIntyre. Check this link for more info about this tokenizer.

Data cleaning

We have a separate file named **datacleaner.py** for cleaning data. Cleaning is the combination of the preprocessing techniques mentioned lately. After cleaning the data we save data in a new file named **cleaned_news.csv.** It has almost 15000 news rows and in each row, we have 2 columns, first one is the clean news and the second row is the category of the news.

Size of data

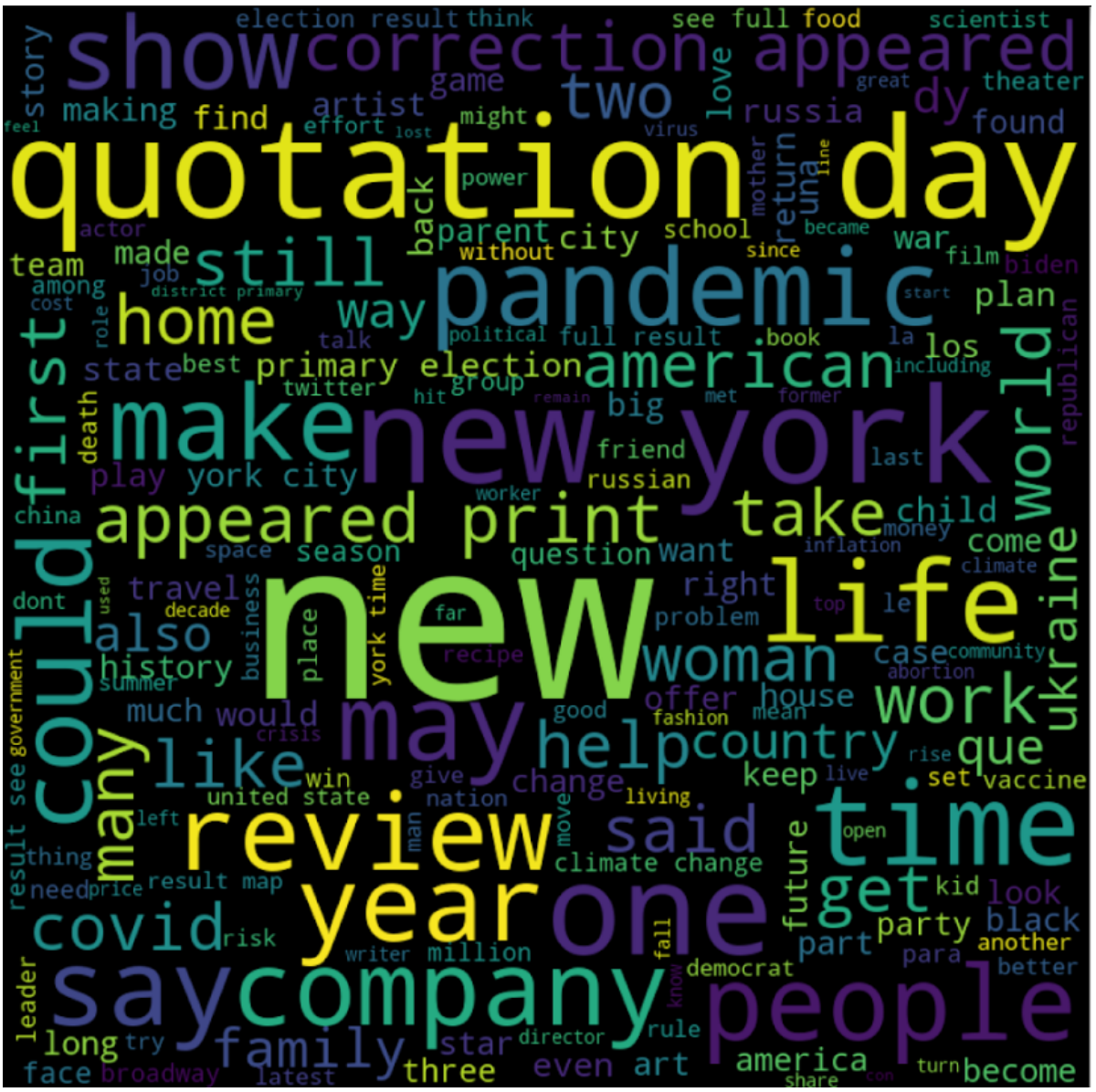
We have 14942 news before cleaning the data which is saved in the **shuffle_data.csv** and after cleaning the data we have 14902 cleaned news.

Data Tagging

While we fetching data from the nytimes website, we can also get the category or section of each news and tag each news with its section. We assign each news a specific tag and use them as our classes for nlp task.

Data Metrics

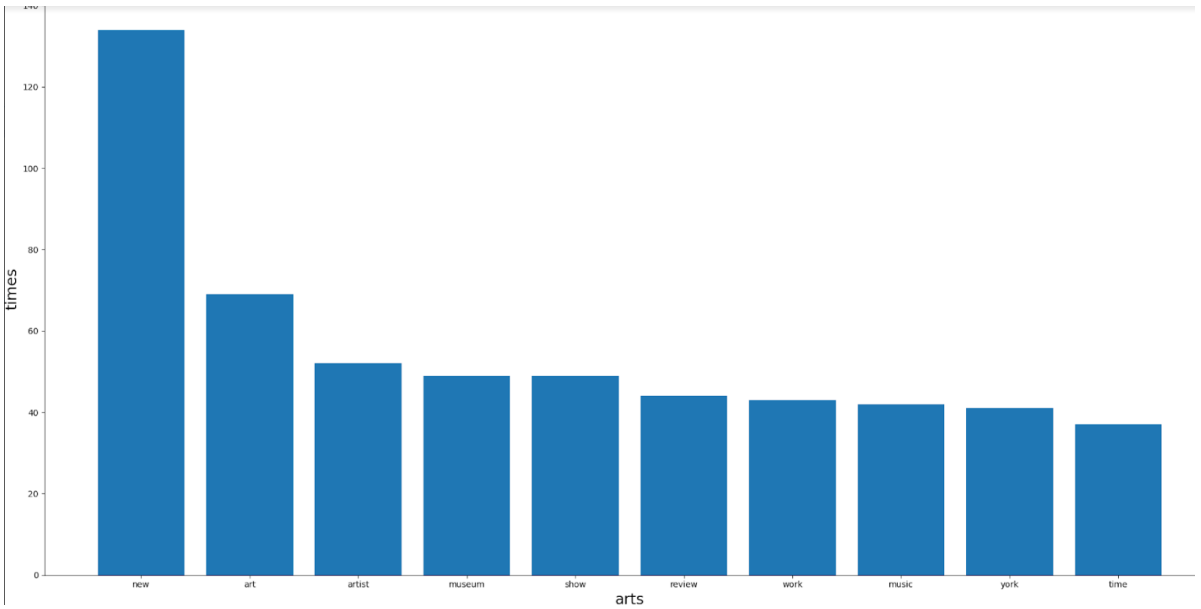
You can see the wordcloud of the whole news. This file is available in the attached file.

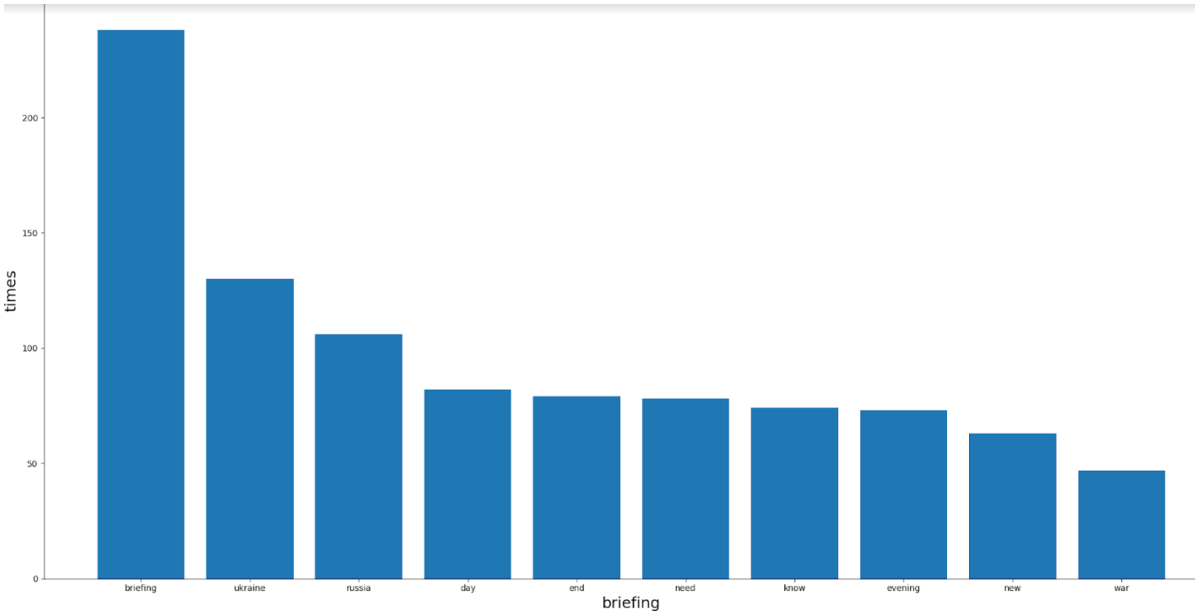


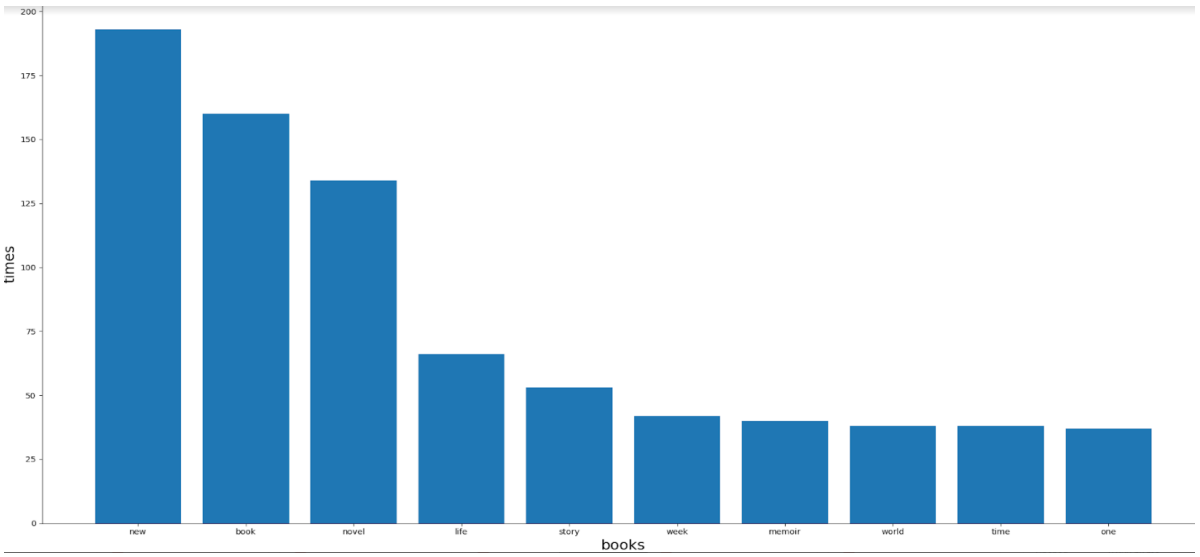
Also in the below table, you can check the count of sentences, words, types, vocab, and units per each section. You can find this table in the attachment.

	unit num	sent num	token num	vocab num
arts	499	1140	9172	3915
books	500	1099	8593	3751
briefing	500	1049	4678	1566
business	494	974	8506	3030
climate	500	1172	9653	2853
corrections	399	463	2817	34
en español	500	1291	12651	4436
fashion	500	1097	7225	3258
food	500	1069	7980	3342
health	500	1104	9109	2570
magazine	498	1203	7015	3466
movies	500	1021	8265	3641
new york	500	1088	9410	3194
obituaries	185	438	3658	2096
opinion	500	1159	6700	2746
parenting	145	326	1799	1002
podcasts	500	1115	6599	2667
real estate	500	1151	7979	2262
science	498	1037	8446	3324
sports	499	1251	8417	3108
style	500	1149	7525	3470
t magazine	497	1045	8254	3709
technology	500	1060	8564	3030
the learning network	500	1340	7287	2084
the upshot	348	833	5640	2113
theater	500	1089	9294	3628
times insider	258	608	4730	2189
today’s paper	406	851	4834	1533
travel	500	1304	9487	3485
u.s.	498	1011	8725	2661
well	500	1150	7053	2369
world	481	854	7296	2887
your money	200	566	3882	1578

Also, you can find bar charts of the top 10 frequencies words per each section in our drive. Below are some of them to find out more visit this link.


times
0
20
40
60
80
100
120
new
art
artist
museum
show
review
work
music
york
time
arts


times
0
50
100
150
200
briefing
ukraine
russia
day
end
need
know
evening
new
war
briefing



As you can see for example in the book section we have a lot of words related to reading or books like novels, books, stories…

Challenges

- One of the most important challenges and difficulties we faced in this phase was in fetching data state. Due to some small political problems and limitations 😊😊 we could not send get requests from our local devices and needed to run our code in the google colab which is provided in the attachment.

- The second challenge we faced was again fetching data. We have some general limitations from nytimes website too. We could not send more than 10 get requests per minute so we solved it by adding some sleep between sending consequences of requests.

Here is the link to our data and other images about data.

Here is the link to our Github project.