NLP Assignment

Submitted to:
SB Web Technologies

Submitted by:
Abhishek Koirala

Table of content

Problem Objective

The requirements of this assignment are listed as below:

1) A page having a form with an input and a submit button.
2) Allow, giving a valid URL to scrap/mining the information from the page.
3) Identify the factual data and list them to proceed. (Consider data authenticity)
4) List out the factual information/data.
5) Request to generate an introductory sentence using the information/data.

The expected outcomes are:

1) Identify the factual information contained in a given scrapy/mineable URL's content.
2) Generate an introductory sentence assuring the high quality of grammar based on the identified factual information.

Methodologies and results

As per the requirements listed above, this problem can be broken down into following steps:

1) Identify the data source/stream for the data
2) Scrap the data stream and extract useful data
3) Explore credibility of data
4) Extract credible data
5) Use the credible data to train and build a text generator model
6) Use the text generator model to generate introductory sentence
7) User Interface

Now let us go through each of these steps.

1) **Identify the data source/stream for the data**
   I explored various news providers(BBC, KathmanduPost, New York Times,etc) as well as various domains(sports, finance, politics) for the news articles. Each of these providers had a very limited amount of data which would possess problems later on during training the generator model. I also started exploring scraping

tools available online and came across https://mediastack.com/. With a trial version api key, it allowed me to scrap news articles based on my preferences(categories, language).

2) **Scrap the data stream and extract useful data**

I decided to scrap technology related news in English language using the following script

```
def getResponse(offset):
    conn = http.client.HTTPConnection('api.mediastack.com')

    params = urllib.parse.urlencode({
        'access_key': '97bd1533357df03f595417b0d962e28c',
        'languages': 'en',
        'categories': 'technology',
        'limit': 100,
        'offset':offset,
        })

    conn.request('GET', '/v1/news?{}'.format(params))
    res = conn.getresponse()
    return res
```

I was able to scrap 5600 news articles, which was near sufficient for me to carry out the task. The extracted articles are saved to a SQLITE database **news_data.db.**

Author	Title	Description	URL	Source	Country	Date
Manish Singh	Facebook rolls out vaccine finder tool in Indi...	Facebook has announced a $10 million grant to ...	https://techcrunch.com/2021/05/03/vaccine-find...	TechCrunch	us	2021-05-03T11:33:48+00:00
Darrell Etherington	Wealthsimple raises $610 Mata 4B valuation	Canadian fintech giant Wealthsimple has raised...	https://techcrunch.com/2021/05/03/wealthsimple...	TechCrunch	us	2021-05-03T11:30:37+00:00
Elizabeth Lopatto	Why Epic is burning its own cash to cook Apple	Illustration by Alex Castro / The Verge Follow...	https://www.theverge.com/2021/5/3/22412899/epi...	The Verge	us	2021-05-03T14:09:30+00:00
Jacob Kastrenakes	G4's newest host is virtual streamer CodeMiko	G4's newest host looks a little different from...	https://www.theverge.com/2021/5/3/22417239/cod...	The Verge	us	2021-05-03T15:20:24+00:00

I was mainly interested in the Description column as it consisted of the news body. The average number of words in the news body was 53 words. There were a total of 9 media sources from which these articles were extracted.

```
df['Source'].value_counts()
```

```
TechCrunch                 3023
The Verge                  1100
Hacker News                 386
Engadget                    353
The Next Web                291
The New York Times          203
The Age                      88
Wired                        87
The Sydney Morning Herald    69
Name: Source, dtype: int64
```

Parameters like language and categories can be tweaked in order to extract data. The available languages are **ar** - Arabic, **de** - German, **en** - English, **es** - Spanish, **fr** - French, **he** - Hebrew, **it** - Italian, **nl** - Dutch, **no** - Norwegian, **pt** - Portugese, **ru** - Russian, **se** - Swedish, **zh** - Chinese.

Also the available categories are:

- **general** - Uncategorized News
- **business** - Business News
- **entertainment** - Entertainment News
- **health** - Health News
- **science** - Science News
- **sports** - Sports News
- **technology** - Technology News

3) **Explore credibility of data**

There are different approaches to exploring credibility of the extracted data

- Manual Inspection
- Source credibility exploration
- Identifying clickbait headlines
- Inspecting difference in title and news body
- News classifier

The above mentioned 4 steps are the most reliable steps for exploring credibility of data because it involves manual inspection. Our computers are yet not advanced to identify advanced contextual information. I went on with the 5th approach to identify credibility of these data. I decided to train my own news classifier and pass these data to the model.

I took a collection of 21417 true news articles(**True.csv**) and 23481 false news articles(**Fake.csv**). The news articles consisted of several domains as shown in the figure below

```
politicsNews       11272
worldnews          10145
News                9050
politics            6841
left-news           4459
Government News     1570
US_News              783
Middle-east          778
Name: subject, dtype: int64
```

Since we were scraping technology related news, a lot of our classifier predictions were expected to turn out false. Again we were only interested in the news body. I removed english stopwords from the news body using the NLTK library. The entire dataset was split to train and validation set with 20% split on validation set. The distribution of true news and false news in both training and validation set were as follows:

```
Training data
Number of true news:  17159
Number of false news:  18759
```

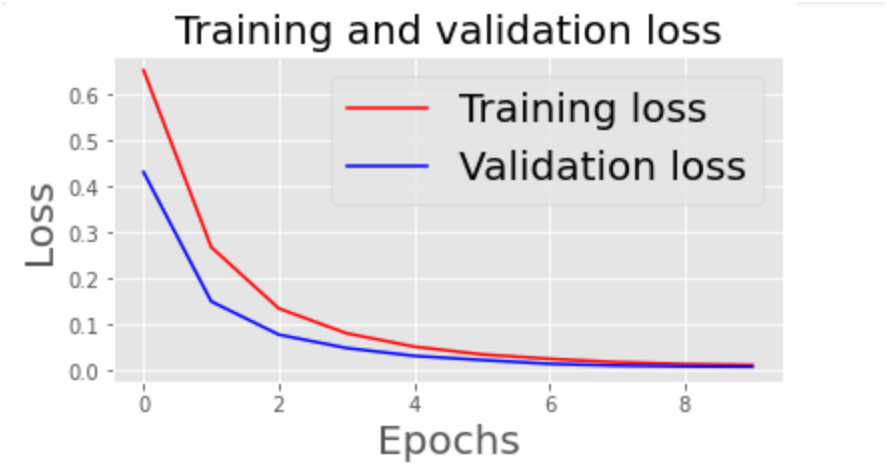
```
Testing data
Number of true news:  4258
Number of false news:  4722
```

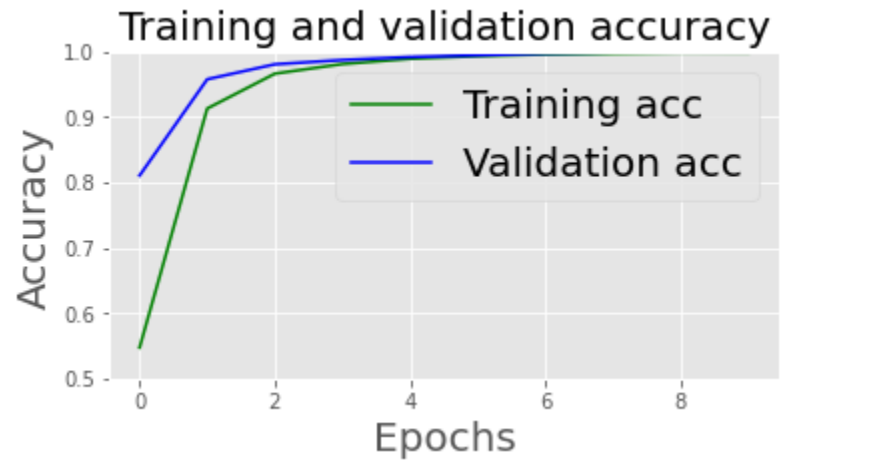
I used tokenizer to tokenize my training and test data and padded the train and test sequence with maximum length 256. The trainable parameters in the model were over 3 million. The complete model configuration for the classifier can be found on **Fake news model.ipynb.**The model summary can be seen below

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 32)          3424000
_________________________________________________________________
bidirectional (Bidirectional (None, None, 128)         49664
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32)                18560
_________________________________________________________________
dense (Dense)                (None, 64)                2112
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 3,494,401
Trainable params: 3,494,401
Non-trainable params: 0
```

I used a bidirectional LSTM model. This is because our prediction not only depends on previous input but also on future input. Bidirectional LSTM provides us with the feature of understanding previous as well as future context. Dropout is used here to prevent overfitting. Sometimes we might have words in our validation set that the training set has never seen and the model has never understood. Such words can lead to overfitting issues. Neurons in a fully connected layer develop co-dependency among each other during training and lead to overfitting of training data. Dropout layer loosens this tight codependency between neurons and helps in regularizing overfitting. Binary cross entropy was used as a loss function as we only had 2 target classes. Adam optimizer was used as optimization function

I only trained the model for 10 epochs. With more than 3 million parameters, the training was slow on my device without GPU. At the end of 10 epoch the training accuracy was 99.81% while validation accuracy was 99.87%. The training loss and validation loss were 0.0123 and 0.085 respectively. The accuracy and loss plot are shown below





Also the classification report is shown below

	precision	recall	f1-score	support
False	0.98	0.99	0.98	4722
True	0.99	0.98	0.98	4258
accuracy			0.98	8980
macro avg	0.98	0.98	0.98	8980
weighted avg	0.98	0.98	0.98	8980

4) Extract credible data

Our extracted news articles were then passed to this classifier model for prediction. Only 586 news articles from 5600 news were classified to be true. This was expected because the domains used in building the classifier model and our data were completely different. The true predicted data were then stored in a different database(**actual_news_data.db)** to be used for language model training and text generation.

5) **Use the credible data to train and build a text generator model**

Using the truth classified data, I trained a language model for text generation. The entire news body was concatenated to a single string. The model was built in such a way that the model took the first 5 words from the whole string and predicted the 6th word. Then it would consider a phrase consisting of 2nd word to 6th word and predict the 7th word and so on. The model was trained for 100 epochs. The number of trainable parameters was around 1.5 million. Here categorical cross entropy is used as a loss function with adam optimizer The model summary can be seen below

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 5, 5)              1310
_________________________________________________________________
lstm_6 (LSTM)                (None, 5, 50)             11200
_________________________________________________________________
lstm_7 (LSTM)                (None, 500)               1102000
_________________________________________________________________
dense_6 (Dense)              (None, 500)               250500
_________________________________________________________________
dense_7 (Dense)              (None, 262)               131262
=================================================================
Total params: 1,496,272
Trainable params: 1,496,272
Non-trainable params: 0
```

The training accuracy of 99% was achieved and training loss was 0.056.

6) **Use the text generator model to generate introductory sentence**

The trained model was then used to generate texts. The generator function is defined in such a way that it needs a seed text to begin generation with and the total number of generated words is currently set to 10. Below are the result of generated text with various input as seed text

```
output=generate_text_sequence("The doctors",10)
print(output)
```

```
the the which owns of the darkside of rivian organization
```

```
output=generate_text_sequence("The US governemnt",20)
print(output)
```

```
restrictions especially the stock market chinese led cited a lack of raise will gather large
st shares firms tracking the iconic
```

```
output=generate_text_sequence("President Trump",20)
print(output)
```

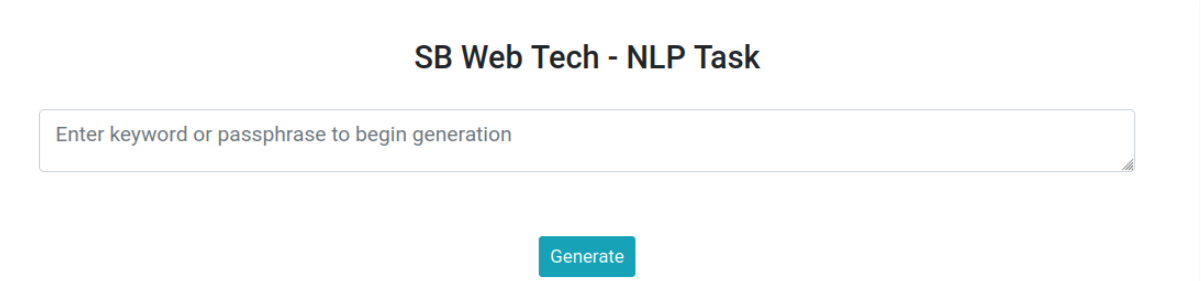
```
enforcement wants wants enforcement of last double a the host trade commission the seed aero
space has an which the of
```

```
output=generate_text_sequence("The government has never been",20)
print(output)
```

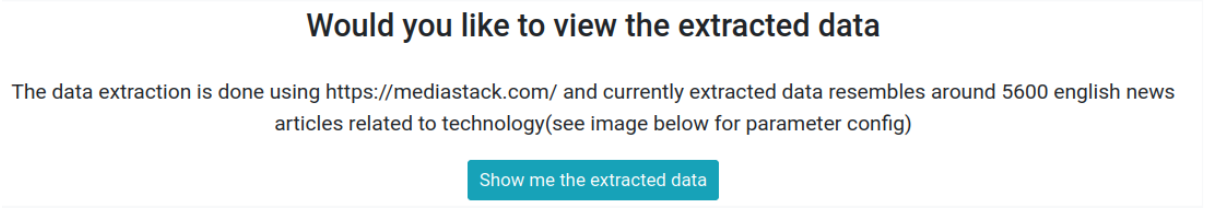
```
the noncompliance with local now storage rules the south a indian than of and angeles agrawa
l back south restrictions market
```

7) **User Interface**

I built a flask server and used a html page for the user interface. The first part of the UI would ask users for seed text for text generation.



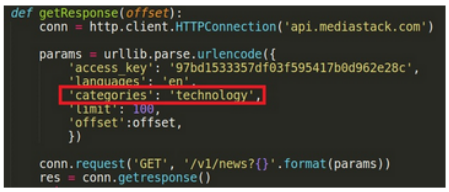
The second part of the UI consists of a button from which one can observe the complete extracted data based on predefined configuration.in json format.



Users also have an option to extract data on their own with few configuration changes in the data extractor file. The api key has limited request capacity so users will have to input their own access key if they want to extract customized data for themselves. The complete instruction is also shown in the UI

You can also extract data on your own. But the limited version on the extraction tool only allows 500 requests per month which might be over soon. You can upgrade your own api key and request results for different categories. Please make sure you make changes to the "categories" parameter in data_extractor.py as shown in image below. The other available parameters are: general, business, entertainment, health, science, sports, technology

The data extraction will take some time. Please check your log for details



Extract data and show me

The last part of the UI consists of the option of viewing the true classified data by the model. User can directly see a table representation of these values or can get them in json format

The extracted datas are passed to a news classifer. The classifier is built with 44,898 labelled news articles. You can find the classifier in Fake news model.ipynb file

Show me the classified data in json

Show me the classified data in table

Limitation of the Solution

Since this project was done in a limited timeframe, it has its limitations.

- The first limitation is the flexibility of scraping news articles. Because we are using a certain tool, we can only extract data provided by the tool and the options provided are also very limited. Hence a better extraction tool can be built for leveraging complete power of data scraping
- The classifier data consisted of news of every other domain except technology. But our news articles all belonged to the technology domain. Thus the application

of our data to the classifier model did not yield a very good result. We can do better by focusing on a single domain and improving the model.

- There are other manual approaches to identification of credibility of news articles through the means of natural language understanding. It's observed that fake news sources don't actually care about a particular format of news writing and also neglect grammatical errors. But a credible news source does so. Analyzing the grammatical structure with NER(Named Entity Recognition) can also provide us with better results.
- Training a language model is a big task and requires a huge amount of data. I have trained a language model with around 500 data which is not sufficient if we want to build something meaningful. Thus the text generation model does not look very convincing. But this can be improved significantly as we increase the data and focus more on the detail of data we are working with.
- No exploratory analysis on the texts has been done here, which is a huge task but always beneficial while building a language model.

References

Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., & Yu, Y. (2018, June). Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 1097-1100).

Danlos, L. (1987, April). The linguistic basis of text generation. In *Proceedings of the third conference on European chapter of the Association for Computational Linguistics* (pp. 1-1).

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., & Weston, J. (2019). Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

https://mediastack.com/

https://flask.palletsprojects.com/en/2.0.x/

Thota, A., Tilak, P., Ahluwalia, S., & Lohia, N. (2018). Fake news detection: a deep learning approach. *SMU Data Science Review*, *1*(3), 10.

Koirala, A. (2020). COVID-19 Fake News Classification using Deep Learning.