Project Description:

- The purpose of this project is to successfully perform named-entity recognition by training the model on the dataset and then making predictions on the test dataset. The code first takes in the dataset and cleans it up by dropping any null values. Each row of the dataset consists of four columns, the sentence number, the actual word, part of speech (POS), and the tag value. For the purposes of the project, the main focus will be on the tag and the word column. In order to convert tags into numeric values, a list of all unique tag values is made. Similarly, a list consisting of all unique word values is also made. The words and tags are then aggregated (word to tag association) in a sentence format. The words and the tags of each of the sentences are then stored into separate arrays. Each sentence is then padded to maximum length. In the next part the tag values in the sentences are then replaced with their numerical values, padded, and then converted to one-hot-vectors. The purpose of the process mentioned above is to get the data ready and in the correct format for the model. Since this project is using ELMo, the input must be in a specific format. In the next part, the data is split into training, validation, and testing. Then the ELMo model is initialized along with the ELMo embedding. ELMo requires all the batches to be of the same length. To satisfy this requirement, the data is then split accordingly. The next step in the code is the network itself. The network takes words per sentence as the input, passes that through the word embedding ELMo (defined earlier). The embedded layer is passed through a Bidirectional LSTM layer, then through relu and softmax activation. The code then uses adam opitimizer with categorical crossentropy as the loss function. Through the different layers the model first trains on the data and then helps predict the tag value for each of the words in the testing data. Accuracy and f1-score were used to measure the performance of the model.

Installation instructions:

1. Please download the code file titled "" from the following link: https://github.com/kaurj7/NLP/tree/main/Information%20Extraction
2. Please download the data file from the following link: https://github.com/kaurj7/NLP/tree/main/Information%20Extraction/Data
3. Sign in to your gmail account.
4. Open google colab through the following link: https://colab.research.google.com
5. Once you have followed the link above, you should see a screen with an upload option on the top right.
6. Click on the "Upload" tab on the top right. You should now see the screen with an option to choose file to upload.
7. Click on "Choose File" and choose downloaded code file.
8. Once the upload is successful, you should be able to now see the file opened in google colab, ready for use.

Usage instructions:

1. Make sure you have successfully downloaded and opened the code file in Google Colab by following the installation steps mentioned above.
2. Once the file is open in Colab, upload the downloaded data file by clicking on the folder icon on the left side of the screen, and then choosing the upload option. When prompted, choose downloaded data file for upload.
3. To be able to follow the code easier, the code file consists of several code blocks, which can be executed individually. In order to run the entire code from start to finish, please start at the beginning of the file and execute each code block in order.
4. Each code block has a "play" button on the left, used to run that particular code block.
5. Please wait for each code block to be successfully executed (denoted by a green check mark, underneath the "play" button), before running the next block of code.
6. Repeat this process until the end of the file.

Method:

- ELMo was chosen as the tool for word embedding, in this project. The embedded output from ELMo was then used as input into the network. The network first consists of the embedded layer, which is passed to the Bidirectional LSTM layer. The Bi-LSTM layer is then passed through the relu dense layer and then finally through the dense softmax layer.

Data:

1. The data chosen for this project was taken from Kaggle: https://www.kaggle.com/datasets/abhinavwalia95/entity-annotated-corpus

2. Data information:
    a. Size: 48k
    b. NER categories(8):
        i. **geo** = Geographical Entity
        ii. **org** = Organization
        iii. **per** = Person
        iv. **gpe** = Geopolitical Entity
        v. **tim** = Time indicator
        vi. **art** = Artifact
        vii. **eve** = Event
        viii. **nat** = Natural Phenomenon

3. Data Split:
    a. Training → 75%

b. Validation → 15%
c. Testing → 10%

Results:

	Precision	Recall	f1-score	support
0	0.22	0.18	0.20	28
1	0.81	0.88	0.84	1689
2	0.96	0.94	0.95	1617
3	0.78	0.79	0.78	1688
4	0.82	0.73	0.77	669
5	0.43	0.29	0.35	34
6	0.87	0.89	0.88	1982
7	0.87	0.88	0.87	3669
8	0.11	0.07	0.09	28
9	0.80	0.78	0.79	679
10	0.86	0.89	0.88	1705
11	0.58	0.41	0.48	17
12	0.76	0.75	0.75	2023
13	0.10	0.13	0.11	23
14	0.40	0.29	0.33	7
15	1.00	0.43	0.60	21
16	1.00	1.00	1.00	594425
accuracy			0.99	610304
macro avg	0.67	0.61	0.63	610304
weigthed avg	0.99	0.99	0.99	610304

*Numbers 0-16 represent each of the unique tags available in the dataset. These number assignments change every time the code is executed.

Discussion:

- As seen by the classification report above, the accuracy of the model was 0.99. This high accuracy is due to the overwhelming O tags present in the dataset. Since majority of the words in each of the sentences belong to the O tag, the model had a lot more O tags to train on. Additionally, it is also expected for the testing dataset to have majority O tags (as per sentence structure). Due to which the model accuracy is high and the model was able to correctly predicted the O tag values. The model also, performed well training and validation data. The model was executed on different epoch sizes, but the overall accuracy remained about the same regardless of the epoch size.

The following are some observations to note before implementing ELMo. Even though, ELMo is quite easy to implement, there are a lot of pre-requisites that need to be fulfilled prior to the implementation stage. Firstly, tensorflow 2 does not support ELMo, so through experimentation it was found that the best version compatibly is tensorflow 1.15 and numpy 1.18.5. Additionally, ELMo requires the batch sizes to be the same and have the same number of data, which means there is some data loss. Lastly, ELMo implementation requires a GPU in order for the process to finish in a timely manner.

Future work:

- In the future, it would be interesting to be able to test and train on completely different datasets and then compare results with this project. This experimentation will allow a deeper understanding of ELMo as well as named-entity recognition. Following a regular sentence structure, an overwhelming amount of O tag values can be expected. Therefore, the next logical step is to test this model with different types of data. Testing the model on a larger dataset, with the right equipment, and comparing the results would be another future implementation.