

BUDT737 - BIG DATA AND ARTIFICIAL INTELLIGENCE

SPRING23 - PROJECT_06

Project - Natural Language Processing with Disaster Tweets

Team Members: Avika Jain

Sakar Phuyal

Xingjian Qin

Weian Shi

Sahil Pathan

Account name(s) and/or the team name on Kaggle.com:

Account names: Avika Jain, Sakar Phuyal07, Xingjian Qin, Sahil Pathan, Weian Shi

Team Name: BUDT737_Project_06

Code example:

Code Used: Introduction to NLP with TensorFlow and spaCy

Link: <https://www.kaggle.com/code/olemagnushiback/introduction-to-nlp-with-tensorflow-and-spacy?scriptVersionId=126979046>

Data Cleaning And Preprocessing :

```
+ Code + Text
Cleaning the text

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from bs4 import BeautifulSoup
import string

nltk.download('stopwords') # Downloading the stopwords corpus from NLTK
nltk.download('wordnet') # Downloading the WordNet corpus from NLTK

# Forming an example of the WordNet Lemmatizer
lemmatizer = WordNetLemmatizer()

# removing HTML tags from text
def remove_html(text):
    soup = BeautifulSoup(text, 'html.parser')
    return soup.get_text()

# Cleaning the text
def clean_text(text):
    text = text.lower() # convert text to lower case
    text = re.sub(r'http[s]?|www[s]?|https[s]?', '', text, flags=re.MULTILINE) # remove URLs
    text = re.sub(r'@|\#|$', '', text) # remove @, #, $
    text = text.translate(str.maketrans('', '', string.punctuation)) # remove punctuation
    text = remove_html(text) # remove any HTML tags
    text = text.encode('ascii', 'ignore').decode('ascii') # remove emojis
    text = [lemmatizer.lemmatize(token) for token in text.split(" ")] # lemmatization
    text = [word for word in text if word not in stopwords.words('')] # remove stopwords
    text = " ".join(text) # join the words back into a string
    return text

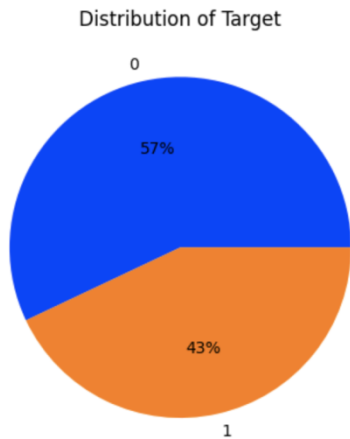
# Cleaning the text in the 'text' column of the 'train' DataFrame
train['text'] = train['text'].apply(clean_text)

# Clean the text in the 'text' column of the 'test' DataFrame
test['text'] = test['text'].apply(clean_text)
```

Exploratory Data Analysis (EDA)

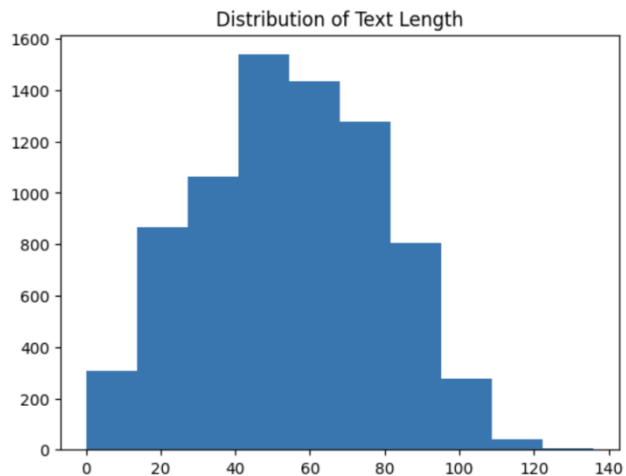
1. Pie Chart of Distribution of Targets

```
# Plotting the distribution of target by pie chart
## https://www.geeksforgeeks.org/how-to-create-a-pie-chart-in-seaborn/
import matplotlib.pyplot as plt
import seaborn
# define Seaborn color palette to use
palette_color = seaborn.color_palette('bright')
plt.pie(count_target.target, labels=count_target.index, colors=palette_color, autopct='%0.0f%%')
plt.title('Distribution of Target')
plt.show()
```



2. Distribution of text length

```
# Plotting the distribution of text length by histogram
plt.hist(train['text length'])
plt.title('Distribution of Text Length')
plt.show()
```



Method 1: Tokenization and Padding (1st Model)

Initialize Tokenizer

```
✓ [4] # Tokenizer Fit on the text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(train['text'])

train_sequences = tokenizer.texts_to_sequences(train['text'])
test_sequences = tokenizer.texts_to_sequences(test['text'])
max_seq_length = max([len(x) for x in train_sequences]) # maximum sequence length among the training sequences
train_sequences = pad_sequences(train_sequences, maxlen=max_seq_length, padding='post') # padding the sequences to have a consistent length
test_sequences = pad_sequences(test_sequences, maxlen=max_seq_length, padding='post')
```

Method 2: Pre- Trained word embeddings (2nd Model)

```
#making a dict mapping words (strings) to their NumPy vector representation and we have used the 300D ones.
#Citation: https://keras.io/examples/nlp/pretrained\_word\_embeddings/
path_to_glove_file = "glove.6B.300d.txt"

embeddings_index = {}
with open(path_to_glove_file) as f:
    for line in f:
        word, coefs = line.split(maxsplit=1)
        coefs = np.fromstring(coefs, "f", sep=" ")
        embeddings_index[word] = coefs

print("Found %s word vectors." % len(embeddings_index))

num_tokens = len(voc) + 2
embedding_dim = 300
hits = 0
misses = 0

# Prepared embedding matrix
embedding_matrix = np.zeros((num_tokens, embedding_dim))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # Words not found in embedding index will be all-zeros.
        # This includes the representation for "padding" and "OOV"
        embedding_matrix[i] = embedding_vector
        hits += 1
    else:
        misses += 1
print("Converted %d words (%d misses)" % (hits, misses))

#loaded the pre-trained word embeddings matrix into an Embedding layer
from tensorflow.keras.layers import Embedding

embedding_layer = Embedding(
    num_tokens,
    embedding_dim,
    embeddings_initializer=keras.initializers.Constant(embedding_matrix),
    trainable=False,
```

Training the Model and Predictions on the test data:

```

#Citation: https://keras.io/examples/nlp/pretrained_word_embeddings/
from tensorflow.keras import layers
seed = 123 #Set the random seed
int_sequences_input = keras.Input(shape=(None,), dtype="int64")
embedded_sequences = embedding_layer(int_sequences_input)
x = layers.Conv1D(128, 5, activation="relu")(embedded_sequences)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(128, 5, activation="relu")(x)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(128, 5, activation="tanh")(x)
x = layers.GlobalMaxPooling1D()(x)
x = layers.Dense(128, activation="tanh")(x)
x = layers.Dropout(0.5)(x)
preds = layers.Dense(1, activation="sigmoid")(x)
model1 = keras.Model(int_sequences_input, preds)

+ Code + Text

[23] #trained the model1
#Citation: https://keras.io/examples/nlp/pretrained_word_embeddings/
x_train = vectorizer(np.array([[s] for s in train["text"]])).numpy()
y_train = np.array(train["target"])

[24] #Citation: https://keras.io/examples/nlp/pretrained_word_embeddings/
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model2.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
history2 = model2.fit(x_train, y_train, batch_size=128, epochs=5, validation_split=0.2)

Epoch 1/5
48/48 [=====] - 9s 26ms/step - loss: 0.6723 - accuracy: 0.5977 - val_loss: 0.6404 - val_accuracy: 0.6855
Epoch 2/5
48/48 [=====] - 1s 12ms/step - loss: 0.5690 - accuracy: 0.7470 - val_loss: 0.4951 - val_accuracy: 0.7768
Epoch 3/5
48/48 [=====] - 1s 13ms/step - loss: 0.4654 - accuracy: 0.7966 - val_loss: 0.4570 - val_accuracy: 0.7978
Epoch 4/5
48/48 [=====] - 1s 12ms/step - loss: 0.4112 - accuracy: 0.8256 - val_loss: 0.4492 - val_accuracy: 0.8017
Epoch 5/5
48/48 [=====] - 1s 12ms/step - loss: 0.3728 - accuracy: 0.8494 - val_loss: 0.4500 - val_accuracy: 0.8024

```

Accuracy

```

[27] print("Train Accuracy: ", history2.history['accuracy'][-1])
     print("Validation Accuracy: ", history2.history['val_accuracy'][-1])

```

```

Train Accuracy:  0.8494253158569336
Validation Accuracy:  0.8023637533187866

```

```

[28] # Find the maximum accuracy achieved
     max_accuracy = max(history2.history['val_accuracy'])
     print("Maximum accuracy is :", max_accuracy)

```

```

Maximum accuracy is : 0.8023637533187866

```

Make predictions

```

[29] # Making predictions on the test set

#x_test = vectorizer(np.array([[s] for s in val_samples])).numpy()
#y_test = np.array(val_labels)

x_test = vectorizer(np.array([[s] for s in test['text']])).numpy()

```

```

[30] test_preds = model2.predict(x_test)
     test_preds = [1 if pred > 0.5 else 0 for pred in test_preds]

102/102 [=====] - 1s 3ms/step

```

```

test_preds

```

```

[1,

```

Methods used:

During the course of our project, we endeavored to explore and implement three distinct machine learning frameworks, namely **Keras**, **PyTorch** and **Random Forest Classifier** in order to enhance the accuracy and optimize the performance of our models.

Below, we provide a description of the three methods employed:

Keras: This framework was utilized as one of our approaches in developing the models. We leveraged the functionalities and abstractions offered by Keras to facilitate the construction and training of our machine-learning models, aiming to achieve improved accuracy and performance.

Our Model Interpretation and Analysis:

Initially, we developed an initial base model for experimentation purposes. We incorporated various enhancements, including modifications to layers, learning rates, batch sizes, epochs, and other relevant parameters, with the aim of maximizing its accuracy. The model creation process involved several steps, such as data and text preprocessing, tokenization, padding, model creation, and training. Upon completion of these steps, we proceeded to generate visualizations for analysis and evaluation.

Subsequently, we introduced global embedding methods into our model, which led to the development of Model B. Notably, this approach yielded a significant improvement in accuracy, achieving a nearly 80% accuracy rate.

GloVe: To process the text into numerical data, we need word embedding, and in our case, we have used a technique called glove embedding. As our methodology, we used pre-trained word embedding from the Keras site and followed the instructions. We initially created a vocab index and vectorized text. Then we loaded pre-trained word embedding, prepared embedding matrices, and loaded those into the layers.

PyTorch: As part of our efforts, we also employed PyTorch, another prominent machine learning library, to explore alternative techniques and methodologies for enhancing our models. By harnessing the capabilities and flexibility provided by PyTorch, we aimed to further optimize the accuracy and performance of our models.

As a part of our experiment and process of improving accuracy, we created a base model for pytorch but the accuracy was not going higher than 0.73.

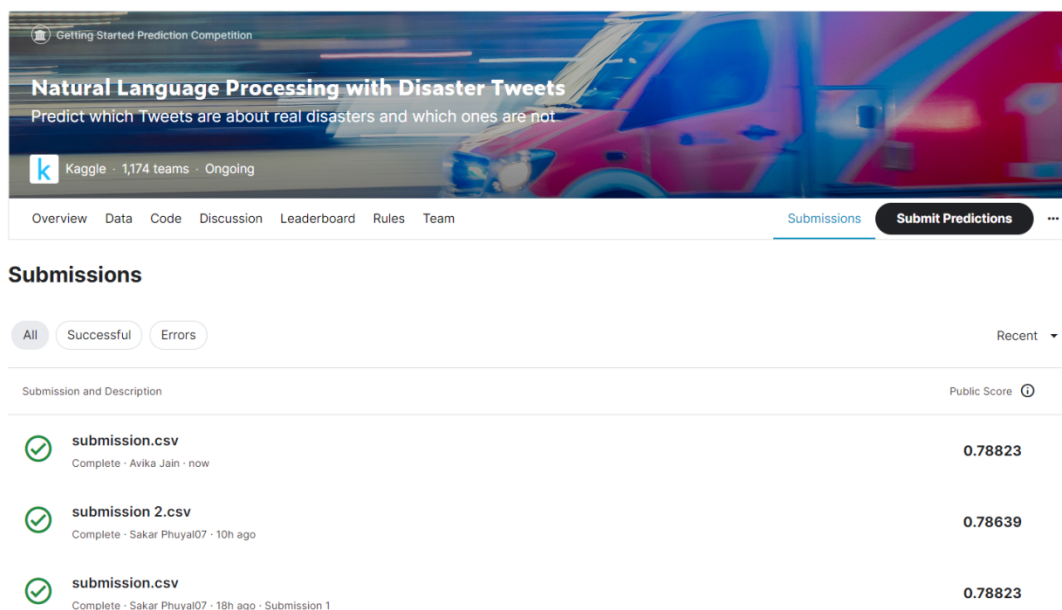
Random Forest:

For the previous two methods, we used Pytorch and Keras, and to further enhance and improve accuracy, we worked on Random Forest Classifier. Random Forest Classifier is an estimator that uses decision trees to improve predictive accuracy. Initially, the model's accuracy was around 74, and I was unable to improve significantly. It was just a slight improvement. So, we decided to stick with the Keras Model.

Following extensive experimental analysis and interpretation, we focused on incorporating global embedding in our Keras model, which resulted in achieving the highest accuracy of 0.8 (refer to google colab file) . Consequently, we have decided to use this model as our final submission.

Score from Kaggle.com:

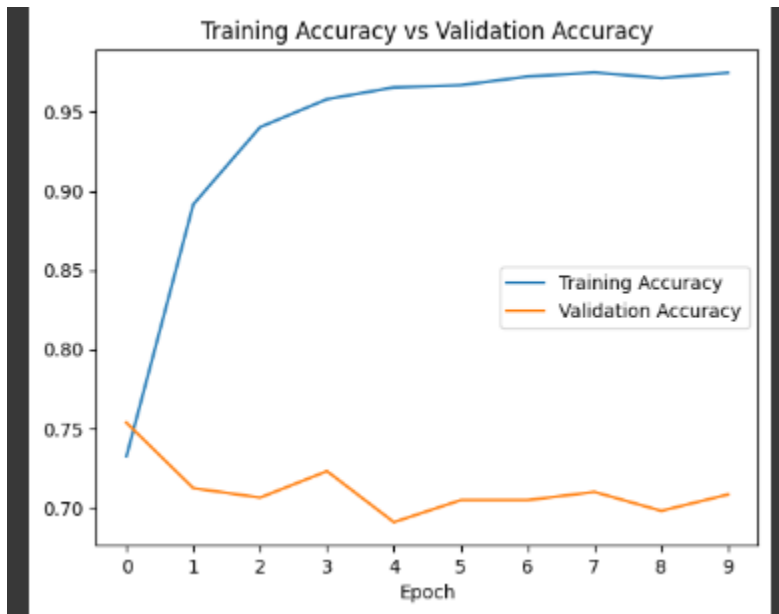
We made three submissions of our code to Kaggle, resulting in accuracies of 78.8% 78.6% and 78.8% respectively.



Submission and Description		Public Score
✓ submission.csv Complete · Avika Jain · now		0.78823
✓ submission 2.csv Complete · Sakar Phuyal07 · 10h ago		0.78639
✓ submission.csv Complete · Sakar Phuyal07 · 18h ago · Submission 1		0.78823

Results and reports:

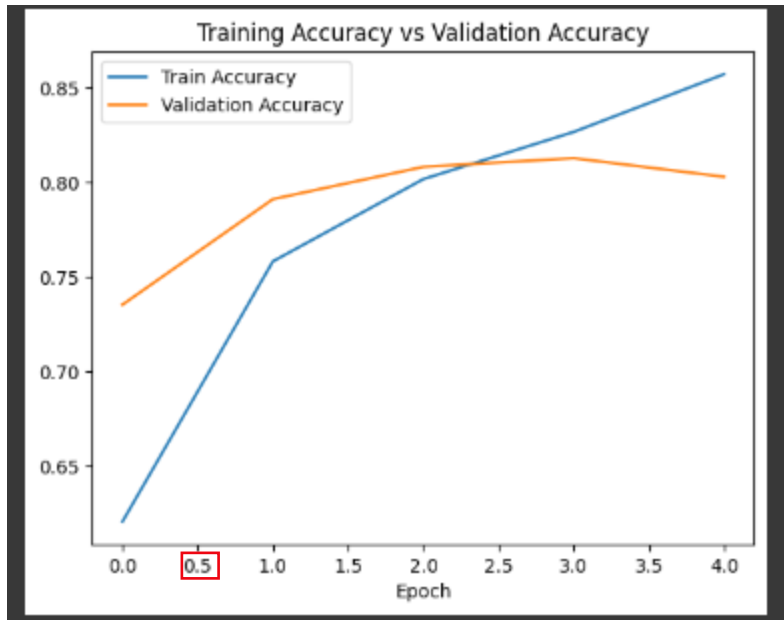
The initial model, also known as the base model, yielded an accuracy of 70.8 percent, showcasing a notable improvement compared to the original model's accuracy of 64 percent. To visualize the performance of the model, we generated a graph plotting the training accuracy against the validation accuracy, resulting in the following outcomes.



The validation accuracy for the initial epochs displayed the highest values; however, as the number of epochs increased, the slope of the validation accuracy graph gradually flattened and eventually declined.

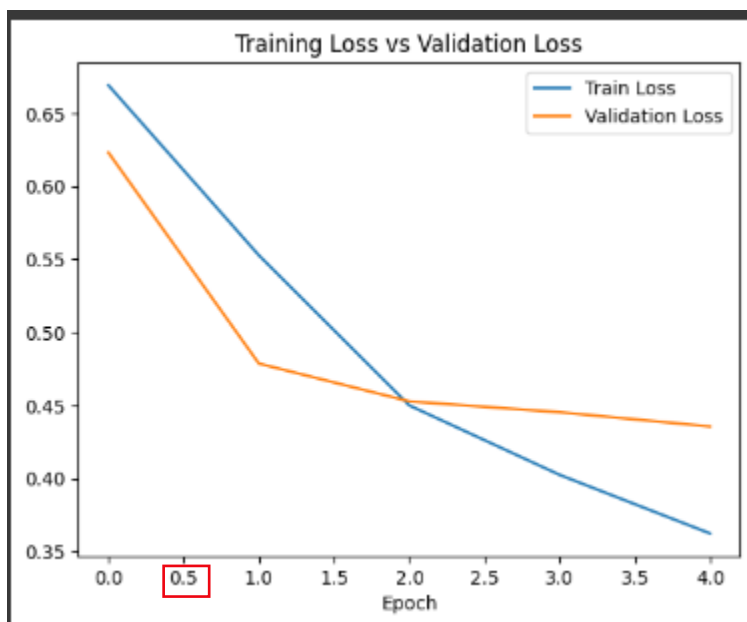
Consequently, we devised an alternative model, referred to as Model 2. By leveraging pre-trained word embeddings, we constructed a Keras model with a train-validation split of 80% and 20% respectively.

Upon executing the model in Google Colab, we achieved a validation accuracy of 80.2%, marking a significant improvement compared to Model 1. This enhancement amounted to a noteworthy 12% increase from the base case, which was deemed highly satisfactory by the entire team. Consequently, we deemed this model as the final selection.



We generated the graph displaying the relationship between Training accuracy and Validation accuracy. From the graph, it is evident that the validation accuracy initially rises and reaches a value of 0.8, after which it levels off, whereas the training accuracy continues to increase.

Additionally, we created another graph illustrating the correlation between Train loss and Validation loss, yielding the subsequent outcomes.



As it is clearly visible, we were able to minimize the validation loss upto the value of 0.43 which is again a decent number.

Contributions of each team member:

Avika Jain - I conducted experiments using both Keras and PyTorch models. In Keras, I explored the impact of adding different layers to observe how they affected the accuracy. Subsequently, I developed a PyTorch model to further experiment and assess if it could potentially improve the accuracy. However, it became apparent that PyTorch was not suitable and efficient for our specific dataset.

Once the models were finalized, I merged the base model with the glove embedding model into a single file. Additionally, I took the initiative to compile a comprehensive document, which included an analysis and interpretation of the models, a description of the methodologies employed, and other relevant details.

Sakar Phuyal - Building random forest classifier models was the first thing I did when I started. Then I moved on to building the pytorch model and still I was unable to get a high level of accuracy. After that, I began to work on the Keras model, which the other group members approved. The accuracy of our original Keras model built was around 0.74.

I began by introducing the idea of glove embedding, then developed a second model in Keras by making use of pre-trained word embeddings, where I first created a vocabulary index, then loaded the word embeddings, then built the model and finally achieved an accuracy in the Google Collaboratory that was more than 0.80.

Xingjian Qin - I took the initiative to construct the initial Keras model as a point of reference and as a foundation for the entire team to build upon. This allow us to explore more ideas and options for initially establishing models. During the coding process, I did a sufficient comments for my teammates to understand the code easily and make improvements on them. I first achieved an accuracy of 0.75 for my model. Based on that, my teammates final enhance the accuracy to 0.78. I believe this progress means a big improvement and reflects that everyone contributes a lot on this project. Lastly, I diligently checked and correctly referenced all sources and citations to ensure the completion of our project, encompassing both the coding and report aspects.

Weian Shi - First of all, it took me some time to contact everyone in the group since I was the initial group leader assigned by the professor. I set up a group discussion room online and

organized a meeting once everyone was in. I suggested splitting into subgroups and working on multiple models to compare their performances. Then after the original Keras model was built, I began to work on improving the accuracy. In addition, I also conducted EDA by exploring and visualizing the distribution of key variables in the training dataset. Finally, I added more comments in the codes to make the codes more readable and make adjustments on the format of the visualization plots. I think everyone has contributed to this project by completing his or her part in time and with high quality.

Sahil Pathan - I first tried to build the pytorch version of the reference model taking the keras model as a reference. The Pytorch model had so much debugging to do and the maximum accuracy I was able to achieve was below par. It became apparent that i had to move on and start working on the keras model which was my initial guess that this method would actually suit our case. As a result, I started working on the current version of the keras model. The accuracy i could achieve after playing around with optimizer, activation function and other parameters was “ “.

After finalising the model that we would eventually go ahead with, I was able to give a brief overview of the project. I made the visualizations and was able to successfully explain them in layman terms so that anyone without a prior knowledge would understand our outcome. I explained in detail the results we got and how we were able to achieve a significant accuracy increasing from “0.7 ” to “0.8” .
I handled the reports and give a detailed explanation about the objectives we set for ourselves and how we were able to achieve them in a satisfactory manner.

Learnings for Each team member:

Avika Jain - Throughout the course of this project, I had the opportunity to develop a wide range of valuable skills, including project management, programming, and analytical skills. Furthermore, this project allowed me to expand my knowledge in previously unfamiliar areas, notably Glove embedding. Through our model, I gained valuable insights and understanding of how to leverage pre-trained models and embeddings like GloVe, which in turn enhanced my understanding of transfer learning techniques.

Through the project, I acquired understanding the architecture, selecting appropriate layers, optimizing hyperparameters, and training the models.
One significant aspect of my learning journey was understanding the impact of different model features on accuracy, such as batch sizes, epochs, and layers. By experimenting with these variables, I was able to observe trends and make informed decisions to optimize the model's performance.

Moreover, working in a team setting was a rewarding experience. The collaborative environment fostered effective communication and allowed us to learn from one another. The spirit of cooperation among team members greatly contributed to the overall success of the project.

Sakar Phuyal - As the project progressed I gained a lot of useful knowledge. The concept of the Natural Learning Process was new to me. Now, I have a better understanding of Python and the Natural Language Toolkit (NLTK), and tokenization, which is basically used for breaking phrases because we utilized tokenization in one of the methods to build our first Keras model. Working with my group enhanced my communication and collaboration skills. Another aspect I focused on during my learning experience was improving the validation accuracy without overfitting. In doing so, I gained insight into the impact of layer counts, epoch counts, and learning rates on validation accuracy.

One initiative I took to improve the accuracy was via GloVe embedding. GloVe, which stands for "Global Vectors for Word Representation," is a well-known method for the generation of word embedding. Previously, I was unaware of embedding and embedding matrices. Working on this project I came away with the deeper understanding of GloVe. In addition, I learned how to create a vocabulary index using text vectorization. I learned that even text-encoded vectors are of different sizes, such as 50, 100, 200, and 300 Dimensional. In the project, we ended up using the 300 dimensions. Overall, the project broadened my skills and knowledge of text preprocessing, building and evaluating models, and their interpretation.

There was equal contribution from the group members. After finalizing the model, everyone worked towards a common goal of error analysis and improvement, which helped to successfully complete the project.

Link: <https://medium.com/analytics-vidhya/word-vectorization-using-glove-76919685ee0b>

Xingjian Qin - Completing this project has been an invaluable learning experience for me. Firstly, I achieved significant growth in acquiring project-specific skills. I notably enhanced my proficiency in constructing models, particularly with Keras, and implementing data preprocessing techniques.

Additionally, I expanded my knowledge by learning and successfully applying new libraries that were instrumental in the project's success. I expanded my knowledge of Natural Language Processing (NLP) by learning how to create text classification models. This project not only enhanced my understanding but also contributed to the development of my collaboration and teamwork skills.

Through working with a diverse group of teammates and gaining practical experience as a part of the project team, I acquired a deeper understanding of the importance of effective cooperation.

Overall, this project has played a crucial role in improving my skills, advancing my understanding of NLP, and enhancing my proficiency in collaborating and working efficiently as part of a team.

Weian Shi - This project is a great opportunity for me to combine what we have learned from class to practice. In this project, we were given a high freedom for what model to use with many code examples online. We tried to work on different models instead of just choosing one so that we could compare their performances. To begin with, I conducted EDA by exploring and visualizing the distribution of key variables in the training dataset. For example, I plotted a pie chart to show the distribution of the target. The two classes were nearly half-half, indicating a balanced dataset. Then I learned from both class activities and online resources to build a better model. Although the accuracy of the model was not improved obviously as we expected after trying different layers in the model, we made an improvement on accuracy successfully by using pre-trained word embeddings, which was nearly 12% higher than the public code example. I was inspired by how team collaboration can motivate our creativity.

To sum up, I not only become more confident in constructing a model independently, but also develop a comprehensive understanding of machine learning knowledge and techniques.

Sahil Pathan -

I was quite excited with the fact that this was an online competition and we got the opportunity to compete with teams all over the world and not only our school.

The project was a challenging one and I tried my best to contribute most to my learnings and also play a valuable role in the team.

First of all, with the knowledge from in class activities and also some online resources, I was able to figure out how to convert the keras model into a pytorch.

Using dataloaders for pytorch taught in class, I was able to convert the keras model into the pytorch. Although I was not able to get a good result from it, I still learned a great deal of commands and modules in pytorch.

Eventually, I had to shift to the keras model. By changing different parameters differently, I was able to learn how to implement different optimizers such as 'Adafactor', 'adamW' and also different loss functions and other parameters which can be used to ultimately produce a better and more efficient model.

Although it was challenging to increase the accuracy, it made me learn about the various parameters that can be used to at least achieve a higher accuracy than before.

The visualization we plotted showed me how the validation accuracy and training accuracy work contradictory to each other and helped me understand the idea that visualization makes everything easy.

In addition to it, The report writing part helped me to assess my writing skills which is again significant in order to be a better analyst. I made a major contribution in writing the report and also interpreting results which is a significant part of the analysis of our model. Part of being an analyst is to effectively showcase your writing skills to make someone understand your result in easy terms and writing the report and interpreting the results helped me to do so. Overall, I am glad to be a part of this outstanding team who worked together towards a common goal and was able to achieve the set objective.

References and Citations-

Keras. Using pre-trained word embeddings.

https://keras.io/examples/nlp/pretrained_word_embeddings/

Kaggle. Introduction to NLP with TensorFlow and spaCy.

<https://www.kaggle.com/code/olemagnushiback/introduction-to-nlp-with-tensorflow-and-spacy?scriptVersionId=126979046>

Chollet, F., et al. (2015). Keras: The Python deep learning library. Retrieved from

<https://keras.io/>

Matthew Mayo. Tokenization and Text Data Preparation with TensorFlow & Keras. Retrieved from: <https://www.kdnuggets.com/2020/03/tensorflow-keras-tokenization-text-data-prep.html>

Abadi, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems.

Retrieved from <https://www.tensorflow.org/>

Bird, S., et al. (2009). Natural Language Processing with Python: Analyzing Text with the

Natural Language Toolkit. Retrieved from <https://www.nltk.org/>

Python Software Foundation. (n.d.). re - Regular expression operations. Retrieved from

<https://docs.python.org/3/library/re.html>

Matplotlib Development Team. (2021). Matplotlib: Visualization with Python. Retrieved from

<https://matplotlib.org/>

BeautifulSoup Documentation. (2021). BeautifulSoup Documentation. Retrieved from

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

Count NaN Values in Pandas DataFrame. Retrieved from
<https://sparkbyexamples.com/pandas/count-nan-values-in-pandas/>

pandas.DataFrame.value_counts. Retrieved from
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value_counts.html#pandas.DataFrame.value_counts

Vinamrayadav. How to Create a Pie Chart in Seaborn? Retrieved from
<https://www.geeksforgeeks.org/how-to-create-a-pie-chart-in-seaborn/>

Kartikaybhutani. Python | Pandas Series.str.len(). Retrieved from
<https://www.geeksforgeeks.org/python-pandas-series-str-len/>

Matplotlib: Visualization with Python. Retrieved from <https://matplotlib.org/>

Skr141. How to Adjust Number of Ticks in Seaborn Plots?. Retrieved from
<https://www.geeksforgeeks.org/how-to-adjust-number-of-ticks-in-seaborn-plots/>

Fchollet. (2020). Using pre-trained word embeddings. Retrieved from
https://keras.io/examples/nlp/pretrained_word_embeddings/