CSCE5290: Natural Processing Language
Project Increment 2
Team: Linh Ha, Thanh Le

Introduction:
Our group's project is to work with a dataset of quora question pairs similarity. We found various models performed on Kaggle. Which generates relatively good results. Yet we believe we can improve the model for a better accuracy rate.

Background:
With 67,000 questions posted per day, an efficient suggesting machine will help users access more appropriate answers in shorter time and avoid Quora's resources from wasting data storage. There are some research papers that used this dataset to explore NLP (natural language processing) in Natural Language Understanding [4], [5].

Model:

Machine Learning models:

In the Machine Learning models route we used: SVM (Support Vector Machine) and LR (Logistic Regression).

In this route, we started extracting text features from the clean dataset using N-Grams models; We extracted using Unigram, Bigram, and Trigram models. After that, we feed the N-Gram models to the SVM and LR.

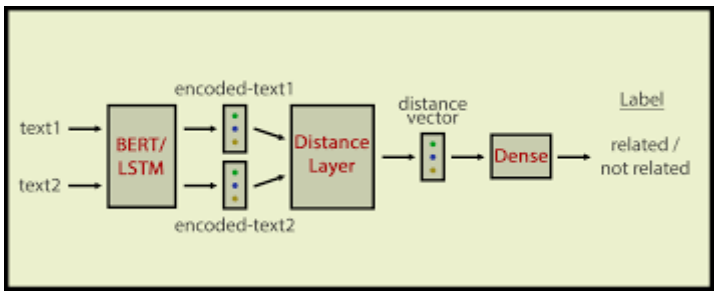
- From what we understand, SVM's goal is to separate the data classes by line (if the dataset is 2d plane) or hyperplane (the hyperplane would be one dimensional lower than the dataset) [7].
- For Logistic Regression, It is trying to find the relationship between features to classify the labels.

Deep Learning models:
Every Deep Learning model in this project has two input layers for each question in a pair.

- LSTM (Vanilla, Bidirectional, and Attention module)
    - Vanilla: In this project, we use 128 units LSTM for each input layer. Then, both outputs of LSTM are fully connected with 64 units (FCN) (fully connected neural network) and 1 neuron for output.
    - Bidirectional: We use 128 Bidirectional LSTM for each input. Then each LSTM output will be connected with 60 units FCN and 2 neurons for text extraction. Then, the output neuron of each Bidirectional LSTM will be multiply together and connect with multiple Fully Connected Neural Network layers (100 units - 50 units - 2 units (output layer))
    - Attention: The model architecture of this LSTM is quite similar to Bidirectional LSTM. We just replace 128 Bidirectional LSTM components

with 128 Vanilla LSTM with the Attention module at the end of the Vanilla LSTM. And the rest of the architecture is exactly the same as Bidirectional Architecture.

- BERT - Siamese Network architecture: [6] [8]



An Illustration of Siamese Network Architecture

  - In this project, we are using BERT - the encoder part (Sister Network) to transform the question pairs into two text features vectors. Then, we calculate similarity distance between the two vectors and feed the result to a single neuron for prediction. We use either Contrastive Loss or Binary Crossentropy loss to train the Siamese Network.

Dataset and Data Analysis

The dataset is collected from Kaggle and includes 6 data fields:

- **id** - the id of a training set question pair
- **qid1, qid2** - unique ids of each question (only available in train.csv)
- **question1, question2** - the full text of each question
- **is_duplicate** - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

The dataset can be downloaded here:
https://www.kaggle.com/competitions/quora-question-pairs

Details design of Features

Preprocessing: The train file of questions is preprocessed by removing non-ASCII characters. Modified output then is written to a new file.

```
In [4]: df.loc[df['question1'].apply(lambda x: x.isascii())==False].head(1)
```

Out[4]:

	Unnamed: 0	id	qid1	qid2	question1	question2	is_duplicate	question1_non_ascii	question2_non_ascii
8	8	8	17	18	When do you use シ instead of し?	When do you use "&" instead of "and"?	0	When do you use instead of ?	When do you use "&" instead of "and"?

Visualizing questions that has non-ASCII characters

In [5]:
```
df['is_duplicate'].value_counts()
```

Out[5]:
```
0    255024
1    149263
Name: is_duplicate, dtype: int64
```

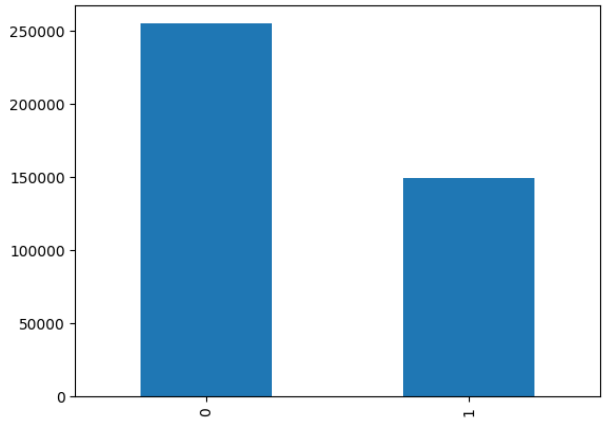
In [6]:
```
df.loc[df['is_duplicate']==1].head(2)
```

Out[6]:

	Unnamed: 0	id	qid1	qid2	question1	question2	is_duplicate	question1_non_ascii	question2_non_ascii
5	5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and c...	I'm a triple Capricorn (Sun, Moon and ascendan...	1	Astrology: I am a Capricorn Sun Cap moon and c...	I'm a triple Capricorn (Sun, Moon and ascendan...
7	7	7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1	How can I be a good geologist?	What should I do to be a great geologist?

In [7]:
```
df['is_duplicate'].value_counts().plot(kind='bar')
print("graph distribution is_duplicate")
```

```
graph distribution is_duplicate
```



Total questions that is_duplicated is 1 in the dataset (As you can see that around 25% of the dataset is duplicated questions)

In [8]:
```
df['question1_non_ascii'].duplicated().value_counts()
```

Out[8]:
```
False    290400
True     113887
Name: question1_non_ascii, dtype: int64
```

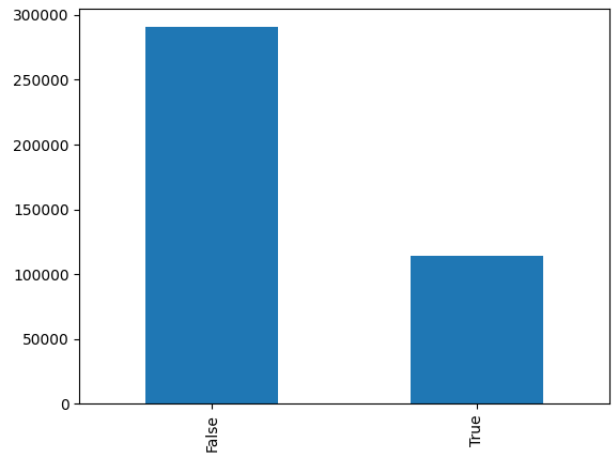
In [9]:
```
df.loc[df['question1_non_ascii'].duplicated()==True].head(2)
```

Out[9]:

	Unnamed: 0	id	qid1	qid2	question1	question2	is_duplicate	question1_non_ascii	question2_non_ascii
264	264	264	81	529	Why do Slavs squat?	Do squats work for men?	0	Why do Slavs squat?	Do squats work for men?
1081	1081	1081	1863	2156	Did Trump win the election?	What do you think of Trump winning the elections?	0	Did Trump win the election?	What do you think of Trump winning the elections?

In [10]:
```
df['question1_non_ascii'].duplicated().value_counts().plot(kind='bar')
```

Out[10]:
```
<AxesSubplot: >
```



Visualizing total duplicated questions (duplicated rows in the dataset)

From the graph, we can see that duplicated rows occur 20% of the whole dataset.

In this phase 1, we are trying to recreate some machine learning models that are created in this research paper [4] to create a baseline so that we can compare them with our proposal model. According to [4], the authors did not eliminate the duplicate rows, they only removed non-ASCII words and performed n-gram models.

Models Analysis:

After preprocessing data from the train.csv file, the design is divided into two sections. The first section is Linear Model Class and the second one is Neural Network Model.

1. Linear Model Class:
    - Question1, question2, and is_duplicated fields are extracted to set up N-gram models (uni-gram, bi-gram, tri-gram).
    - Each set of n-gram then is tuned with SVM (Support Vector Machines). As expected, Tri-gram model give the best result in three features:

SVM unigram feature's accuracy is 0.728 which is close to 0.735 in table 5 [4]

SVM FOR UNIGRAM

```
clf = LinearSVC(random_state=42, tol=1e-3, max_iter=10000)
clf = GridSearchCV(clf, parameters, cv=SSP)
clf.fit(Unigram_train, y_train)
cv_results = clf.best_score_
clf = clf.best_estimator_
print(cv_results)
```

```
0.7285897847735304
```

SVM bigram feature's accuracy is 0.761 which is close to 0.769 in table 5 [4]

SVM FOR BIGRAM

```
clf_2 = LinearSVC(random_state=42, tol=1e-3, max_iter=10000)
clf_2 = GridSearchCV(clf_2, parameters, cv=SSP)
clf_2.fit(Bigram_train, y_train)
cv_results = clf_2.best_score_
clf_2 = clf_2.best_estimator_
print(cv_results)
```

```
0.7609594174965199
```

SVM trigram feature's accuracy is 0.783 which is close to 0.788 in table 5 [4]

SVM FOR TRIGRAM

```
clf_3 = LinearSVC(random_state=42, tol=1e-3, max_iter=10000)
clf_3 = GridSearchCV(clf_3, parameters, cv=SSP)
clf_3.fit(Trigram_train, y_train)
cv_results = clf_3.best_score_
clf_3 = clf_3.best_estimator_
print(cv_results)
```

```
0.7828032979976443
```

- Each set of N-gram also tuned with LR (Logistic Regression), which gave a similar result as above model:

LR unigram set gave result 0.748 which is close to 0.754 in table 1 in LR section [4]

LR FOR UNIGRAM

```
clf_LR = SGDClassifier(penalty='l2',
                    loss = 'log_loss',
                    learning_rate='optimal',
                    max_iter=20,
                    alpha = 0.00001,
                    tol=1e-3,
                    n_jobs = 10)

clf_LR = GridSearchCV(clf_LR, parameter_LR, cv=SSP)
clf_LR.fit(Unigram_train, y_train)
cv_results = clf_LR.best_score_
clf_LR = clf_LR.best_estimator_
print(cv_results)
```

```
/home/thanhle/.virtualenvs/nlp/lib/python3.8/site-packages/sklearn/linear_model/_stochastic_gradient.py:705: ConvergenceWarning: Maximum n
umber of iteration reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
/home/thanhle/.virtualenvs/nlp/lib/python3.8/site-packages/sklearn/linear_model/_stochastic_gradient.py:705: ConvergenceWarning: Maximum n
umber of iteration reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
/home/thanhle/.virtualenvs/nlp/lib/python3.8/site-packages/sklearn/linear_model/_stochastic_gradient.py:705: ConvergenceWarning: Maximum n
umber of iteration reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
0.7478102580576079
```

LR bigram set gave result 0.784 which is close to 0.795 in table 1 in LR section [4]

LR FOR BIGRAM

```
clf_LR_2 = SGDClassifier(penalty='l2',
                    loss = 'log_loss',
                    learning_rate='optimal',
                    max_iter=20,
                    alpha = 0.00001,
                    tol=1e-3,
                    n_jobs = 10)

clf_LR_2 = GridSearchCV(clf_LR_2, parameter_LR, cv=SSP)
clf_LR_2.fit(Bigram_train, y_train)
cv_results = clf_LR_2.best_score_
clf_LR_2 = clf_LR_2.best_estimator_
print(cv_results)
```

```
0.7835689045936395
```

LR trigram set gave result 0.798 which is close to 0.808 in table 1 in LR section [4]

LR FOR TRIGRAM

```
clf_LR_3 = SGDClassifier(penalty='l2',
                    loss = 'log_loss',
                    learning_rate='optimal',
                    max_iter=20,
                    alpha = 0.00001,
                    tol=1e-3,
                    n_jobs = 10)

clf_LR_3 = GridSearchCV(clf_LR_3, parameter_LR, cv=SSP)
clf_LR_3.fit(Trigram_train, y_train)
cv_results = clf_LR_3.best_score_
clf_LR_3 = clf_LR_3.best_estimator_
print(cv_results)
```

```
0.7978209658421672
```

2. Neural Network Model:
    a. Vanilla LSTM:
    - Preprocessed data is tokenized and padded to constant size, whose result below:

In [16]:
```
df['seq1'] = list(sequence1)
df['seq2'] = list(sequence2)
df.head()
```

Out[16]:

	id	qid1	qid2	question1	question2	is_duplicate	question1_non_ascii	question2_non_ascii	seq1	seq2
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	[3, 4, 2, 1224, 58, 1224, 2588, 8, 581, 9, 766...	[3, 4, 2, 1224, 58, 1224, 2588, 8, 581, 9, 766...
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	[3, 4, 2, 557, 11, 1, 1, 6, 1, 4566, 0, 0, 0, ...	[3, 44, 184, 26, 2, 83, 238, 1, 2, 1, 1, 6, 1,...
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	[5, 14, 6, 219, 2, 441, 11, 18, 362, 1832, 202...	[5, 14, 362, 441, 25, 3340, 58, 1349, 222, 1, ...
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	[17, 73, 6, 2779, 314, 2762, 5, 14, 6, 652, 20...	[88, 2, 4174, 38, 231, 2226, 1341, 231, 4, 246...
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	[24, 50, 7124, 9, 232, 1, 1896, 2050, 1, 13, 1...	[24, 1951, 44, 1245, 9, 2050, 232, 0, 0, 0, 0,...

- The model is fitted after embedding LSTM layer and compile Adam optimizer with sparse categorical crossentropy loss

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
embedding_input (InputLayer)    [(None, 30)]         0           []

embedding_1_input (InputLayer)  [(None, 30)]         0           []

embedding (Embedding)           (None, 30, 200)      18369800    ['embedding_input[0][0]']

embedding_1 (Embedding)         (None, 30, 200)      18369800    ['embedding_1_input[0][0]']

lstm (LSTM)                     (None, 30, 128)      168448      ['embedding[0][0]']

lstm_1 (LSTM)                   (None, 30, 128)      168448      ['embedding_1[0][0]']

dropout (Dropout)               (None, 30, 128)      0           ['lstm[0][0]']

dropout_1 (Dropout)             (None, 30, 128)      0           ['lstm_1[0][0]']

dense (Dense)                   (None, 30, 60)       7740        ['dropout[0][0]']

dense_2 (Dense)                 (None, 30, 60)       7740        ['dropout_1[0][0]']

dense_1 (Dense)                 (None, 30, 2)        122         ['dense[0][0]']

dense_3 (Dense)                 (None, 30, 2)        122         ['dense_2[0][0]']

multiply (Multiply)             (None, 30, 2)        0           ['dense_1[0][0]',
                                                                  'dense_3[0][0]']

flatten (Flatten)               (None, 60)           0           ['multiply[0][0]']

dense_4 (Dense)                 (None, 100)          6100        ['flatten[0][0]']

dropout_2 (Dropout)             (None, 100)          0           ['dense_4[0][0]']

dense_5 (Dense)                 (None, 50)           5050        ['dropout_2[0][0]']

dropout_3 (Dropout)             (None, 50)           0           ['dense_5[0][0]']

dense_6 (Dense)                 (None, 2)            102         ['dropout_3[0][0]']

==================================================================================================
Total params: 37,103,472
Trainable params: 37,103,472
Non-trainable params: 0
__________________________________________________________________________________________________
```

- This model gave a result of 0.8458 after 5 epochs.

```
Epoch 1/5
203/203 [==============================] - 24s 75ms/step - loss: 0.5621 - accuracy: 0.7030
Epoch 2/5
203/203 [==============================] - 16s 76ms/step - loss: 0.4755 - accuracy: 0.7692
Epoch 3/5
203/203 [==============================] - 15s 76ms/step - loss: 0.4252 - accuracy: 0.7994
Epoch 4/5
203/203 [==============================] - 16s 77ms/step - loss: 0.3806 - accuracy: 0.8238
Epoch 5/5
203/203 [==============================] - 16s 77ms/step - loss: 0.3389 - accuracy: 0.8458
```

Bidirectional LSTM:

```
Epoch 1/5
203/203 [==============================] - 33s 135ms/step - loss: 0.5675 - accuracy: 0.6963
Epoch 2/5
203/203 [==============================] - 29s 143ms/step - loss: 0.4761 - accuracy: 0.7709
Epoch 3/5
203/203 [==============================] - 29s 142ms/step - loss: 0.4273 - accuracy: 0.7996
Epoch 4/5
203/203 [==============================] - 28s 138ms/step - loss: 0.3808 - accuracy: 0.8247
Epoch 5/5
203/203 [==============================] - 29s 141ms/step - loss: 0.3369 - accuracy: 0.8474
```

Bidirectional LSTM obtained 0.8425 accuracy at Epoch 5.
LSTM with Attention module:

```
Epoch 1/5
203/203 [==============================] - 34s 140ms/step - loss: 0.5656 - accuracy: 0.7062
Epoch 2/5
203/203 [==============================] - 29s 141ms/step - loss: 0.4848 - accuracy: 0.7653
Epoch 3/5
203/203 [==============================] - 28s 140ms/step - loss: 0.4360 - accuracy: 0.7940
Epoch 4/5
203/203 [==============================] - 28s 140ms/step - loss: 0.3891 - accuracy: 0.8190
Epoch 5/5
203/203 [==============================] - 29s 141ms/step - loss: 0.3439 - accuracy: 0.8425
```

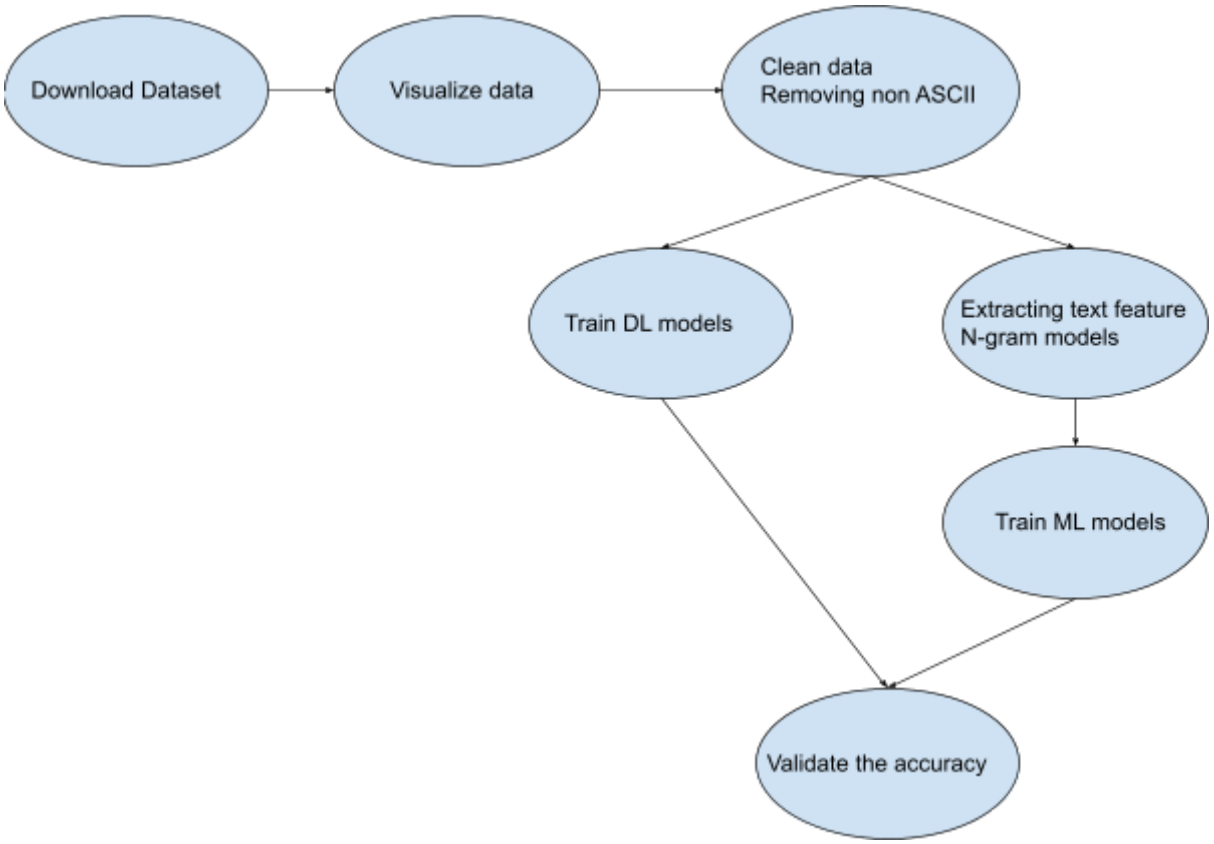
LSTM with Attention module obtained ~ 84% accuracy at Epoch 5, which is somewhat on par with LSTM Bidirectional LSTM. Both models Bidirectional LSTM and LSTM with Attention can improve the accuracy further if we train them with more epochs. The result also improve since increment 1 because we use GloVe for embedding. Which gave us a desired goal within 5 epochs.
BERT Siamese network test accuracy:

```
In [81]:
# GETTING THE ACCURACY (CORRECTED / TOTAL SAMPLES)
np.count_nonzero(corrected[0]) / len(corrected[0])

Out[81]: 0.6307961216278609
```

Compared with traditional N-grams ML and LSTM with multiple variants, BERT Siamese performs the worst; One reason that we can specify is that we only use BERT as text feature extractor without fine tuning the model to fit with the dataset. Another reason could be the number of epochs that we train the model is slightly small. We only trained the model with 2 epochs.

Project workflow (Implementation)



Workflow diagram for the Quora Question Pair Project start from top left and finish at right bottom

Firstly, we downloaded the dataset Quora Question Pair from https://www.kaggle.com/competitions/quora-question-pairs before preprocessing data. Secondly, we started to uploading the dataset to pandas dataframe and start to do data visualization like finding how many duplicating rows (question pairs that appear multiple times in the dataset), what question pairs contain non-ASCII characters, what is the label distribution between duplicated question pair and non-duplicate question pair. After that, we started to do data cleaning by removing non-ASCII characters, reformatted questions after removing non-ASCII, splitted the dataset into train/validation/test sets and saved them in a folder for future use.

After finishing cleaning the data, the dataset was ready for extracting features for Machine Learning (ML) algorithms to learn or feed to Deep Learning (DL) neural networks (LSTM - Bidirectional with or without Attention, Transformer (BERT)).

In the Machine Learning route, we extracted the text feature from the train set using n-Gram models; In this project, we used Unigram, Bigram, Trigrams features. After extracting the n-gram features, we fed them to Machine Learning Algorithms which were SVM (Support Vector Machine) and LR (Logistic Regression) to obtain results.

In the Deep Learning Route, we feed the train text to vanilla LSTM model, LSTM bidirectional model, LSTM with Attention module, and Siamese Network with BERT transformer as text feature extractor.

Validation set was used to validate the models performance in ML and DL.

After ML and DL finished training we tested them with the test set to get the accuracy in overall.

Report:

- Work completed
    - Linh Ha:
        - Researched reference resources
            - Description: Finding research papers, sources that using quora's question pairs dataset for references
            - Responsibility: Finding internet sources that demonstrating how to understand and visualize the dataset along with its features
            - Contribution: 50%
        - Built neural network model (LSTM model)
            - Description: Understanding and creating LSTM model on Quora's dataset
            - Responsibility:
                - Understanding how LSTM work
                - Creating LSTM models on this dataset.
            - Contribution: 100%
            - Limitation: We cannot recreate the suggested model like [4] because the authors did not disclose their architecture like how many LSTM layers that we should have or what are the activation layers or what loss function that we should use, and so on.
        - Build neural network model (Bidirectional LSTM model)
            - Description: Understanding and creating Bidirectional LSTM model on Quora's dataset
            - Responsibility:
                - Understanding how Bidirectional LSTM work
                - Creating Bidirectional LSTM models on this dataset
            - Contribution: 100%
        - Build neural network model (LSTM model with Attention module)
            - Description: Understanding and creating LSTM with Attention module model on Quora's dataset
            - Responsibility:
                - Understanding how LSTM with Attention module work
                - Creating LSTM with Attention module models on this dataset
            - Contribution: 100%

        - Created document
            - Description: Write the document layout
            - Contribution: 100%
        - Kept plan on track
            - Description: Set up meetings and keep track the progress
            - Contribution: 100%
    - Thanh Le:
        - Research reference resources
            - Description: Finding research papers, sources that using quora's question pairs dataset for references
            - Responsibility: Finding and reading research papers about natural language understanding and projects that use machine learning on quora's question pair similarity dataset to create a baseline for comparison.
            - Contribution: 50%
        - Performed preprocessing
            - Description: Doing some text preprocessed techniques
            - Responsibility: Removing non-ASCII characters
            - Contribution: 100%
        - Set up n-gram feature
            - Description & Responsibility: Setting unigram, bigram, and trigram models
            - Contribution: 100%
        - Set up SVM and LR model
            - Description & Responsibility: Create SVM (Support Vector Machine) models and LR (Logistic Regression) on each n-grams
            - Contribution: 100%
        - Applying Pretrain-BERT
            - Description & Responsibility: Understand how BERT works, and how to make BERT extract text features on quora's Question pairs dataset.
            - Contribution: 100%
        - Creating BERT Siamese Network Model
            - Description: Understand what is Siamese architecture
            - Responsibility: How to create a Siamese network that use BERT as a backbone in this dataset
            - Contribution: 100%

References

1. https://www.kaggle.com/competitions/quora-question-pairs/data
2. https://arxiv.org/pdf/1907.01041.pdf
3. https://medium.com/analytics-vidhya/quora-question-pairs-similarity-problem-8e3ae90441f0
4. https://arxiv.org/pdf/1907.01041.pdf
5. https://arxiv.org/pdf/1702.03814v3.pdf
6. https://aclanthology.org/W16-1617.pdf (BERT Siamese was inspired from this research paper)
7. https://www.youtube.com/watch?v=_PwhiWxHK8o&t=679s (For understanding SVM)
8. https://www.google.com/url?sa=i&url=https%3A%2F%2Ftowardsdatascience.com%2Fsiamese-nn-recipes-with-keras-72f6a26deb64&psig=AOvVaw2zZWTFx9zONmATmmOkv0oD&ust=1670131665777000&source=images&cd=vfe&ved=0CBAQjRxqFwoTCLDFjbfb3PsCFQAAAAAdAAAAABAJ Link for the BERT-Siamese network
9. Github: https://github.com/haroldle/NLP_Class_project
10. Video Link in case Canvas video does not have audio: https://myunt-my.sharepoint.com/:v:/g/personal/thanhle5_my_unt_edu/EYvIypqkqOhGi74ImZz2bGoBjlRJo7BVgge65oqxtApLeg?e=xh5k7W https://drive.google.com/file/d/13YqGHyVf75WewhjO_D_XziU1x-FKHW9I/view?usp=sharing