Quora Question Pair Similarities using NBC, LSTM and BERT

Northeastern University – MS in Information Systems

INFO 7610 – Natural Language Engineering Methods and Tools

Akshay Suresh Bhosale (001001091)

Quora Question Pair Similarities

1. NBC
    1. Lemmatization, Preprocessing,
    2. Vectorization
    3. Modeling
2. LSTM-Scratch
    1. Preprocessing
    2. Tokenization
    3. Padding
    4. Neural Network Modeling
3. LSTM using Google News Vectors
    1. Indexing Vectors
    2. Preprocessing
    3. Tokenization
    4. Padding & Embedding
    5. Neural Network Modeling
4. BERT
    1. Preprocessing
    2. Semantic Data Generator
    3. Distribution Strategy Scope
    4. Modeling

Naïve Bayes Classifier

1. Import Dataset
2. Explore Dataset
    1. Drop Null Values
3. Preprocessing the text
    1. Lemmatize
    2. Remove Stop Words
    3. Concat Questions1 and Questions2
4. Vectorizing using TF-IDF
5. Split Dataset into Train-test
6. Model Training
7. Fit model on Test Set

NAÏVE BAYES CLASSIFIER

Lemmatize and Removing Stop Words

```
#preprocessing
def preprocess(series):
  #remove characters other than alphabets & numerics
  words = re.sub("[^A-Za-z0-9]"," ",series).lower().split()

  #lemmatize words
  lemm = WordNetLemmatizer()
  stpwords = stopwords.words('english')
  lemmitized = [lemm.lemmatize(word) for word in words if word not in stpwords]
  sent = ' '.join(lemmitized)
  return sent
```

Combined Question 1 & Question 2

	id	qid1	qid2	question1	question2	is_duplicate	combine
0	0	1	2	step step guide invest share market india	step step guide invest share market	0	step step guide invest share market india step...
1	1	3	4	story kohinoor koh noor diamond	would happen indian government stole kohinoor ...	0	story kohinoor koh noor diamond would happen i...
2	2	5	6	increase speed internet connection using vpn	internet speed increased hacking dns	0	increase speed internet connection using vpn i...
3	3	7	8	mentally lonely solve	find remainder math 23 24 math divided 24 23	0	mentally lonely solve find remainder math 23 2...
4	4	9	10	one dissolve water quikly sugar salt methane c...	fish would survive salt water	0	one dissolve water quikly sugar salt methane c...
5	5	11	12	astrology capricorn sun cap moon cap rising say	triple capricorn sun moon ascendant capricorn say	1	astrology capricorn sun cap moon cap rising sa...
6	6	13	14	buy tiago	keep childern active far phone video game	0	buy tiago keep childern active far phone video...

NAÏVE BAYES CLASSIFIER

Vectorization using
TF-IDF

```
cv = TfidfVectorizer(max_features=50000)#Word to Vectors using Tf-Idf

#Take combine questions data as X
X = cv.fit_transform(train['combine'])
y = np.array(train['is_duplicate'])
print(X.shape)

#Tarin-Test Spilt
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.05)
print(X_train.shape,X_test.shape)
```

```
(404287, 50000)
(384072, 50000) (20215, 50000)
```

Naive Bayes Modeling

```
naive_model = MultinomialNB()#Training
naive_model.fit(X_train,y_train)

#Predictions
y_pred_train = naive_model.predict(X_train)
y_pred_test = naive_model.predict(X_test)
```

Fitting the model on testing set

```
accuracy_train = sum((y_pred_train == y_train).astype(int))/len(y_train)
accuracy_test = sum((y_pred_test == y_test).astype(int))/len(y_test)
print(accuracy_train,accuracy_test)
```

```
0.7519241183944677 0.7406381399950531
```

Accuracy on Test Set = 74.06%

LSTM (Long Short-Term Memory): Scratch

1. Explore dataset
    1. Shape, size and duplicates
    2. Fill missing values using 'fillna()' method
2. Define function for Text Preprocessing:
    1. Remove Punctuations
    2. Remove Stopwords
    3. Use Regular expressions library to sort through misspelled words
    4. Stemming of text
3. Text to Tokens or Tokenization of Text and Replace them
4. Add Padding Sequences using Keras predefined methods
5. Neural Network Modeling
    1. Define LSTM Model
    2. Train Model
    3. Check Validation Accuracy for Optimum Result
6. Predict on Test Set

LSTM (Long Short-Term Memory) [contd.]

- Pre-defined Variables Used:
  - Maximum Text Length = 50
  - Maximum Token Length = Max[max(training tokens), max(testing tokens)] using 'np.max'
  - Batch size for neural network training = 128
  - Epochs = 16 (randomly selected)
  - Train size : Validation size = 82 : 18

LSTM (Model Architecture & Layers) [contd.]

- Sequential Model
- Embedding Layer Dimensions : Max_Token +1000 , 32
- Dropout at 0.3
- LSTM layer with shape None, 32
- Dropout Layer 2 at 0.3
- Dense Layer using Sigmoid activation function
- Loss Function = Binary Cross Entropy
- Optimizer = 'Rmsprop'

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, None, 32)          2930560

 dropout_2 (Dropout)          (None, None, 32)          0

 lstm_1 (LSTM)                (None, 32)                8320

 dropout_3 (Dropout)          (None, 32)                0

 dense_1 (Dense)              (None, 1)                 33

=================================================================
Total params: 2,938,913
Trainable params: 2,938,913
Non-trainable params: 0
_________________________________________________________________
```

LSTM (RESULTS)

[contd.]

```
loss: 0.4742 - val_accuracy: 0.7864
Epoch 11/16
2594/2594 [==============================] - 52s 20ms/step - loss: 0.3656 - accuracy: 0.8368 - val_
loss: 0.4666 - val_accuracy: 0.7830
Epoch 12/16
2594/2594 [==============================] - 51s 20ms/step - loss: 0.3639 - accuracy: 0.8367 - val_
loss: 0.4748 - val_accuracy: 0.7777
Epoch 13/16
2594/2594 [==============================] - 52s 20ms/step - loss: 0.3634 - accuracy: 0.8370 - val_
loss: 0.4621 - val_accuracy: 0.7877
Epoch 14/16
2594/2594 [==============================] - 52s 20ms/step - loss: 0.3607 - accuracy: 0.8389 - val_
loss: 0.4640 - val_accuracy: 0.7873
Epoch 15/16
2594/2594 [==============================] - 51s 20ms/step - loss: 0.3595 - accuracy: 0.8395 - val_
loss: 0.4679 - val_accuracy: 0.7855
Epoch 16/16
2594/2594 [==============================] - 52s 20ms/step - loss: 0.3579 - accuracy: 0.8395 - val_
loss: 0.4681 - val_accuracy: 0.7867
```

```
In [39]: results.head()
```

Out[39]:

	test_id	is_duplicate
0	0	0.221046
1	1	0.327794
2	2	0.859311
3	3	0.069816
4	4	0.367610

Accuracy on Validation Set
78.67%

LSTM USING GOOGLE NEWS VECTORS EMBEDDINGS

1. Predefine Parameters
2. Index Word Vectors
3. Text Preprocessing
    1. Remove Stopwords
    2. Convert text to lower case
    3. Remove Punctuations
    4. Stemming texts
4. Tokenization: Convert text to sequences
5. Pad sequences to determine shape to build Neural Network Tensor
6. Prepare Embedding Matrix
7. Split Training set into Train and Validation Set
8. Define Neural Network
    1. Set up embedding layer
    2. Set up LSTM Layer
    3. Set up Sequential Layers
    4. Merge Layers
    5. Check Whether to Re-weight Classes to fit 17.5% share in test set
    6. Compile and Train the Model
    7. Check Validation Accuracy
9. Predict on Test Set

LSTM USING GOOGLE NEWS VECTORS EMBEDDINGS [contd.]

- Pre-defined Variables Used:
  - Maximum Text Length = 30
  - Embedding Dimension = 300
  - Batch size for neural network training = 2048
  - Epochs = 27 (randomly selected)
  - Random Input to LSTM layer between 175 - 275
  - Random Input to Dense layer between 100 - 150
  - Train size : Validation size = 90 : 10

LSTM using Google News Vectors Embeddings (Model Architecture & Layers) [contd.]

- Sequential Model: 2 Input layers
- Embedding Layer Dimensions : 30, 300
- LSTM: with shape None, 188
- Concatenate Followed by Dropout Layer
- Batch Normalization followed by Dense Layer with shape None, 119
- Second Dropout Layer
- Batch Normalization Layer
- Second Dense Layer
- Loss Function = Binary Cross Entropy
- Optimizer = 'N-Adam'

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 30)]	0	[]
input_2 (InputLayer)	[(None, 30)]	0	[]
embedding (Embedding)	(None, 30, 300)	36150000	['input_1[0][0]', 'input_2[0][0]']
lstm (LSTM)	(None, 188)	367728	['embedding[0][0]', 'embedding[1][0]']
concatenate (Concatenate)	(None, 376)	0	['lstm[0][0]', 'lstm[1][0]']
dropout (Dropout)	(None, 376)	0	['concatenate[0][0]']
batch_normalization (BatchNormalization)	(None, 376)	1504	['dropout[0][0]']
dense (Dense)	(None, 119)	44863	['batch_normalization[0][0]']
dropout_1 (Dropout)	(None, 119)	0	['dense[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 119)	476	['dropout_1[0][0]']
dense_1 (Dense)	(None, 1)	120	['batch_normalization_1[0][0]']

LSTM (RESULTS)

[contd.]

```
356/356 [==============================] - 886s 2s/step - loss: 0.2480 - acc: 0.8120 - val_loss: 0.2705 - val_acc: 0.8041
Epoch 20/27
356/356 [==============================] - ETA: 0s - loss: 0.2447 - acc: 0.8153WARNING:tensorflow:`evaluate()` received a value
for `sample_weight`, but `weighted_metrics` were not provided.  Did you mean to pass metrics to `weighted_metrics` in `compile
()`?  If this is intentional you can pass `weighted_metrics=[]` to `compile()` in order to silence this warning.
356/356 [==============================] - 889s 2s/step - loss: 0.2447 - acc: 0.8153 - val_loss: 0.2654 - val_acc: 0.8055
Epoch 21/27
356/356 [==============================] - ETA: 0s - loss: 0.2421 - acc: 0.8177WARNING:tensorflow:`evaluate()` received a value
for `sample_weight`, but `weighted_metrics` were not provided.  Did you mean to pass metrics to `weighted_metrics` in `compile
()`?  If this is intentional you can pass `weighted_metrics=[]` to `compile()` in order to silence this warning.
356/356 [==============================] - 890s 2s/step - loss: 0.2421 - acc: 0.8177 - val_loss: 0.2661 - val_acc: 0.8105
Epoch 22/27
356/356 [==============================] - ETA: 0s - loss: 0.2394 - acc: 0.8203WARNING:tensorflow:`evaluate()` received a value
for `sample_weight`, but `weighted_metrics` were not provided.  Did you mean to pass metrics to `weighted_metrics` in `compile
()`?  If this is intentional you can pass `weighted_metrics=[]` to `compile()` in order to silence this warning.
356/356 [==============================] - 883s 2s/step - loss: 0.2394 - acc: 0.8203 - val_loss: 0.2675 - val_acc: 0.8099
Epoch 23/27
356/356 [==============================] - ETA: 0s - loss: 0.2372 - acc: 0.8220WARNING:tensorflow:`evaluate()` received a value
for `sample_weight`, but `weighted_metrics` were not provided.  Did you mean to pass metrics to `weighted_metrics` in `compile
()`?  If this is intentional you can pass `weighted_metrics=[]` to `compile()` in order to silence this warning.
356/356 [==============================] - 880s 2s/step - loss: 0.2372 - acc: 0.8220 - val_loss: 0.2714 - val_acc: 0.8161
```

```
In [19]: results.head()
```

Out[19]:

	test_id	is_duplicate
0	0	0.001727
1	1	0.106776
2	2	0.403705
3	3	0.002177
4	4	0.515709

Accuracy on Validation Set : 81.61%

BERT – Transfer Learning using Transformers Tokenizers

1. Predefine Parameters
2. Index Word Vectors
3. Drop Null Values
4. Split Dataset into Train, Test and Validation Set
5. Check Target Distribution
6. Define Preprocessing and BERT Semantic Data Generator Function
    1. Load BERT Tokenizer (bert-base-uncased)
    2. Define number of batches per epoch
    3. Encode both Questions together separated by SEP token
    4. Convert encoded feature to numpy array in batches
    5. Shuffle Indexes
7. Create Model under distribution strategy scope
8. Load Pretrained BERT Model
    1. Set up Encoded token ids from BERT tokenizer
    2. Set up Attention masks
    3. Set up Token Type ids
    4. Loading pretrained BERT model
    5. Freeze the BERT model to reuse the pretrained features without modifying them.
    6. Add trainable layers
    7. Compile & Train Model
9. Evaluate on Test Set

BERT – TRANSFER LEARNING USING TRANSFORMERS TOKENIZERS [contd.]

- Pre-defined Variables Used:
  - Maximum Text Length = 50
  - Embedding Dimension = 300
  - Batch size for neural network training = 168
  - Epochs = 2 (selected on the basis of computing resource and time)
  - Train size : Test size : Validation size = 70 : 15 : 15

BERT – Transfer Learning using Transformers Tokenizers [contd.]

- Input layer: None, 50
- Attention Mask : None, 50
- Token Type Ids: None, 50
- TF_BERT_Model: None, 50, 768
- Idirectional LSTM: None, 50, 256
- Global Average Pooling: None, 256
- Global Max Pooling: None, 256
- Concatenate: None, 512
- Dropout: None, 512
- Dense: None, 2
- Loss Function = Categorical Cross Entropy
- Optimizer = 'Adam'

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 50)]	0	[]
attention_masks (InputLayer)	[(None, 50)]	0	[]
token_type_ids (InputLayer)	[(None, 50)]	0	[]
tf_bert_model (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 50, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	109482240	['input_ids[0][0]', 'attention_masks[0][0]', 'token_type_ids[0][0]']
bidirectional (Bidirectional)	(None, 50, 256)	918528	['tf_bert_model[0][0]']
global_average_pooling1d (GlobalAveragePooling1D)	(None, 256)	0	['bidirectional[0][0]']
global_max_pooling1d (GlobalMaxPooling1D)	(None, 256)	0	['bidirectional[0][0]']
concatenate (Concatenate)	(None, 512)	0	['global_average_pooling1d[0][0]', 'global_max_pooling1d[0][0]']
dropout_37 (Dropout)	(None, 512)	0	['concatenate[0][0]']
dense (Dense)	(None, 2)	1026	['dropout_37[0][0]']

Total params: 110,401,794
Trainable params: 110,401,794
Non-trainable params: 0

BERT (RESULTS)

[contd.]

```
17390/17684 [============================>.] - ETA: 50s - loss: 0.2132 - accuracy: 0.9115

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequen
truncation strategy. So the returned list will always be empty even if some tokens have be

17392/17684 [============================>.] - ETA: 49s - loss: 0.2132 - accuracy: 0.9115

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequen
truncation strategy. So the returned list will always be empty even if some tokens have be

17393/17684 [============================>.] - ETA: 49s - loss: 0.2132 - accuracy: 0.9115

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequen
truncation strategy. So the returned list will always be empty even if some tokens have be

17394/17684 [============================>.] - ETA: 49s - loss: 0.2132 - accuracy: 0.9115

17684/17684 [==============================] - 3268s 185ms/step - loss: 0.2133 - accuracy: 0.9115 - val_loss: 0.2426 - val_a
ccuracy: 0.8996
```

Accuracy on validation Set : 89.91%

```
3765/3785 [============================>.] - ETA: 1s - loss: 0.2427 - accuracy: 0.8991

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee

3768/3785 [============================>.] - ETA: 1s - loss: 0.2427 - accuracy: 0.8991

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee

3769/3785 [============================>.] - ETA: 1s - loss: 0.2426 - accuracy: 0.8992

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee

3770/3785 [============================>.] - ETA: 1s - loss: 0.2426 - accuracy: 0.8992

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee

3771/3785 [============================>.] - ETA: 0s - loss: 0.2426 - accuracy: 0.8992

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee

3772/3785 [============================>.] - ETA: 0s - loss: 0.2426 - accuracy: 0.8992

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee

3773/3785 [============================>.] - ETA: 0s - loss: 0.2426 - accuracy: 0.8992

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee

3774/3785 [============================>.] - ETA: 0s - loss: 0.2426 - accuracy: 0.8992

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequ
ncation strategy. So the returned list will always be empty even if some tokens have bee

3775/3785 [============================>.] - ETA: 0s - loss: 0.2426 - accuracy: 0.8992
3785/3785 [==============================] - 261s 69ms/step - loss: 0.2426 - accuracy: 0.8991

[0.24255122244358063, 0.8991413712501526]
```

REFERENCES:

1. https://keras.io/examples/nlp/semantic_similarity_with_bert/
2. https://www.kaggle.com/competitions/quora-question-pairs
3. https://www.kaggle.com/datasets/leadbest/googlenewsvectorsnegative300
4. https://huggingface.co/datasets?task_ids=task_ids:semantic-similarity-classification
5. https://huggingface.co/bert-base-uncased
6. https://huggingface.co/tftransformers/bert-base-uncased

THANK YOU SO MUCH