EAI 6010 – Application of Artificial Intelligence

Assignment 3: Text Classification with Transfer Learning

Submitted to: Prof. Munthali, Richard

Submitted by: Shamim Sherafati

NUID: 002742363

Date: Nov 29th, 2023

College of Professional Studies Northeastern University – Vancouver

Introduction

The report focuses on text classification using transfer learning, aiming to address the assignment requirements. It leverages the AWD_LSTM pre-trained language model in the fastai framework to build a deep learning-based model. Additionally, a traditional NLP model, namely Logistic Regression and Naive Bayes using TF-IDF, is employed to compare the performance against the deep learning model.

Part A:

First, I started with installing necessary libraries like fastbook and setting up the book for future usage. Then I Installed the Kaggle library and download my API and the Kaggle Jason file from my Kaggle account and upload it to the colab.

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  kaggle.json
• kaggle.json(application/json) - 71 bytes, last modified: n/a - 100% done
Saving kaggle.json to kaggle.json
```

Then, as the dataset is large, I copied its API and moved the Kaggle API key to a specific directory and then uses the Kaggle API to download a dataset related to the "Quora Insincere Questions Classification" competition into the Colab environment. After that I unzipped the file

```
# Download the dataset from Kaggle using the API command
!kaggle competitions download -c quora-insincere-questions-classification
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading quora-insincere-questions-classification.zip to /content
... resuming from 746586112 bytes (5729759046 bytes left) ...
100% 6.02G/6.03G [01:13<00:00, 34.8MB/s]
100% 6.03G/6.03G [01:13<00:00, 77.8MB/s]
```

```
import os

# Get the current working directory
current_directory = os.getcwd()

# List the contents of the current directory
directory_contents = os.listdir(current_directory)
print(directory_contents)
```

```
['.config', 'quora-insincere-questions-classification.zip', 'sample_data']
```

```
!unzip quora-insincere-questions-classification.zip -d data
```

```
Archive:  quora-insincere-questions-classification.zip
  inflating: data/embeddings.zip
  inflating: data/sample_submission.csv
  inflating: data/test.csv
  inflating: data/train.csv
```

Then, it utilized the glob module to search for files with a .csv extension within the specified directory (/content/data). Once found, it retrieves a list of CSV file paths. The output shows the discovered CSV files: test.csv, train.csv, and sample_submission.csv within the specified directory. This process enables easy identification and listing of CSV files for further processing or analysis.

```
# Specifying the Test File Path
test_file_path = '/content/data/test.csv'  # Replace with the correct path to your test file

# Opening and Reading the Test File
with open(test_file_path, 'r') as file:
    test_content = file.read()

# Displaying the Start of the Test Data (first 200 characters)
print("Start of the Test:")
print(test_content[:200])  # Adjust the number to display more or fewer characters as needed
```

```
Start of the Test:
qid,question_text
0000163e3ea7c7a74cd7,Why do so many women become so rude and arrogant when they get just a little bit of wealth and power?
00002bd4fb5d505b9161,When should I apply for RV college of
```

The code above shows that, after opening the file, it captures the text content and displays the initial 200 characters of the file's content using the print function. This process offers an initial insight into the test data, showcasing the structure and initial content, which is crucial for preprocessing and understanding the dataset before conducting further analysis or modeling tasks.

```
train_df=pd.read_csv('/content/data/train.csv')
test_df=pd.read_csv('/content/data/test.csv')
```

```
[13] #Splitting the Training Data into Training and Validation Sets
import sklearn
trn_df,val_df = sklearn.model_selection.train_test_split(train_df, test_size=0.1)
```

```
[14] #Extracting Texts and Labels for Training and Validation

trn_texts = trn_df['question_text']
val_texts = val_df['question_text']

trn_labels = trn_df['target']
val_labels = val_df['target']
```

```
[15] # Counting the Class Distribution in Training Labels
trn_labels.value_counts()
```

```
0    1102564
1      72945
Name: target, dtype: int64
```

In the code above, loads two CSV files, train.csv and test.csv, into data frames (train_df and test_df). Subsequently, it splits the training data into training and validation sets using the train_test_split function, where 10% of the data is allocated for validation. The texts and labels for both training and validation sets are extracted, capturing the 'question_text' column as texts and the 'target' column as labels. Lastly, it counts the distribution of classes within the training labels, indicating that class '0' has significantly more instances (1,102,564) compared to class '1' (72,945). These steps are fundamental for data preparation and exploration, ensuring a proper understanding of the dataset's structure, distribution, and separation for training and validation purposes.

After that, in the code below, I have initialized the column names as 'labels' and 'text' for the DataFrame, sets two constants, 'BOS' (beginning-of-sentence tag) and 'FLD' (data field tag), and proceeds to create two DataFrames, df_trn and df_val, from the training and validation texts and labels, respectively.

The df_trn DataFrame is generated from the 'trn_texts' and 'trn_labels', while df_val is created from 'val_texts' and 'val_labels'. Both DataFrames have columns named 'text' and 'labels'. The 'text' column stores the textual data (question_text), and the 'labels' column contains the corresponding target labels (0 or 1).

```
col_names = ['labels', 'text']

BOS = 'bos'  # beginning-of-sentence tag
FLD = 'field'  # data field tag
```

```
[17] df_trn = pd.DataFrame({'text':trn_texts, 'labels':trn_labels}, columns=col_names)
     df_val = pd.DataFrame({'text':val_texts, 'labels':val_labels}, columns=col_names)
```

```
print (train_df.head)
```

```
<bound method NDFrame.head of                         qid  \
0        00002165364db923c7e6
1        000032939017120e6e44
2        0000412ca6e4628ce2cf
3        000042bf85aa498cd78e
4        0000455dfa3e01eae3af
...                       ...
1306117  ffffcc4e2331aaf1e41e
1306118  ffffd431801e5a2f4861
1306119  ffffd48fb36b63db010c
1306120  ffffec519fa37cf60c78
1306121  ffffed09fedb5088744a

                                                                      question_text  \
0                         How did Quebec nationalists see their province as a nation in the 1960s?
1                    Do you have an adopted dog, how would you encourage people to adopt and not shop?
2                         Why does velocity affect time? Does velocity affect space geometry?
3                                    How did Otto von Guericke used the Magdeburg hemispheres?
4                    Can I convert montra helicon D to a mountain bike by just changing the tyres?
...                                                                                     ...
1306117  What other technical skills do you need as a computer science undergrad other than c and c++?
1306118    Does MS in ECE have good job prospects in USA or like India there are more IT jobs present?
1306119                                                                 Is foam insulation toxic?
1306120                          How can one start a research project based on biochemistry at UG level?
1306121                                            Who wins in a battle between a Wolverine and a Puma?

         target
0             0
1             0
2             0
3             0
4             0
...         ...
1306117       0
1306118       0
1306119       0
1306120       0
1306121       0

[1306122 rows x 3 columns]>
```

Part B:

First I have started with the deep learning model. The code below, demonstrates the process of training a language model using transfer learning in the context of natural language processing.

```
from fastai.text.all import *

# Assuming 'df' is your DataFrame containing text data
# Replace 'question_text' with the actual name of your text column

# Step 1: Prepare the Data
data_lm = TextDataLoaders.from_df(train_df, text_col='question_text', is_lm=True, valid_pct=0.1)

# Step 2: Load the Pre-trained Language Model
learn = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.3)

# Step 3: Fine-tune the Language Model
learn.fine_tune(1, 1e-2)
```

The output represents the training progress across two epochs. The metrics observed are the training loss and validation loss.

100.00% [105070592/105067061 00:02<00:00]

epoch	train_loss	valid_loss	time
0	3.668260	3.539215	46:24

epoch	train_loss	valid_loss	time
0	3.304371	3.276904	51:05

For the initial epoch, the training loss starts at 3.668260, while the validation loss is 3.539215. Subsequently, after the completion of the first epoch, the training loss decreases to 3.304371, and the validation loss reduces to 3.276904. These loss values are used as indicators of how well the model is learning during training. The aim of training is to minimize these loss values, as lower losses typically indicate better performance and improved predictive capability of the model.

obtain accuracy metrics from a deep learning model trained for natural language processing using the FastAI library. However, upon executing the learn.validate() function to retrieve the accuracy, an IndexError occurred after a significant runtime, approximately 1 hour and 45 minutes.

```
test_results = learn.validate()

# Extract the accuracy from the test results
accuracy_value = test_results[1]
print("Accuracy:", accuracy_value)
```

```
1 frames
/usr/local/lib/python3.10/dist-packages/fastcore/foundation.py in _get(self, i)
    114
    115     def _get(self, i):
--> 116         if is_indexer(i) or isinstance(i,slice): return getattr(self.items,'iloc',self.items)[i]
    117         i = mask2idxs(i)
    118         return (self.items.iloc[list(i)] if hasattr(self.items,'iloc')

IndexError: list index out of range
```

This error is typically caused by the test_results list not having the expected structure or length, resulting in an attempt to access an index that doesn't exist. The specific reason for this error can be due to various factors such as incorrect data format, inappropriate function usage, or an issue during model training.

For solving it and using the additional option, first I get its length and then printed the test result:

```
test_results = learn.validate()
print(len(test_results))
```

```
1
```

```
print(test_results)
```

```
[5.319873809814453]
```

After obtaining the accuracy, which was erroneously calculated to be more than 1, it was necessary to normalize it within the range of 0 to 1. Upon normalization, the accuracy value was found to be 0.18, indicating a lower-than-expected performance or a normalization calculation issue.

```
accuracy_value = test_results[0]
print("Accuracy:", accuracy_value)
```

```
Accuracy: 5.319873809814453
```

```
normalized_accuracy = min(accuracy_value, 1.0) / max(accuracy_value, 1.0)
print("Normalized Accuracy:", normalized_accuracy)
```

```
Normalized Accuracy: 0.18797438355683066
```

Part C:

Here, I tried to train the model based on traditional NLP models and get their accuracy. First I used the TF-IDF vectorization technique along with a Logistic Regression model to perform text classification. It first prepares the data by splitting it into training and testing sets, vectorizes the text using TF-IDF, trains the Logistic Regression model, and then evaluates the model's performance on the test set. The reported accuracy achieved by this model is 95.42%.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Step 1: Prepare the data
X = train_df['question_text']
y = train_df['target']

# Step 2: Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Vectorize text using TF-IDF
tfidf = TfidfVectorizer()
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# Step 4: Train a Logistic Regression model
clf = LogisticRegression(max_iter=1000)
clf.fit(X_train_tfidf, y_train)

# Step 5: Evaluate the model
predictions = clf.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.9542080581873863
```

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

X = train_df['question_text']
y = train_df['target']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
# TF-IDF Vectorization
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Initialize Naive Bayes Classifier
nb_classifier = MultinomialNB()

# Train the Naive Bayes model
nb_classifier.fit(X_train_tfidf, y_train)

# Make predictions on the test set
predictions = nb_classifier.predict(X_test_tfidf)

# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy of Naive Bayes with TF-IDF: {accuracy}")
```

```
Accuracy of Naive Bayes with TF-IDF: 0.9420461288161547
```

Then, I tried another traditional NLP model; It performs text classification using the Naive Bayes classifier with TF-IDF vectorization. It splits the data into training and testing sets, vectorizes the text using TF-IDF, trains the Multinomial Naive Bayes model, predicts the test set, and finally computes and reports the accuracy achieved by the Naive Bayes classifier, which is approximately 94.20%.

Part D: Questions

1. **How did you pick your dataset, and what type of business problems would it be used to solve?**

The Quora Insincere Questions Classification dataset was selected for its pertinence in addressing the crucial task of identifying insincere questions on Quora. This dataset aligns perfectly with the assignment's objective of text classification using transfer learning methods. By discerning and filtering out insincere inquiries, it contributes significantly to enhancing the platform's content quality and credibility, ensuring users access authentic and relevant content.

2. **Were there any constraints you needed to add to the dataset to allow you to complete the assignment?**

Several constraints and preprocessing steps were necessary to prepare the dataset for modeling. This included cleaning the text, handling missing values, splitting the data for training and validation, and converting the text into a format suitable for both deep learning and traditional NLP models.

3. **How does the accuracy of the deep learning model compare to the traditional NLP model?**

The deep learning model, specifically the AWD_LSTM architecture, demonstrated a normalized accuracy of 0.18, indicating a relatively lower performance compared to traditional NLP models. This outcome highlights the challenges or potential limitations faced by the deep learning approach in this scenario. Conversely, the traditional NLP models such as Logistic Regression and Naive Bayes yielded higher accuracies of 0.9542 and 0.9420, respectively. This contrast underscores the competitive advantage of the conventional NLP approaches in achieving higher accuracy rates compared to the AWD_LSTM model in the given task.

4. **How does the development effort of the deep learning model compare to the traditional NLP model?**

The deep learning model (AWD_LSTM) required more development effort due to its complex architecture, longer training times, and the need for extensive hyperparameter tuning. In contrast, traditional NLP models like Logistic Regression and Naive Bayes are simpler, quicker to train, and demand fewer computational resources, making them more accessible and less resource-intensive during development.

5. **Which model would you suggest if you had to productize this solution and why?**

For deployment, the traditional NLP models, such as Logistic Regression or Naive Bayes, are recommended due to their simplicity, faster training, and lower computational requirements. These models offer a good balance between accuracy and efficiency, making them more practical for

real-world applications compared to complex deep learning models like AWD_LSTM, which might be computationally intensive and harder to deploy.
While deep learning models might offer higher accuracy with extensive data and computational power, they might not be the most practical choice for deployment due to their complexity, resource intensiveness, and potential challenges in maintaining and interpreting the model in a production setting. Therefore, for practical deployment considerations, a traditional NLP model like Logistic Regression or Naive Bayes might be a more suitable choice.

Reference:

1. Quora insincere Questions Classification | Kaggle.

   (n.d.). https://www.kaggle.com/competitions/quora-insincere-questions-classification