CSCE 5290: Natural Language Processing Project Increment 1

Project Description:

Project Name: Fight Online Abuse Using Natural Language Processing

Team Members: Aimen Chaudhry and Kalyan Bandaru

Goals and Objectives: The project goal was to identify the type of comment toxicity during online interaction. Given the rise in trolling and hatred on social media and elsewhere these days, this is a very real problem. The task is set up as a standard text classification problem in which one have to predict the probability of a comment being toxic or a threat, and submissions are evaluated. The project will be finished by the team to allow for efficient brainstorming on how we can enhance the model accuracy and how we can lend the NLP to be competent to combat online abuse that can have a damaging impact. Identifying toxicity is a lot more than just detecting abusive words in the text as it requires the project to be implemented in different steps and handled appropriately.

For this project, in the first team meeting, we decided a layout for the project implementation, where it was decided that the four main sections of the project include data loading, data preprocessing, train and testing, and model evaluations. It was decided that Aimen Chaudhry will work on the first two sections and Kalyan Bandaru will complete the last two sections. The dataset used for this project is called "Toxic_Comment" and was taken from Kaggle. This dataset includes a significant number of Wikipedia comments, labeled for toxic behavior. The types of toxicity are:

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

Data Loading (Completed by Aimen Chaudhry)

In this section, we uploaded the Toxic_Comment.zip and unzipped the folder to be able to use the files it contained. A description of some of the files used and included are provided below:

- train.csv - the training set with comments with their binary labels
- test.csv - the test set

Random samples of data were printed and plotted in this section to get a deeper understanding of the data we would be working with.

Preprocessing (Completed by Aimen Chaudhry)

In the preprocessing section, we removed unnecessary elements like stopwords, URLs and HTMLs, newline characters, numeric data, punctuation marks from the data, as shown in Figure 1. The data was also tokenized in this section before passing it along to the training and testing section.

```
#removes new line character
train['preprocess'] = train.apply(lambda row: row['comment_text'].replace("\n"," "), axis=1)
test['preprocess'] = test.apply(lambda row: row['comment_text'].replace("\n"," "), axis=1)

#removes urls and html
train['preprocess']=train.apply(lambda row: re.sub('http://\S+|https://\S+', 'urls',row['preprocess']).lower(), axis=1)
test['preprocess']=test.apply(lambda row: re.sub('http://\S+|https://\S+', 'urls',row['preprocess']).lower(), axis=1)

#remove all non-alphanumeric values
train['preprocess']=train.apply(lambda row: re.sub('[^A-Za-z\' ]+', '',row['preprocess']).lower(), axis=1)
test['preprocess']=test.apply(lambda row: re.sub('[^A-Za-z\' ]+', '',row['preprocess']).lower(), axis=1)

#remove stopwords
train['preprocess'] = train['preprocess'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
test['preprocess'] = test['preprocess'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

Figure 1

Training and Testing (Completed by Kalyan Bandaru)

After preprocessing the data, the new data was saved as new train and new test data. We split the data in to features and targets, and converted text into an array of integers with the help of tokenizer. A model was created to predict the type of toxicity for each comment, shown in Figure 2.

```
#Lets create a model and train it
model = Sequential()
model.add(Embedding(30000, 128))
model.add(LSTM(units = 128, dropout = 0.2, recurrent_dropout = 0.2,return_sequences=True)
model.add(LSTM(units = 128, dropout = 0.2, recurrent_dropout = 0.2))
model.add(Dense(units = 6, activation = 'sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 128)         3840000
_________________________________________________________________
lstm (LSTM)                  (None, None, 128)         131584
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense (Dense)                (None, 6)                 774
=================================================================
Total params: 4,103,942
Trainable params: 4,103,942
Non-trainable params: 0
```

Figure 2

Model Evaluation (Completed by Kalyan Bandaru)

In this section, we simply calculated the performance and accuracy of the model created using plots and data frames. The following screenshot shows a snippet of it.

```
from sklearn.model_selection import train_test_split
```

```
model.compile(loss = "binary_crossentropy", optimizer = "adam", metrics = ["AUC"])
```

```
x_train, x_val, y_train, y_val = train_test_split(X_train, y_train,shuffle=True, random_state = 42)
```

```
print(x_train.shape, y_train.shape, x_val.shape, y_val.shape)
```

```
(119678, 200) (119678, 6) (39893, 200) (39893, 6)
```

```
history=model.fit(x_train, y_train, batch_size = 256, epochs = 6, validation_data = (x_val, y_val))
```

```
Epoch 1/6
468/468 [==============================] - 2264s 5s/step - loss: 0.1043 - auc: 0.9015 - val_loss: 0.0556 - val_auc: 0.9777
Epoch 2/6
468/468 [==============================] - 2250s 5s/step - loss: 0.0510 - auc: 0.9780 - val_loss: 0.0533 - val_auc: 0.9775
Epoch 3/6
468/468 [==============================] - 2249s 5s/step - loss: 0.0481 - auc: 0.9809 - val_loss: 0.0526 - val_auc: 0.9733
Epoch 4/6
468/468 [==============================] - 2315s 5s/step - loss: 0.0421 - auc: 0.9860 - val_loss: 0.0517 - val_auc: 0.9767
Epoch 5/6
468/468 [==============================] - 2294s 5s/step - loss: 0.0392 - auc: 0.9875 - val_loss: 0.0537 - val_auc: 0.9700
Epoch 6/6
468/468 [==============================] - 2288s 5s/step - loss: 0.0367 - auc: 0.9891 - val_loss: 0.0559 - val_auc: 0.9648
```

Figure 3

References

Aimenchaudhry, “CSCE-5290-4290/online_abuse_comm.ipynb at Main · Aimenchaudhry/CSCE-5290-4290,” GitHub. [Online]. Available: https://github.com/aimenchaudhry/CSCE-5290 4290/blob/main/online_abuse_comm.ipynb. [Accessed: 08-Nov-2021].

“Toxic comment classification challenge,” Kaggle. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. [Accessed: 08-Nov-2021].