Data Collection and Preprocessing Phase

Date	26 November 2024
Team ID	SWTID1726490119
Project Title	Toxic Comment Classification for Social Media
Maximum Marks	6 Marks

Preprocessing Template

The text data will be preprocessed by cleaning, normalizing, tokenizing, augmenting, denoising, embedding, applying batch normalization, and, if necessary, whitening. These steps will improve data quality, support model generalization, and enhance convergence during neural network training, ensuring effective and reliable performance across various natural language processing tasks.

Section	Description
Data Overview	Overview of the text data used in the project, including the train.csv and test.csv files, with features like comment_text and multi-label targets (toxic, severe_toxic, obscene, threat, insult, and identity_hate).
Text Cleaning	Lowercasing: converting all text to lowercase to maintain uniformity. Removing special characters: Eliminating symbols, punctuation, and numbers. Removing single characters: Removes isolated single characters (e.g., "a", "b"). Removing multiple spaces: Consolidates multiple spaces into a single space.
Tokenization	Split sentences into individual tokens (words or sub-words) to allow neural network processing. Using a tokenizer to convert comment_text into sequences for further input into models.
Padding	Padding standardizes all input sequences by adding zeros to shorter sequences or truncating longer ones at the end, creating uniform input shapes for the model.
Handling imbalanced data	Calculates class weights to handle class imbalance by converting multi-label data to single-label, computing balanced class weights, and storing them in a dictionary for model

	training.
Data splitting	Split the data into training and validation sets to monitor model performance and prevent overfitting.
Data Preprocessing Code Screenshots	
Data Overview	`[ ] train_data.describe()`

	toxic	severe_toxic	obscene	threat	insult	identity_hate	target
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805	0.101679
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420	0.302226
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Text Cleaning

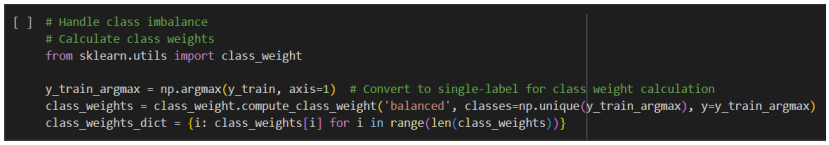
```
[ ] def clean_text(text):
        text = text.lower()  # Convert to lowercase
        text = re.sub(r'\W', ' ', text)  # Remove special characters
        text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)  # Remove single characters
        text = re.sub(r'\s+', ' ', text)  # Remove multiple spaces
        return text

[ ] train_data['comment_text'] = train_data['comment_text'].apply(clean_text)
```

Tokenization

```
[ ] num_words=20000
    tokenizer = Tokenizer(num_words)  # Only the top 20,000 words
    tokenizer.fit_on_texts(train_data['comment_text'])
    X_train = tokenizer.texts_to_sequences(train_data['comment_text'])
    X_test = tokenizer.texts_to_sequences(test_data['comment_text'])

[ ] import pickle

    # Assuming `tokenizer` is your fitted Tokenizer instance
    with open('tokenizer.pkl', 'wb') as handle:
        pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

Padding

```
[ ] maxlen = 200  # Maximum length of input sequences
    X_train= pad_sequences(X_train, padding='post', maxlen=maxlen,truncating='post')
    X_test = pad_sequences(X_test, padding='post', maxlen=maxlen,truncating='post')

[ ] # Output the shapes to verify
    print(f'Train data shape: {X_train.shape}, Train labels shape: {y_train.shape}')
    print(f'Test data shape: {X_test.shape}')

    Train data shape: (159571, 200), Train labels shape: (159571, 6)
    Test data shape: (153164, 200)
```

Handling imbalanced data	
Data splitting