CY Shield

Homonyms Problem in the text

Overview

The goal of this project is to perform sentiment analysis using the BERT (Bidirectional Encoder Representations from Transformers) model in order to classify movie reviews as either positive or negative. The BERT model is a powerful language representation model that has achieved state-of-the-art performance on various natural language processing tasks.

Sentiment analysis is a text classification task that aims to determine the sentiment expressed in a piece of tex with respect to context of the t, in this case, movie reviews. By training a machine learning model on a labeled dataset of movie reviews, we can teach the model to recognize patterns and make predictions on unseen reviews.

Data set

The dataset used for this project is the IMDB Movie Review dataset, which contains a collection of 50,000 movie reviews labeled as positive or negative. Each review is associated with a sentiment label indicating whether the review expresses a positive or negative sentiment.

The dataset is divided into training and testing sets. Due to limited computational resources, a **stratified sampling** technique was applied to randomly select 200 positive and 200 negative reviews, resulting in a balanced dataset for training and evaluation.

Baseline Experiments

1.Data preprocessing

Removing punctuation: eliminating punctuation marks from the text data before further analysis or processing. Punctuation marks include characters such as commas, periods, exclamation marks, quotation marks. regular expressions (regex) are used to match specific characters or character groups in a string. In this case, the regular expression pattern is used to identify any characters that are not alphanumeric (letters or numbers) or whitespace

Converting text to lowercase: By converting all characters to lowercase, we ensure that the same word or term, regardless of its capitalization, is treated as identical throughout the text .Also we ensure compatibility with these pre-trained models and facilitate better word representation learning.

Eliminating stop words:The goal of eliminating stop words is to remove these common words from the text before further analysis. By doing so, we can focus on the more meaningful and content-bearing words in the text,
performing word stemming using the Porter stemmer algorithm.Examples of stopwords in English include words like "the," "and," "is," "are," "in," "to," etc.

Lemmatization: normalize words so that different forms of the same word can be treated as a single word, thereby reducing the complexity and improving the accuracy of natural language processing tasks.

2.Tokenization and encoding

BERT tokenizer : When using the **BERT tokenizer**, the preprocessed reviews (cleaned and lemmatized) are passed as input. The tokenizer will split the text into tokens, taking into account word boundaries, punctuation, and special characters. For example, the sentence "I love this movie!" might be tokenized into ['I', 'love', 'this', 'movie', '!'].

Mapping : After tokenization, the tokens need to be encoded into numerical representations that can be fed into the BERT model. Each token is mapped to a unique integer, called an index, and additional tokens are added to mark the beginning and end of the sequence.

Padding: the reviews are padded or truncated to a maximum sequence length of 256 tokens to prevent session crash .

Context word embeddings: The BERT model is loaded, and the tokenized and encoded reviews are passed through the model to generate context word embeddings, which capture the contextual meaning of the reviews.

After processing reviews and generating context embedding .These embeddings are then passed through a dense layer to classify these reviews as positive or negative

Tensorflow Model

1. Context Embeddings:

- The input to the model is a sequence of context embeddings generated using the BERT model.
- The input shape of the model is `(X_train.shape[1], X_train.shape[2])`, which implies a 2D input with a fixed number of features

2. **Dense Layers:**

- The model architecture consists of multiple dense layers, which are fully connected layers.
- Each dense layer has a specified number of neurons (units) and an activation function.
- he `Flatten` layer is used to flatten the input and convert it into a 1D representation before passing it to the dense layers

3. **Dropout Layers:**

- Dropout layers are used after each dense layer to introduce regularization and prevent overfitting.
- A dropout layer randomly sets a fraction (specified as a parameter) of input units to 0 during training, which helps reduce interdependent learning among neurons.

4. **Output Layer:**

- The last dense layer has a single neuron with a sigmoid activation function.
- This configuration is suitable for binary classification, where the output represents the probability of belonging to one class (activated) or the other class (deactivated).

5. **Model Compilation:**

- The model is compiled with the binary cross-entropy loss function, which is appropriate for binary classification tasks.
- The Adam optimizer is used to optimize the model's parameters during training.
- The chosen metrics for evaluation are accuracy, which measures the percentage of correctly classified samples.

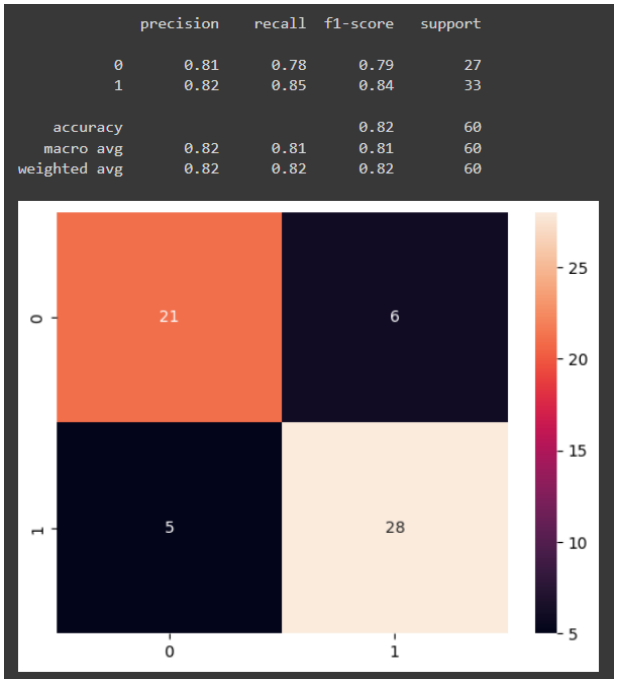
6. **Model Training:**

- The model is trained on the training data (`X_train` and `y_train`) for a specified number of epochs (200) and a given batch size (32).
- During training, the model adjusts its internal parameters using the optimization algorithm to minimize the defined loss function

7. **Model Validation:**

- The validation data (`X_val` and `y_val`) are used to assess the model's performance after each epoch during training.
- The validation data helps monitor the model's generalization and detect overfitting.
-

By fitting the model to the training data and validating it on the validation data, the model learns to classify new examples and can be evaluated using various metrics, such as accuracy because we have a balanced data and also F1 score was applied



Overall Conclusion

In conclusion, this project focused on sentiment analysis using the BERT model for movie reviews. The baseline experiments provided an initial understanding of the model's performance. By preprocessing the data, encoding the reviews, and training a neural network, we were able to achieve reasonable accuracy in predicting the sentiment of movie reviews.

Further experiments can be conducted to explore different architectures, hyperparameters, to improve the model's performance.

TOOLS

. **Natural Language Processing:**

- nltk (Natural Language Toolkit): A library for natural language processing tasks.
- stopwords: A module from nltk that provides a list of commonly used stop words.
- WordNetLemmatizer: A class from nltk used for lemmatization, which reduces words to their base or root form.

3. Machine Learning and Deep Learning:

- **tensorflow (tf):** An open-source machine learning framework used for building and training neural networks.

- **sklearn (scikit-learn)**: A library for machine learning tasks, including model evaluation and data preprocessing.

- **transformers**: A library that provides pre-trained models and tokenization tools for natural language processing, including BERT models.

- **TFBertModel**: A class from transformers that represents the BERT model.

- **BertTokenizer**: A class from transformers used for tokenization, converting text into numerical representations suitable for input to the BERT model.

tf.keras: The Keras API within TensorFlow used for building and training neural networks.

- **numpy:** A library for mathematical operations and handling arrays, used for converting embeddings to NumPy arrays.

Data Visualization:

- matplotlib.pyplot: A library for creating visualizations, used for plotting the confusion matrix.
- seaborn: A data visualization library based on matplotlib, used for creating heatmaps.

Challenges Faced

The biggest challenge faced in this project was the limited availability of computational resources, particularly GPU. As the dataset contained a large number of reviews, it was not feasible to process the entire dataset due to memory constraints. Stratified sampling was employed to create a balanced subset of positive and negative reviews for training and evaluation purposes.

Lessons Learned

Resource limitations and sampling techniques: In situations with limited computational resources, stratified sampling can be employed to create balanced subsets of the data for training and evaluation.