Reviews Sentiment Analysis Techniques: A Study on SVM, LSTM, and BERT Models

H3Art
International School
Jinan University
Guangzhou, China

***Abstract*—This project explores sentiment tri-classification in product reviews using Support Vector Machine (SVM), Long Short-Term Memory (LSTM) networks, and Bidirectional Encoder Representations from Transformers (BERT). Data from e-commerce and restaurant reviews were collected and preprocessed for analysis. The SVM, LSTM, and BERT models were employed for sentiment categorization into positive, neutral, and negative. Each model's performance was evaluated based on accuracy, precision, recall, and F1 score. The results indicated that while SVM was fastest in training and inference, it was less accurate. LSTM performed slightly better but was prone to overfitting. BERT emerged as the most effective, offering high accuracy despite higher computational requirements. This project highlights the potential of advanced machine learning models in enhancing sentiment analysis in textual data.**

***Index Terms*—reviews sentiment, SVM, LSTM, BERT**

I. Introduction

Sentiment analysis, an essential aspect of natural language processing (NLP), seeks to understand and categorize opinions expressed in textual data. Its significance lies in its vast applications, ranging from gauging public sentiment on social media to analyzing customer feedback in business. This technology enables the automatic extraction of subjective information, transforming unstructured data into actionable insights. As the volume of online textual data burgeons, sentiment analysis has become a pivotal tool for organizations to understand consumer behavior, track brand reputation, and make informed decisions.

The evolution of machine learning and deep learning has significantly enhanced the capabilities of sentiment analysis. This project focuses on the performance of three important methods: Support Vector Machines (SVM) [1], Long Short-Term Memory (LSTM) [2] networks, and Transformers Bidirectional Encoder Representation (BERT) [3] in Chinese reviews sentiment classification (negative, neutral, positive).

II. Algorithm Summary

A. *Support Vector Machine (SVM)*

SVM is a supervised machine learning algorithm, is renowned for its effectiveness in classification tasks. In sentiment analysis, SVM analyzes features extracted from text to classify sentiments. Its strength lies in handling high-dimensional data, making it suitable for text classification tasks where feature vectors are often extensive.

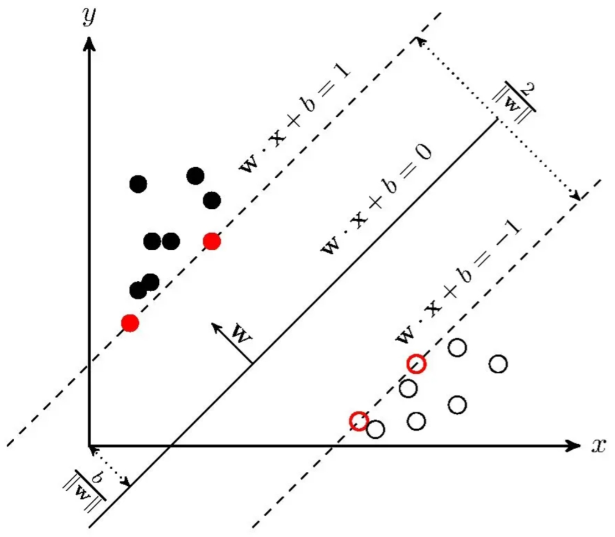

y
x
$\mathbf{w}\cdot\mathbf{x}+b=1$
$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}\cdot\mathbf{x}+b=-1$
$\frac{2}{\|\mathbf{w}\|}$
$\mathbf{w}$
$\frac{b}{\|\mathbf{w}\|}$

Fig. 1: Simplified SVM in two-dimensional space

At its core, SVM seeks to find an optimal hyperplane in a high-dimensional space that maximally separates different classes of data. In its simplest form, for linearly separable data, SVM constructs a hyperplane with the largest margin between data points of different classes, where the closest points to the hyperplane are termed as support vectors. For non-linearly separable data, SVM utilizes kernel functions, such as linear, polynomial, and radial basis function (RBF), to transform data into a higher-dimensional space where a separating hyperplane can be found. This kernel trick enables SVM to handle complex, non-linear relationships efficiently.

In multi-class classification, such as three-class problems, SVM typically employs strategies like One-vs-Rest (OvR) or One-vs-One (OvO). The OvO strategy, which is the default in many implementations like scikit-learn's SVC, involves training an SVM model for each pair of classes, and the final classification is based on a majority voting mechanism.

B. *Long Short-Term Memory (LSTM)*

LSTM is a type of recurrent neural network (RNN), are designed to address the shortcomings of traditional RNNs in processing sequences. LSTM units include memory cells that retain information over long sequences, effectively capturing the context in textual data.

The key innovation of LSTM is its cell structure, comprising four interacting components: the cell state, the input gate, the output gate, and the forget gate.

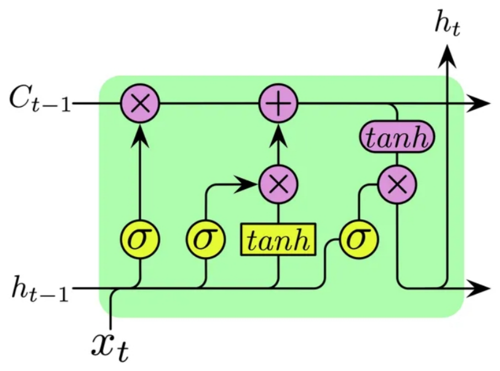


Fig. 2: LSTM unit structure

- **Cell State**: The core of LSTM, acting as a conveyor belt that runs through the entire chain. It can retain information over long periods, and slight modifications are made to it at every step via gates.
- **Forget Gate**: Determines what information to discard from the cell state. It takes the current input and the previous hidden state, applies a sigmoid function, and outputs a number between 0 and 1 for each number in the cell state, with 0 indicating "completely forget" and 1 indicating "completely retain'.
- **Input Gate**: Decides the new information to be stored in the cell state. A sigmoid layer decides which values to update, and a tanh layer creates a vector of new candidate values that could be added to the state.
- **Output Gate**: Determines what the next hidden state should be. A sigmoid layer decides which parts of the cell state to output, and then the cell state is put through tanh (to normalize values between -1 and 1) and multiplied by the output of the sigmoid gate, so that only the decided parts are output.

This structure allows LSTMs to effectively retain or discard information over long sequences, providing them with a form of memory, also makes LSTMs particularly adept at sentiment analysis, where understanding the sequence and context of words is crucial for accurate interpretation.

C. Bidirectional Encoder Representations from Transformers (BERT)

BERT represents a significant leap in NLP. Developed by Google, BERT is a transformer-based model that uses bidirectional training to understand the context from both left and right sides of a target word within a text. This approach allows for a more profound and nuanced understanding of language, greatly enhancing the model's ability to discern sentiment. BERT's effectiveness in sentiment analysis stems from its deep understanding of language context, making it highly efficient in accurately categorizing sentiments in complex textual data. At the same time, fine-tuning the pre-trained BERT model can also handle many NLP tasks.

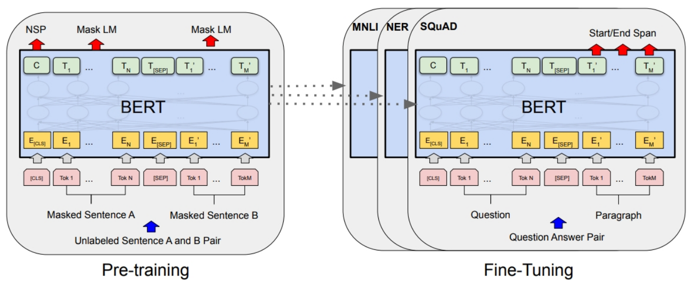


Fig. 3: Overall pre-training and fine-tuning procedures for BERT

A key feature of BERT is its minimal preprocessing requirement. Traditional NLP models required extensive preprocessing, including tokenization and generation of various linguistic annotations. BERT, however, simplifies this process significantly. It uses the WordPiece tokenization method, capable of breaking down words into smaller units (tokens), allowing the model to understand and process even the words it hasn't seen before.

Moreover, BERT comes with pre-trained word embeddings, which are representations of words in a high-dimensional space. These embeddings encapsulate a wealth of language understanding and context, acquired through training on vast amounts of text data. When BERT is used for specific tasks, these embeddings are fine-tuned in conjunction with additional output layers, enabling the model to adapt to a wide range of NLP tasks, including sentiment analysis, question answering, and language inference.

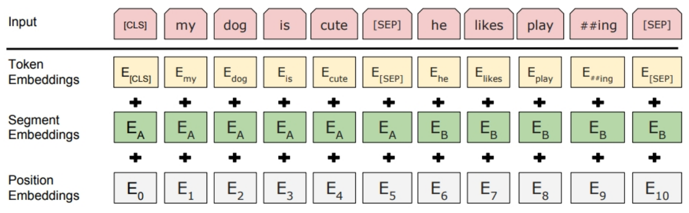

Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Fig. 4: BERT input representation

The combination of bidirectional context awareness, minimal preprocessing requirements, and versatile applicability makes BERT a highly efficient and powerful tool in the field of NLP. Its ability to understand nuanced language context and adapt to various tasks with limited additional training has set a new standard in the field.

III. Data Construction

A. Data Collection

1) Data Source: The data for this study was primarily sourced from e-commerce platforms(e.g. https://www.tmall.com/) and restaurant review sites(e.g. https://www.dianping.com/guangzhou). Throughout the training and evaluation process, the label for positive evaluation is 0, the label for neutral evaluation is 1, and the label for negative evaluation is 2.

2) Data Collection Methods: To efficiently gather this information, I utilized the Selenium web automation tool [4], which enabled me to programmatically navigate through these websites and extract textual reviews.

The process of data collection was met with certain challenges. Websites often employ anti-scraping measures, which I circumvented by implementing manual login processes through QR code scans. This step was crucial to gain access to the desired data without violating website policies.

To extract relevant review content, I employed techniques like targeting specific class attributes and XPath selectors. Some platforms offered explicit sentiment labels for reviews, aiding in initial classification, in contrast, others required manual categorization.

3) Handling Data Imbalance and Partitioning Data Sets: A notable issue in the dataset was the imbalance in sentiment categories—negative, neutral, and positive.

- **Positive Reviews**: These were more readily available, as customers often leave feedback when their experience is exceptionally good. This abundance, however, could skew the data towards a positive sentiment bias.
- **Negative Reviews**: While crucial for a balanced dataset, negative reviews were less prevalent compared to positive ones. Customers dissatisfied to the point of leaving feedback are fewer, making it a challenge to gather a substantial number of negative reviews.
- **Neutral Reviews**: These were the rarest. Customers typically do not leave reviews when their experience is neither distinctly positive nor negative. Additionally, classifying a review as neutral often depended on subjective judgment, adding a layer of complexity to the data collection process.

This imbalance creates a potential bias in model training that may lead to a "long-tailed effect", requiring strategies to ensure balanced representation of all classes to avoid bias during training. Therefore, I unified the number of samples for each label in the data set to a similar size. Finally, data with different labels are distributed in different data set files. The training and validation sets are stored in train_pos.csv, train_neu.csv, train_neg.csv, and the test set is stored in test_pos.csv, test_neu.csv, test_neg.csv.

Sentiment Category	Training Set & Validation Set	Test Set
Positive	2400	603
Negative	2400	601
Neutral	2400	601

TABLE I: Data Set Partition

Finally, during training, the training and validation sets were divided again at a ratio of 4:1, using the train_test_split method of the scikit-learn library, and setting the random_state value to 42. The final structure of the data set is shown in TABLE I.

B. Feature Enginneering

In feature engineering, I adopted a more comprehensive data processing and feature engineering method for the first two models (SVM, LSTM). As for the BERT model, it has certain restrictions on the input. I used the encoding method that comes with BERT directly encodes the raw text input.

1) Data Cleaning Strategy for SVM and LSTM models: For the SVM and LSTM models, a comprehensive data cleaning strategy was essential to ensure the quality and relevance of the input data.

- **Chinese Word Segmentation**: Utilizing the Jieba library [5], which is specifically designed for Chinese text segmentation, to accurately break down the reviews into individual words.
- **Removal of Stop Words and Special Symbols**: The application of the Harbin Institute of Technology's stop word list effectively filtered out irrelevant words of sentiment analysis, while manual cleaning efforts were directed at removing non-essential special symbols from the text.

2) Feature Extraction for SVM and LSTM Models: I employed the Word2Vec model [6], words were transformed into numerical vectors. Word2Vec is an innovative approach in the field of NLP to represent words in a numerical form, specifically as vectors in a high-dimensional space. Developed by a team led by Tomas Mikolov at Google, Word2Vec allows for the capture and representation of semantic and syntactic word relationships based on their context in large text corpora. (this very influential work won the NeurIPS Test of Time Award just a few days ago)

Word2Vec operates on the principle that words appearing in similar contexts tend to have similar meanings. By analyzing a large corpus of text, it learns to associate words with their contextual usage, effectively capturing a wide range of semantic and syntactic word relationships. It primarily employs Continuous Bag-of-Words model (in the default implementation of gensim library), averages the context words' vectors and uses this average to predict the target word. CBOW is efficient for smaller datasets and better at representing more frequent words.

Finally, to manage the high-dimensional nature of text data, I set a truncation length at 100 words, based on the average length of comments in the dataset. This approach ensured that the size of the feature vectors was practical for model processing.

3) Feature Engineering for BERT Model: In contrast to the SVM and LSTM models, the BERT model streamlined the feature engineering process. Because BERT's pre-trained feature extractor can be directly employed for tokenizing the raw text and extracting context-aware word embeddings. This approach significantly reduced the need for extensive data preprocessing, as BERT's advanced capabilities allowed for effective handling of various text structures. The extracted features from BERT were then passed into a fine-tuned downstream model, specifically constructed for the sentiment analysis task.

IV. Model Training and Optimization

A. Hardware and Software Setup for Model Development

1) Hardware Configuration:

- Device: Zephyrus G14 Laptop.
- CPU: AMD Ryzen 9 5900HS.
- RAM: 32GB of memory.
- GPU: NVIDIA GeForce RTX 3060 Laptop GPU with 6GB VRAM.

2) Software Environment:

- Operating System: Windows 10
- Python Version: Python 3.8.18
- Key Libraries and Frameworks:
  - Tensorflow 2.10.1 and Keras 2.10.0: For building and training neural network models.
  - Gensim 4.3.0: Employed for Word2Vec feature extraction.
  - Scikit-learn 1.3.0: Used for traditional machine learning models and metrics.
  - Transformers 4.35.2: For implementing and fine-tuning the BERT model.
  - Jieba 0.42.1: For text segmentation.
  - NumPy 1.21.0, Pandas 2.0.3, Matplotlib 3.7.2, and Seaborn 0.12.2: For data manipulation and visualization.

B. SVM Model

For optimizing the SVM model, GridSearchCV, a powerful tool from the scikit-learn library, was employed. This method systematically works through multiple combinations of parameter tunes, cross-validating as it goes to determine which tune gives the best performance.

SVM has several main parameters, and their meanings are as follows:

1) **C - Regularization Parameter**: The C parameter in SVM acts as a regularization factor. Regularization is a technique used to prevent overfitting by penalizing the model for being too complex.
   A small value for C makes the decision surface smooth and simple, which might lead to underfitting. It places more emphasis on finding a larger margin separating hyperplane, even if that hyperplane misclassifies more points.
   A large value of C aims for a higher classification accuracy by making the decision boundary more sensitive to the data points, which can lead to overfitting. In this case, the model tries to classify all training examples correctly.
2) **Kernel**: The kernel function transforms the input data into a higher-dimensional space where a linear separator might exist. It's a way of computing the dot product of two vectors (data points) in a higher-dimensional space without actually having to compute their coordinates in that space. Linear kernel and RBF (Radial Basis Function) are two basic types of kernel functions.
   Linear Kernel is suitable for linearly separable data. It is the simplest form of kernel, essentially not transforming the space.
   RBF Kernel is useful for non-linearly separable data. It can map an input space in infinite dimensional space. The RBF kernel, particularly, is useful when there is no prior knowledge about the data.
3) **Gamma**: In the RBF kernel, the gamma parameter defines how far the influence of a single training example reaches. It is a parameter of the Gaussian function used in the RBF kernel.
   A low value of gamma implies a large similarity radius which results in more points being grouped together. For small gamma, the model is too constrained and cannot capture the complexity or "shape" of the data. The decision boundary is very smooth.
   Inversely, A high value of gamma will consider only points close to the plausible hyperplane. For large gamma, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting.

Therefore, I set the following parameters to search for the best parameters of the SVM model. Five-fold cross-validation was used during the search, and the evaluation criterion was F1 weighted:

Parameter	Value 1	Value 2	Value 3	Value 4
C (Regularization)	10	50	100	200
Kernel	RBF	Linear		
Gamma	Scale	Auto		

TABLE II: Grid Search Parameters for SVM

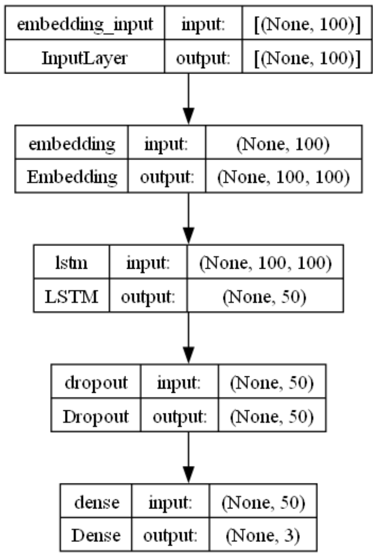
Among these parameters, the best parameter combination obtained by the final search is C=100, using the RBF kernel function, and using the scale Gamma value.

C. LSTM Model

LSTM model can handle sequence data and its capacity to address long-term dependencies, a characteristic inherent in text. In constructing the LSTM model, the Keras framework was utilized. The architecture, comprehensively visualized via "keras.utils.vis_utils", provided a clear depiction of the model's structure, ensuring an in-depth understanding of the data flow within the network.

The input sequence length was configured at 100, aligning with the average length of textual data in the dataset. The "categorical_crossentropy" loss function was selected, aligning with the model's goal of multi-class classification in sentiment analysis. The batch size was capped at 64, dictated by the limitations of the available hardware. This constraint was a balancing act between leveraging computational resources efficiently and ensuring model effectiveness.

In the construction of the LSTM model for this sentiment analysis project, I encountered the phenomenon of overfitting



embedding_input | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
embedding | input: | (None, 100)
Embedding | output: | (None, 100, 100)
lstm | input: | (None, 100, 100)
LSTM | output: | (None, 50)
dropout | input: | (None, 50)
Dropout | output: | (None, 50)
dense | input: | (None, 50)
Dense | output: | (None, 3)

Fig. 5: LSTM model structure

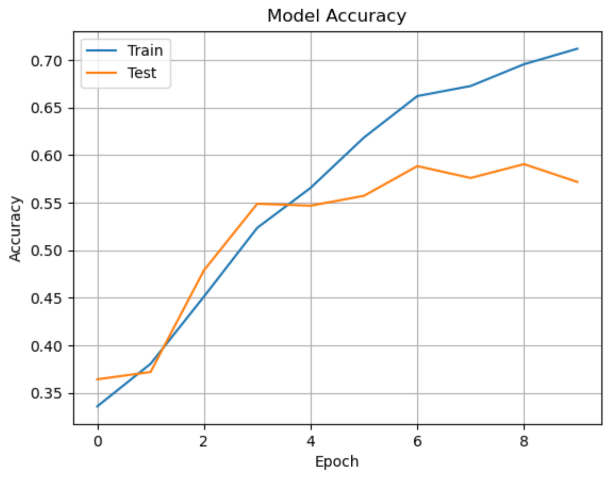


Fig. 6: LSTM model training record

of the model, so I incorporated several techniques to combat the challenge of overfitting, a common issue in deep learning models.

Dropout layers were used both within the LSTM layer and after it. By setting a dropout rate of 0.5, a significant portion of the neural network's connections are randomly dropped during training. This process prevents the network from becoming too dependent on any one set of features, encouraging the development of more robust internal representations of the data.

And specifically for the LSTM layer, recurrent dropout was also set at 0.5. This form of dropout is particularly effective for recurrent neural networks as it randomly omits connections within the recurrent units of the network, further enhancing the model's ability to generalize beyond the training data.

At the same time, both the kernel, the recurrent components of the LSTM layer and the final Dense layer were regularized using L2 regularization with a factor of 0.01. L2 regularization, also known as ridge regularization, adds a penalty equal to the square of the magnitude of coefficients to the loss function. This technique discourages the learning of a model that is too complex by penalizing large weights. This ensures that the decision-making process of the model does not hinge too heavily on any particular aspect of the learned features, further reducing the risk of overfitting.

For the training parameters, the default learning rate of 0.001 of the Adam optimizer was adopted. This rate balanced the convergence speed with training stability. Though the initial plan was to train the model for 20 epochs, an early stopping mechanism was integrated. This feature is crucial in mitigating overfitting by ceasing the training when overfitting indicators appear, specifically during validation.

The LSTM model's training concluded after only 10 epochs, a decision triggered by the early stopping criterion. This cessation underscored the model's propensity for still overfitting and indicated a potential area for refinement in future iterations of model development, possibly in aspects of model architecture or data preprocessing strategies.

D. BERT Model

The BERT model was employed using the BERT-based-Chinese variant from Hugging Face's Transformers library. During the fine-tuning phase, I got an interesting observation. The model was initially set up using "TFBertForSequence-Classification.from_pretrained" with "bert_base_Chinese" and configured to handle three classification labels. Despite the setup seeming adequate in theory, the initial training results were underwhelming, with accuracy hovering around 0.33. This outcome was indicative of the model's tendency to guess rather than accurately classify sentiments.

Therefore, instead of using the contents of the encapsulated TFBertForSequenceClassification class, I built directly from the TFBertModel class and manually applied a GlobalAverage-Pooling1D layer after the BERT output, followed by a dropout layer and finally a fully connected layer for classification.

After that, key parameters were carefully selected to optimize the model's performance. The "max_len" was set to 100, this parameter defined the maximum length of the input sequences, ensuring that the model could process most of the review data effectively. And the learning rate was set within the optimal range of 5×10^{-5} to 2×10^{-5}, with 3×10^{-5} being the chosen value. This rate was found to be effective in balancing the convergence speed and stability of the model training [3].

BERT's fine-tuning required only a few epochs of training for this specific task. After experimentation, it was determined that one epoch was optimal, highlighting the efficiency of BERT in quickly adapting to the task with minimal training. The batch size was set to 16, the maximum value that the available hardware could support. This parameter was chosen to maximize the utilization of computational resources while ensuring effective model training.

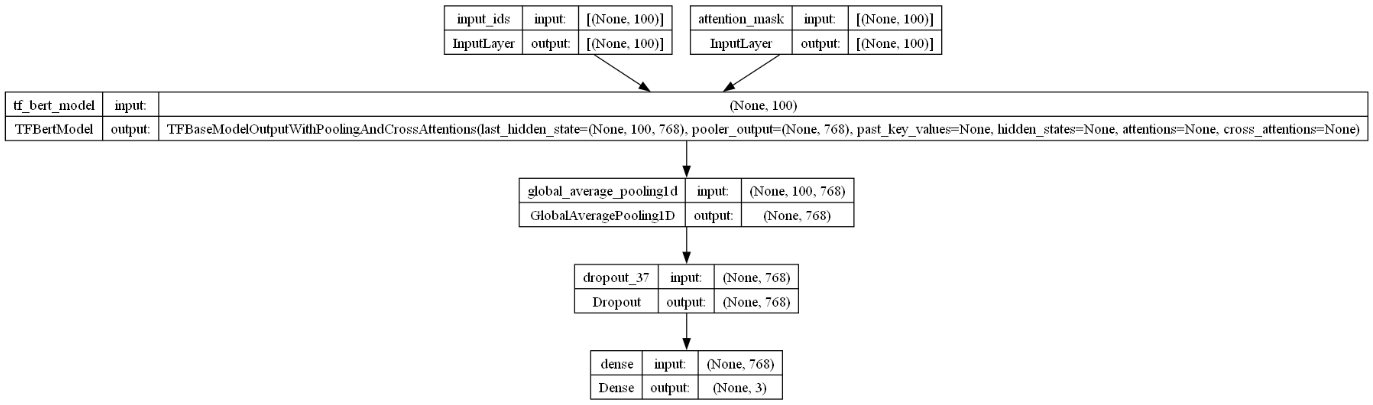

input_ids | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
attention_mask | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
tf_bert_model | input: | (None, 100)
TFBertModel | output: | TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 100, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)
global_average_pooling1d | input: | (None, 100, 768)
GlobalAveragePooling1D | output: | (None, 768)
dropout_37 | input: | (None, 768)
Dropout | output: | (None, 768)
dense | input: | (None, 768)
Dense | output: | (None, 3)

Fig. 7: BERT model structure

V. Result Analysis

A. Evaluation Metrics

In evaluating the models for this sentiment analysis project, several key metrics were used to assess their effectiveness and accuracy. These metrics provide a comprehensive picture of each model's performance and are essential in understanding their strengths and weaknesses.

1) **Accuracy**: It measures the overall effectiveness of the model by calculating the proportion of correct predictions out of the total predictions made. It is represented by the formula:

$$\textbf{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

While accuracy is a useful indicator of model performance, it may not be as reliable for datasets with imbalanced classes.

2) **Precision**: It indicates the proportion of positive identifications that were actually correct. It is especially important in scenarios where the cost of false positives is high. The formula for precision is:

$$\textbf{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

3) **Recall**: Recall reflects the ability of the model to correctly identify all actual positives. It is crucial in situations where missing a positive instance is costly. The formula for recall is:

$$\textbf{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

4) **F1 Score**: The F1 Score is the harmonic mean of Precision and Recall, providing a balance between the two. It is particularly useful when both false positives and false negatives are a concern. The F1 Score is calculated using:

$$\textbf{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}}$$

5) **Confusion Matrix Analysis**: A Confusion Matrix is a tool used to visualize the performance of a classification model, highlighting the number of true positives, false positives, true negatives, and false negatives.

These evaluation metrics were employed to comprehensively assess the performance of the SVM, LSTM, and BERT models in the project, providing insights into their accuracy, reliability, and areas for improvement.

B. Evaluation Result

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Train Time**
SVM	0.611111	0.613478	0.611111	0.596920	32.691707s
LSTM	0.616667	0.624210	0.616667	0.619433	742.944826s
BERT	0.811111	0.812916	0.811111	0.811114	182.910349s

TABLE III: Model Evaluation Result

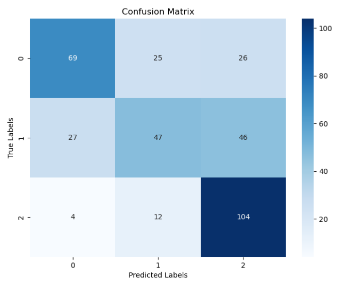
The above table encapsulates the comparative performance of the three models used in the project. The BERT model exhibited the highest scores across four metrics, indicating its superior capability in sentiment analysis. In contrast, the SVM model, while being the fastest in terms of training speed, showed lower accuracy and other performance indicators. The LSTM model demonstrated a performance slightly better than the SVM but faced challenges with overfitting.

The confusion matrix analysis of the SVM, LSTM, and BERT models in the reviews sentiment analysis project revealed insightful details about their performance, particularly in classifying different sentiment categories.

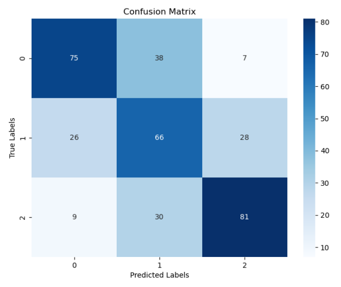
The analysis indicated that the SVM and LSTM models, which performed less effectively compared to BERT, had notably lower classification accuracy for neutral evaluations. The BERT model, despite its resource-intensive nature, showed better adaptability and accuracy in classifying sentiments, including neutral ones. This could be attributed to its advanced pre-training on a vast corpus, equipping it with a more nuanced understanding of language and context.

C. Deficiencies Analysis

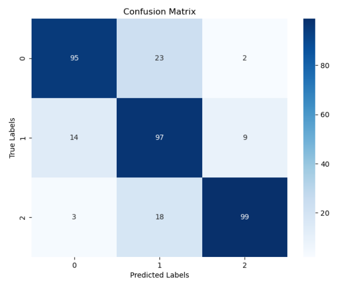
When analyzing the performance of the SVM, LSTM, and BERT models, the causes of deficiencies can be attributed



(a) SVM Model



(b) LSTM Model



(c) BERT Model

Fig. 8: Confusion Matrix of three models' evaluation

to several factors, and I also mainly concern the error of classifying neutral reviews:

1) **Ambiguity in Neutrality**: Defining and categorizing neutral sentiment in the dataset was challenging. The inherent ambiguity in what constitutes neutrality likely led to inconsistencies in training data, which in turn affected the models' ability to accurately classify such reviews.
2) **Text Length in Neutral Reviews**: It was observed that neutral reviews typically had longer text lengths compared to positive and negative reviews. This discrepancy in text length could have introduced additional complexity in the models' training process.
3) **Impact of Stop Words Removal on Neutral Reviews**: In the preprocessing phase, the application of stop word removal might have disproportionately affected neutral reviews. Neutral comments often contain a significant number of words that are emotionally neutral and could be classified as stop words. The removal of these words might have resulted in a substantial reduction in the remaining text content of neutral reviews. Consequently, this could lead to a scenario where the models struggle to learn and predict based on the significantly diminished textual content.
4) **Contextual Emotional Reversals in Neutral Reviews**: Another complexity with neutral reviews is the potential for contextual emotional reversals within the same comment. It is not uncommon for a review marked as neutral to exhibit a mix of sentiments, it starting with positive expressions and transitioning to negative ones, or vice versa. Such emotional reversals within the same text can lead to misclassifications, as the models might struggle to discern the overarching sentiment intended.
5) **Dataset Size**: The dataset, comprising 6000 reviews, was relatively small for training deep learning models like LSTM and BERT. Such a dataset size might not have been sufficient for the models to learn and generalize effectively, especially given the complexity of sentiment analysis in textual data. This limitation was evident in the overfitting phenomena observed in both LSTM and BERT models.

VI. Future Work and Conclusion

A. Future Work

Looking ahead, several avenues are open for further enhancing and utilize the models:

- **Exploring More Accurate Word Lists**: Improving the word segmentation process could refine the input data quality, potentially boosting model performance.
- **Model Deployment and User Interaction**: Efforts to deploy the trained models and facilitate user interaction could lead to practical applications and real-world testing.
- **Star Rating-Based Classification Model**: Given the success in tri-classification, expanding the model to classify based on star ratings presents an interesting direction for future research.
- **Hardware Limitations and Their Impact**: Upgrading to devices with larger memory and more powerful video memory would enable the exploration of larger batch sizes and longer input sequences. This enhancement is expected to significantly improve the learning capabilities of resource-intensive models like LSTM and BERT, thereby optimizing their performance and effectiveness in tasks like sentiment analysis.

B. Conclusion

This project, which explored the development of a sentiment analysis model using three distinct machine learning algorithms SVM, LSTM, and BERT, highlighted the varying performances of these models:

The SVM model was notable for its rapid training and inference speeds, making it a viable option for scenarios requiring quick model deployment. However, its lower accuracy and other performance metrics, even after parameter optimization through GridSearchCV, suggested room for improvement.

The LSTM model showed enhanced capability in managing complex text patterns, slightly outperforming the SVM in overall effectiveness. Yet, it consistently grappled with overfitting issues, impacting its accuracy and limiting its practical application.

Among the three, the BERT model stood out as the most proficient, excelling in accuracy, precision, recall, and F1 score, albeit with the trade-off of higher computational demands. This model demonstrated the advanced potential of modern NLP techniques in sentiment analysis.

The project underscored a crucial aspect of machine learning, particularly the importance of data quality and data processing. The project revealed that high-quality data is a prerequisite for optimal model performance. The limitations encountered in data handling and quality highlighted areas for improvement. It became evident that refining data collection and preprocessing strategies is as vital as selecting and fine-tuning the algorithms themselves.

Through this project, a strong foundation has been laid for sentiment analysis using advanced machine learning techniques. Each model brought unique advantages and offered valuable insights. The experience and challenges faced during this project have been instrumental in shaping a path for future endeavors and studies in machine learning.

References

[1] Z. Zhou, *Machine Learning*. Tsinghua University Press, 2016, pp. 121–140.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[4] SeleniumHQ, "Selenium, A browser automationframework and ecosystem," https://github.com/SeleniumHQ/selenium.

[5] A. Sun, "jieba, Chinese text segmentation," https://github.com/fxsjy/jieba.

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013.