VIETNAM NATIONAL UNIVERSITY

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

----------o0o----------


MODERN APPROACHES IN NATURAL LANGUAGE PROCESSING

# SENTIMENT ANALYSIS REPORT

Lecturer: Assoc. Prof. Quan Thanh Tho

Mai Chí Bảo 2370691

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2024

# TABLE OF CONTENTS


**1. Introduction** **3**

**2. Data and preprocessing** **3**

**In general** **3**

**Preprocessing** **5**

**3. CNN** **8**

**4. LSTM** **12**

**5. CNN and LSTM** **16**

**6. BERT** **19**

**7. Conclusion** **23**

# 1. Introduction

Sentiment analysis, or opinion mining, is a key technique in natural language processing (NLP) used to analyze and classify the sentiment expressed in text. The goal is to determine whether the sentiment is positive, negative, or neutral. In today's digital age, sentiment analysis has become increasingly vital for organizations, businesses, and individuals as it helps gauge public opinion, customer satisfaction, and market trends.

However, sentiment analysis faces several challenges due to the complexity of human language. Sentiments are often subtle, ambiguous, or context-dependent, making them difficult to classify accurately. For example:

- Sarcasm, idioms, and cultural nuances can change the intended meaning of a sentence.
- Informal language, emojis, and abbreviations—especially prevalent in social media—further complicate the analysis.

This report explores the sentiment classification of the VLSP dataset by training and evaluating four models:

1. Convolutional Neural Network (CNN)
2. Long Short-Term Memory Network (LSTM)
3. CNN + LSTM Hybrid
4. BERT-based Approach

The objective is to assess the performance of these models and determine which one is best suited for sentiment classification in this context.

# 2. Data and preprocessing

**In general**

The dataset used originates from the VLSP competition. It consists of two main columns: Class and Data.

Class: This column categorizes the reviews/documents into one of three sentiment classes:

- Positive: Represented by the value 1
- Negative: Represented by the value -1
- Neutral: Represented by the value 0

Data: This column contains the comments or reviews, which can vary in length.

The training dataset includes 5,100 rows of data.

```
dataset.shape
```

```
(5100, 2)
```

```
dataset.columns
```

```
Index(['Class', 'Data'], dtype='object')
```

The dataset is relatively balanced across the three sentiment categories:

- Positive: 1,672 entries
- Negative: 1,693 entries
- Neutral: 1,697 entries

Since the data distribution is already balanced, there is no need to apply additional techniques to address class imbalance in this case.

```
dataset["Class"].value_counts()
```

```
Class
 0    1697
-1    1693
 1    1672
Name: count, dtype: int64
```

The Data column contains comments of arbitrary length. Below are examples of the dataset:

```
dataset["Data"].sample(40)
```

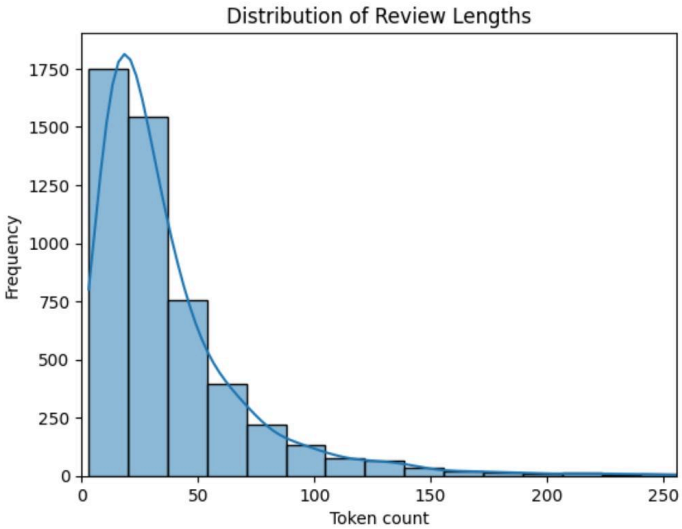
```
2232                                        Qúa kinh khủng!
2049    Công nghệ của SS thật tuyệt, biết lắng nghe, h...
1750                                    Vừa mua xong quá đẹp
1183                                Xấu thế.nhìn như máy tq.
754                                         Camera xấu vậy?
3106    Tôi đang dùng M7 từ lúc mới có One, đến giờ vẫ...
1873    tôi đã dùng nhiều dòng điện thoại khác nhau củ...
4965                               Tiền đó mua dc mi5 32G :v
473     1900 đến 2500 usd. Đắt quá, nhưng vẫn chưa đắt...
2892                                                Tuyệt vời
4913    Không sao cả vì đơn giản nó cũng như các...
1014    móa viettel làm ăn như shit, cpn từ 21 mà hnay...
133                         2016 - Iphone - who cares? :))
4266    Chương trình này của công ty Quihu Trung quốc....
4188    Anh có cảm giác sh01g lưu hình hơi chậm nên ch...
4911    Iphone hết thời làm mưa gió rồi. Nếu muô...
1791    gear fit mà pin kém gì..mình dùng vẫn 4-5 ngày...
```

**Preprocessing**

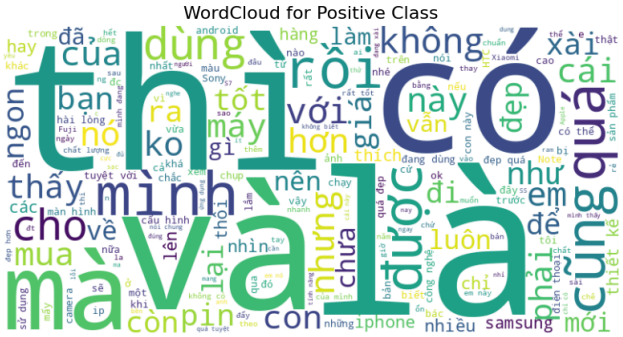
Analysis of the dataset reveals that the majority of reviews are under 90 tokens in length. Therefore, a maximum length of 90 tokens is used for processing:

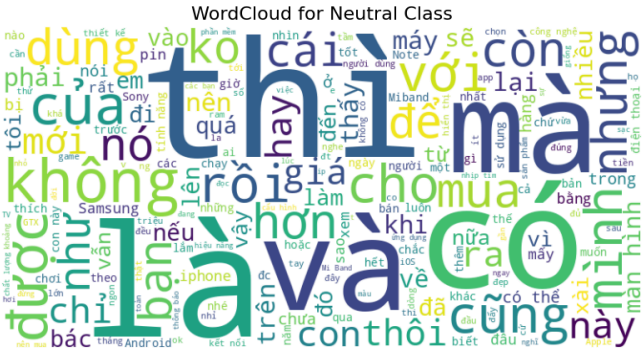
- Shorter reviews are padded with zeros to match the maximum length.
- Longer reviews are truncated to fit the 90-token limit.

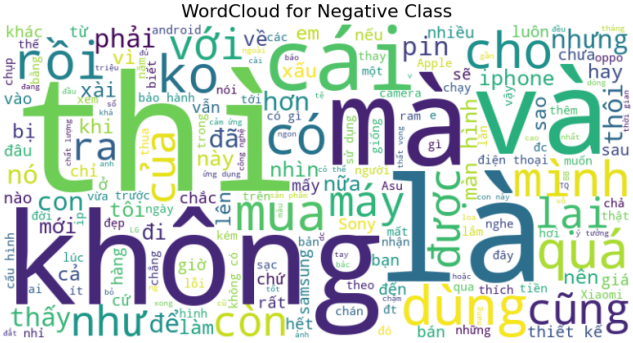
The vocabulary size of the dataset is 14,713 unique tokens, which will be used for text representation during model training.




The following word clouds illustrate the most frequently used words for each of the three sentiment classes (Positive, Negative, and Neutral):




Some prolems with the dataset and the way I solved them

- **Presence of Emojis and Special Characters:**

  Many text samples contained emojis, emoticons, and symbols that do not contribute to sentiment or meaning. To resolve this, a regular expression pattern was used to detect and remove these elements from the text.

- **Repetitive Characters:**

  Words often included repeated characters for emphasis, such as "tốttttt" or "khônggg," which could distort tokenization and recognition. A regex substitution was applied to normalize such repetitions by reducing them to a single instance (e.g., "tốttttt" → "tốt").

- **Improper Handling of Punctuation:**
  Text samples contained misplaced or redundant punctuation (e.g., "Hi!!!") that complicated parsing. Regex-based rules were implemented to normalize spacing around punctuation and remove unnecessary repetitions, creating a cleaner structure.
- **Slang and Abbreviations:**
  Common slang and abbreviations like "cx" (cũng) or "sp" (sản phẩm) varied across the text, leading to inconsistency. A lookup dictionary was created to standardize these terms by mapping them to their full forms.
- **Stop Words:**
  The dataset included stop words such as "và," "thì," and "là," which contributed noise but added little semantic value. These were removed using a predefined list of stop words, ensuring that only meaningful words were retained.
- **Case Sensitivity:**
  Variations in capitalization, such as "Good" vs. "good," caused inconsistencies during analysis. The text was converted to lowercase to standardize the format.
- **Unnormalized Text:**
  The lack of consistent formatting, including excessive or missing whitespace, made processing unpredictable. Whitespace and formatting inconsistencies were fixed using regular expressions to ensure uniformity.
- **Unsuitable Class Labels:**
  The dataset included a class labeled as -1, which was incompatible with some frameworks or models. This issue was resolved by mapping the -1 label to 2, ensuring consistency with the other classes.

To facilitate model development and evaluation, the training dataset was split into three parts:

- 80% for Training: Used to train the machine learning models.
- 10% for Validation
- 10% for Testing

In addition to this split, a separate test file containing 1,000 rows of data was designated as the final evaluation dataset for all models.

# 3. CNN

For 3 models CNN, LSTM and combination of CNN and LSTM. To prepare the textual data for the CNN, LSTM, and CNN-LSTM hybrid models, the following preprocessing steps were applied:

First I create a TF-IDF matrix from the vocabulary:

- TF-IDF (Term Frequency-Inverse Document Frequency) transforms textual data into numerical feature vectors.
- It weighs words based on their frequency in a document (TF) and their rarity across the entire corpus (IDF). This reduces the influence of very common words (like "thì", "là") while emphasizing rare and informative words.
- The max_features=30000 limits the vocabulary to the top 30000 most important words to reduce dimensionality and computational load.
- TF-IDF captures both word occurrence and importance, making it suitable for feature extraction in text data.

Then I applied the dimensionality reduction using SVD:

- SVD (Singular Value Decomposition) reduces the high-dimensional TF-IDF vectors into a lower-dimensional space.
- The reduced representation keeps the most important information while discarding noise or redundant features.
- TF-IDF can produce very high-dimensional vectors, especially for large vocabularies (30,000 features in this case).
- High-dimensional data can lead to:
    - Overfitting in model.
    - Computational inefficiency during training.
    - Curse of Dimensionality: Increased difficulty in finding meaningful patterns.
- By reducing the TF-IDF features to 300 dimensions, SVD helps balance dimensionality reduction and information retention.

**Training CNN:**

Model Architecture:

- Input Layer: Accepts feature vectors of shape (300,), which are reshaped to (10, 30) for 1D convolutions.
- Convolutional Layers:
    - Layer 1: 128 filters, kernel size of 3, ReLU activation, with same padding followed by MaxPooling (pool size 2).
    - Layer 2: 64 filters, kernel size of 3, ReLU activation, with same padding followed by MaxPooling (pool size 2).
- Flatten Layer: Flattens the 1D convolution output for dense layers.
- Dense Layers:
    - Two hidden layers: 512 and 128 neurons, both with ReLU activation.
    - Output layer: 3 neurons with softmax activation for multiclass classification.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 reshape_2 (Reshape)         (None, 10, 30)            0

 conv1d_1 (Conv1D)           (None, 10, 128)           11648

 max_pooling1d (MaxPooling1D  (None, 5, 128)           0
 )

 conv1d_2 (Conv1D)           (None, 5, 64)             24640

 max_pooling1d_1 (MaxPooling  (None, 2, 64)            0
 1D)

 flatten_1 (Flatten)         (None, 128)               0

 dense_4 (Dense)             (None, 512)               66048

 dense_5 (Dense)             (None, 128)               65664

 dense_6 (Dense)             (None, 3)                 387

=================================================================
Total params: 168,387
Trainable params: 168,387
Non-trainable params: 0
```

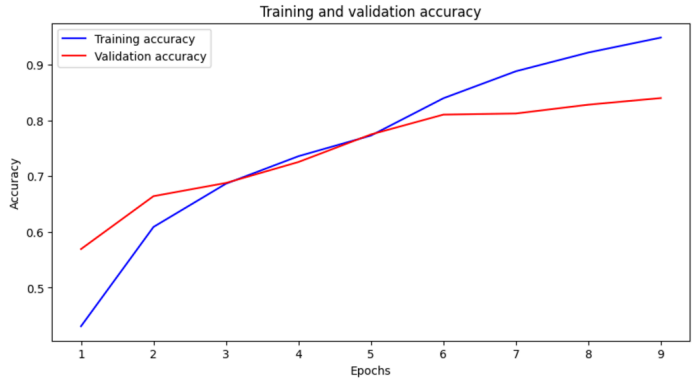
Compilation:

- Optimizer: Adam
- Loss function: Sparse Categorical Crossentropy (for multi-class labels)
- Metric: Accuracy

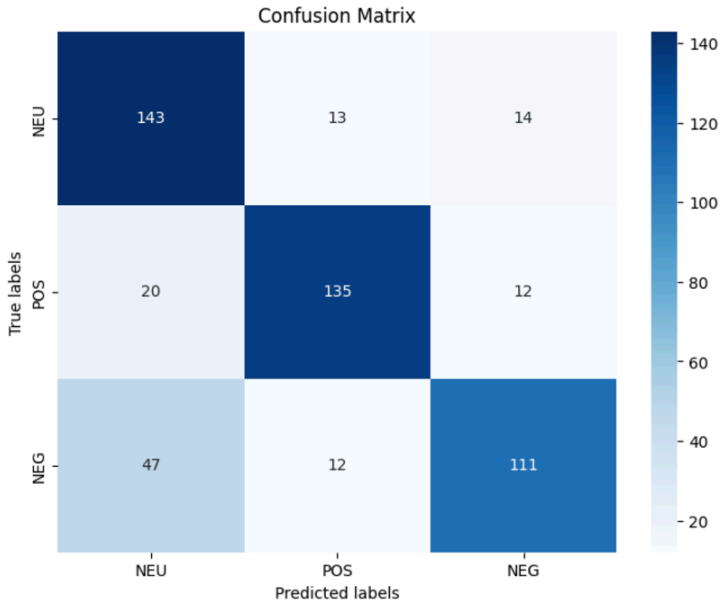
Hyperparameters:

- Epochs: 15
- Batch Size: 64

I also used Early Stopping: Stops training if the validation loss doesn't improve for 3 epochs to prevent overfitting.


Confusion matrix on the test data



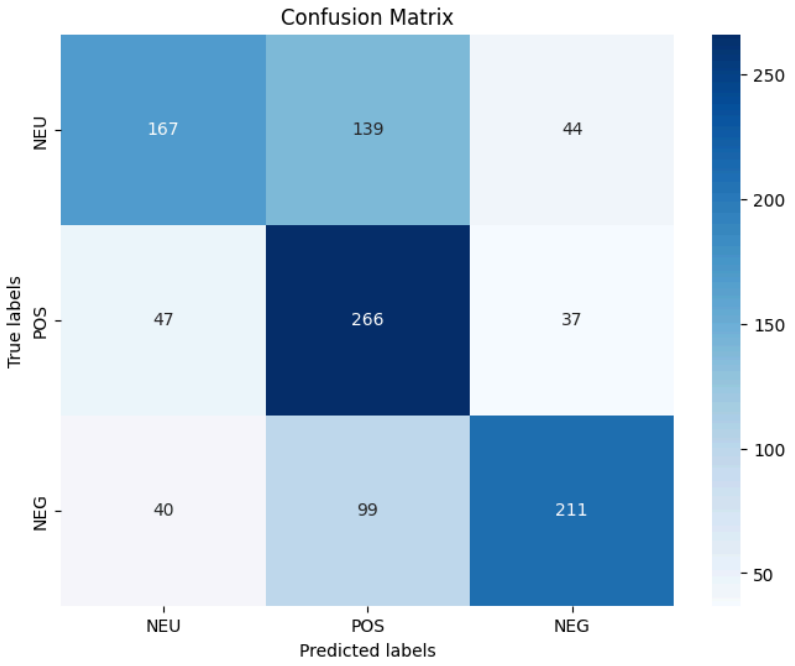

NEU (Neutral): The model performs relatively well on the neutral class, with 143 correct predictions out of a total of 210 samples. However, there's a notable number of misclassifications (13 as POS and 14 as NEG). This suggests the model sometimes struggles to distinguish neutral sentiment from positive or negative ones.

POS (Positive): The model's performance on the positive class is also good, with 135 correct classifications out of 167 samples. Misclassifications are distributed across NEU and NEG, suggesting some confusion between these sentiment classes.

NEG (Negative): The model shows bad accuracy for the negative class, with 111 correct predictions out of a total of 170 samples. However, a significant number are incorrectly classified as NEU (47). This points to a challenge in distinguishing between negative and neutral sentiments.

**Result on real test dataset:**

```
Model Accuracy: 61.33%
Classification Report:
              precision    recall  f1-score   support

         NEU       0.66      0.48      0.55       350
         POS       0.53      0.76      0.62       350
         NEG       0.72      0.60      0.66       350

    accuracy                           0.61      1050
   macro avg       0.64      0.61      0.61      1050
weighted avg       0.64      0.61      0.61      1050
```




**High number of False Positives**: The significant off-diagonal values indicate a substantial number of misclassifications. The model frequently misclassifies instances, particularly in the neutral class (139 instances misclassified as positive and 44 instances as negative).

The F1 score and recall for NEU class is quite bad. The F1 is below 0.5 for NEU

# 4. LSTM

**Training LSTM:**

Model Architecture

- Input Layer:
    - Accepts feature vectors of size (300,).
- Reshape Layer:

    - Reshapes input to (10, 30) to match the expected 3D input shape for LSTM layers.
- Bidirectional LSTM Layer:
    - Consists of 128 units with ReLU activation.
    - Bidirectional LSTM processes the input sequence in both forward and backward directions to capture contextual dependencies.
- Flatten Layer:
    - Flattens the output of the LSTM layer into a single vector for dense layers.
- Dense Layers:
    - Two fully connected layers:
    - Layer 1: 512 neurons with ReLU activation.
    - Layer 2: 128 neurons with ReLU activation.
- Output Layer:
    - 3 neurons with softmax activation for multiclass classification (e.g., "positive", "negative", "neutral").

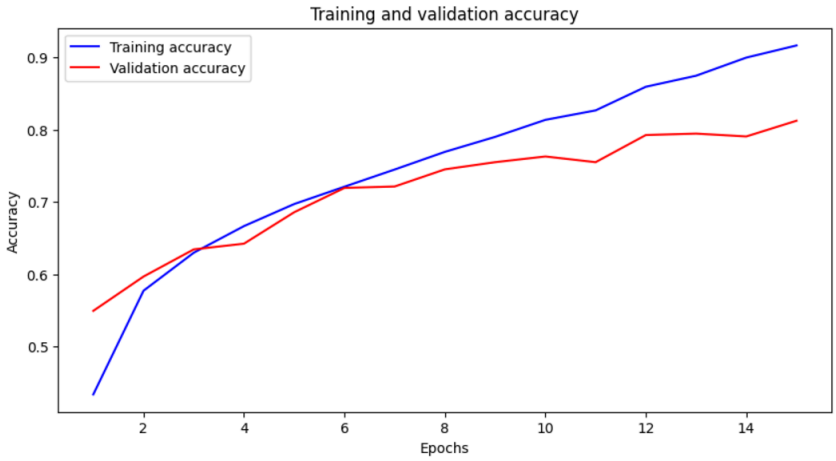
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 reshape_3 (Reshape)         (None, 10, 30)            0

 bidirectional_2 (Bidirectio  (None, 10, 256)          162816
 nal)

 flatten_2 (Flatten)         (None, 2560)              0

 dense_7 (Dense)             (None, 512)               1311232

 dense_8 (Dense)             (None, 128)               65664

 dense_9 (Dense)             (None, 3)                 387

=================================================================
Total params: 1,540,099
Trainable params: 1,540,099
Non-trainable params: 0
```

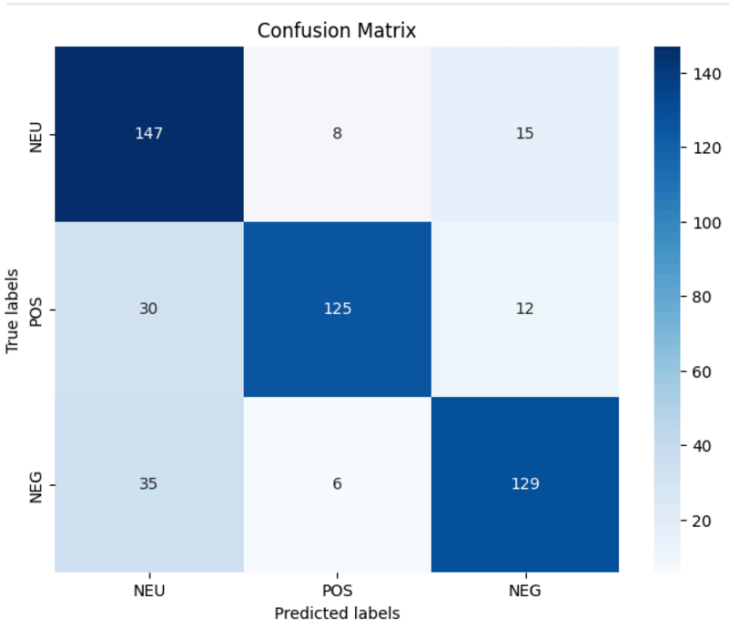
Compilation:

- Optimizer: Adam
- Loss Function: Sparse Categorical Crossentropy
- Metric: Accuracy

Hyperparameters:

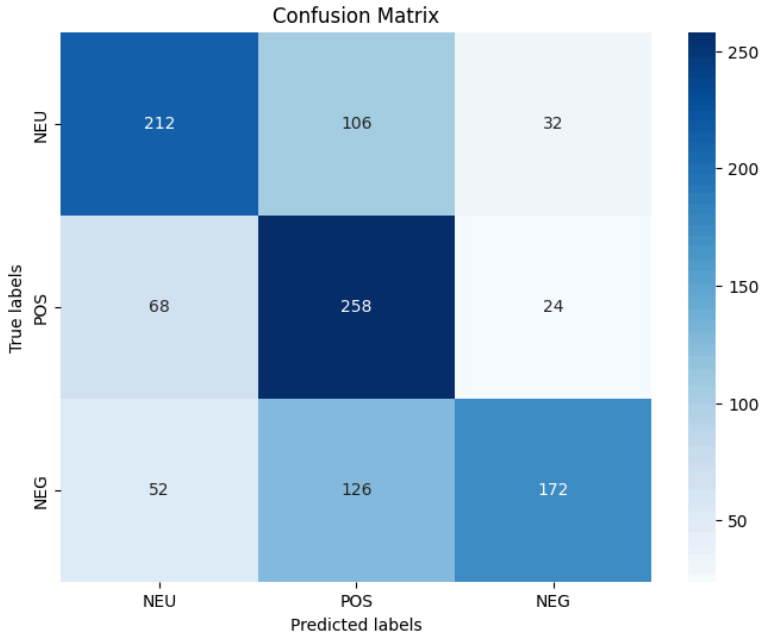
- Epochs: 15
- Batch Size: 64




## Result on real test data

```
Model Accuracy: 61.14%
Classification Report:
              precision    recall  f1-score   support

         NEU       0.64      0.61      0.62       350
         POS       0.53      0.74      0.61       350
         NEG       0.75      0.49      0.60       350

    accuracy                           0.61      1050
   macro avg       0.64      0.61      0.61      1050
weighted avg       0.64      0.61      0.61      1050
```



NEU: Precision is decent (0.64), but recall is lower (0.61). This means while the model correctly identifies a reasonable proportion of true neutral sentiments, it misses a significant portion.

Most ò the answer from the molde is POS. In NEU class A considerable number of neutral samples are misclassified as positive (106), and (126) in NEG class. This indicates the model struggles to differentiate subtle differences between other classes and positive sentiments.

# 5. CNN and LSTM

Model Architecture

- Input and Reshape:
    - Input size: (300,) reshaped into (10, 30) for RNN and CNN layers processing.
- Bidirectional GRU:
    - GRU layer with 128 units using ReLU activation.
    - Bidirectional processing captures both forward and backward dependencies in the sequence.
- Convolutional Layer:
    - 1D convolutional layer with 100 filters and a kernel size of 3.
    - ReLU activation for feature extraction over local contexts.
- Flatten Layer:
    - Converts the 2D feature map into a 1D vector for dense layers.
- Dense Layers:
    - Three fully connected layers:
    - Layer 1 & 2: 512 neurons each with ReLU activation.
    - Layer 3: 128 neurons with ReLU activation.
- Output Layer:
    - 3 neurons with softmax activation for multiclass classification.

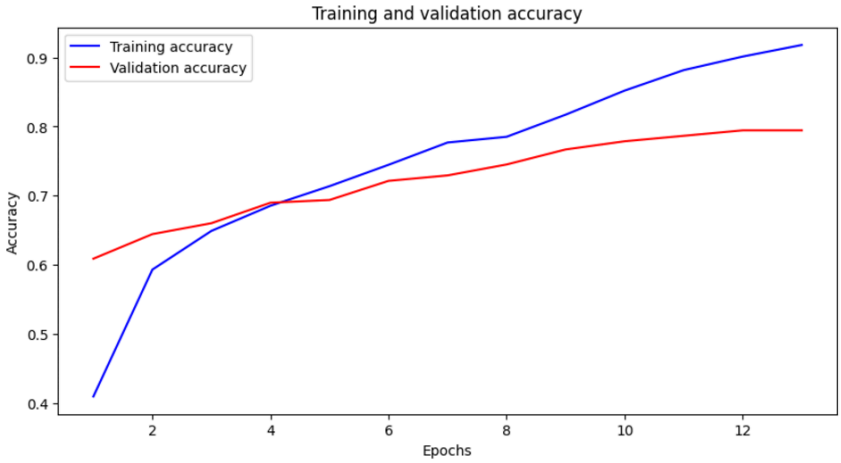
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 reshape_1 (Reshape)         (None, 10, 30)            0

 bidirectional_1 (Bidirectio  (None, 10, 256)          122880
 nal)

 conv1d (Conv1D)             (None, 8, 100)            76900

 flatten (Flatten)           (None, 800)               0

 dense (Dense)               (None, 512)               410112

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 128)               65664

 dense_3 (Dense)             (None, 3)                 387

=================================================================
Total params: 938,599
Trainable params: 938,599
Non-trainable params: 0
```

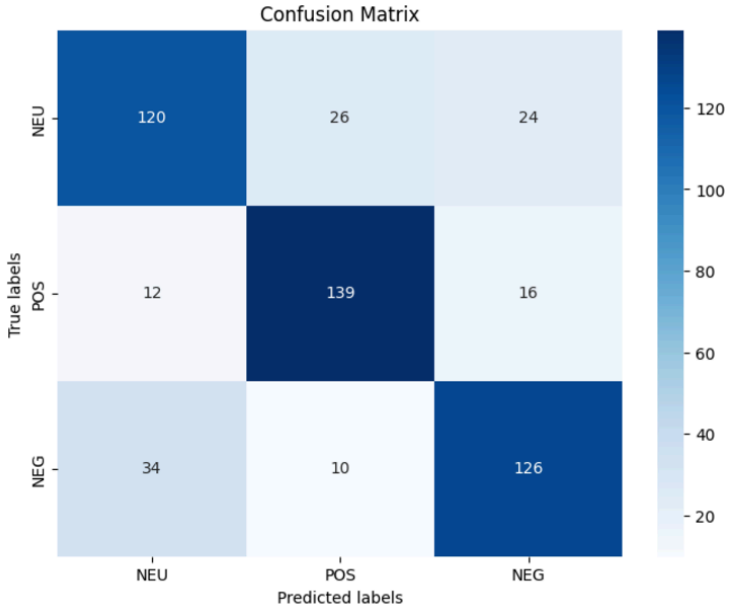
Compilation:

- Optimizer: Adam (adaptive learning rate).
- Loss Function: Sparse Categorical Crossentropy (for integer class labels).
- Metric: Accuracy.

Hyperparameters:

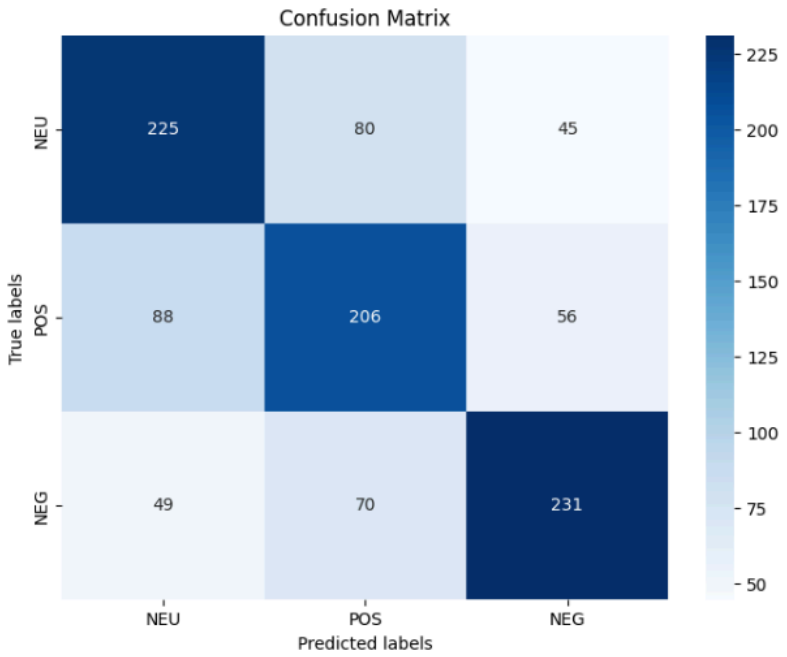
- Epochs: 15
- Batch Size: 64




**Result on real test dataset**

```
Model Accuracy: 63.05%
Classification Report:
              precision    recall  f1-score   support

         NEU       0.62      0.64      0.63       350
         POS       0.58      0.59      0.58       350
         NEG       0.70      0.66      0.68       350

    accuracy                           0.63      1050
   macro avg       0.63      0.63      0.63      1050
weighted avg       0.63      0.63      0.63      1050
```




NEG: The model performs best in this category, with precision (0.70) higher than recall (0.66). This means the model correctly identifies negative sentiments when it predicts them but misses some true negative examples.

# 6. BERT

For this task, I used a pre-trained BERT model specifically designed for Vietnamese text: bert-base-vietnamese-uncased. The model architecture was modified to include:

- A dropout layer to reduce overfitting.
- A fully connected (FC) layer to output probabilities for the three sentiment classes.

Optimizer

- Type: AdamW (Adam optimizer with weight decay for better regularization).

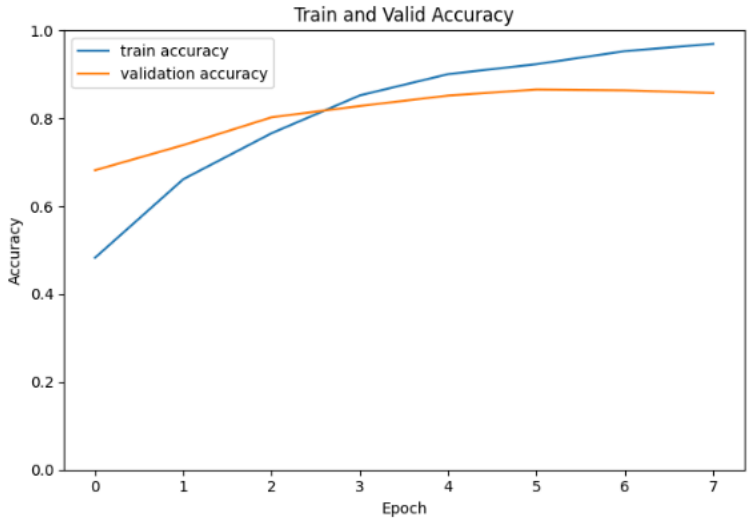
- Learning Rate: 0.00001
- Bias Correction: Disabled (correct_bias=False), a common practice in NLP tasks for better gradient updates.

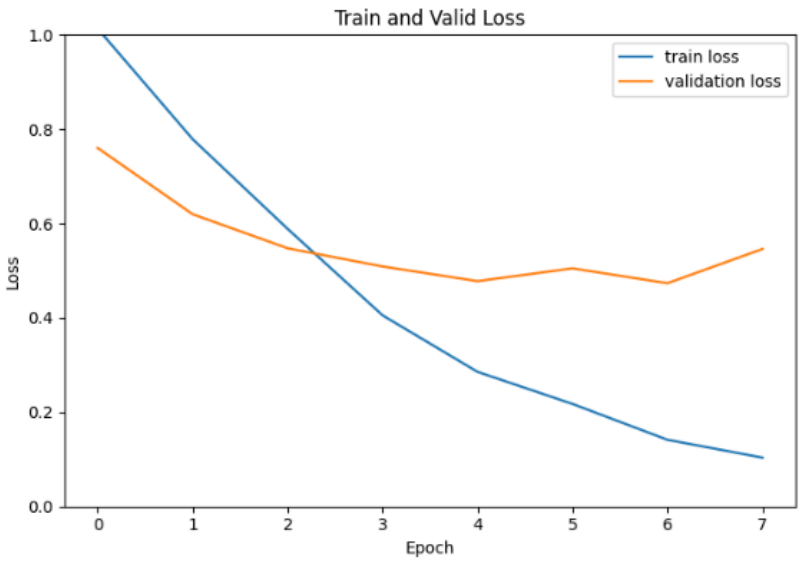
Learning Rate Scheduler

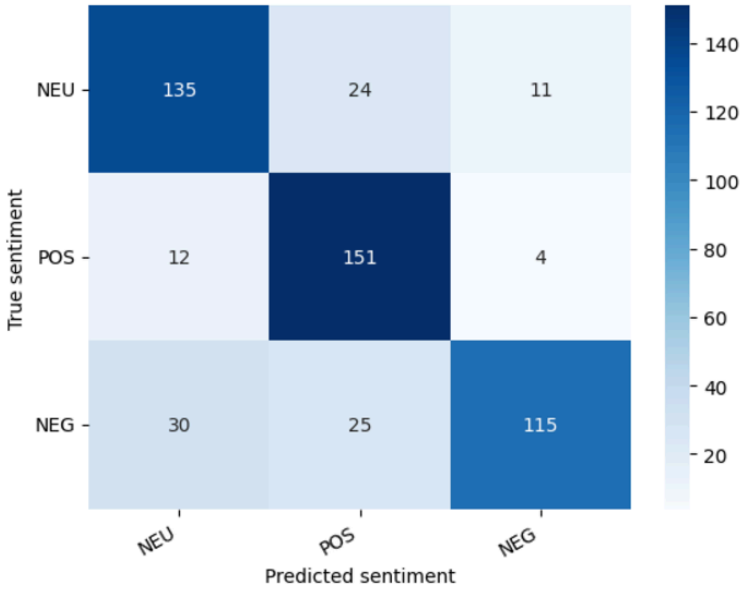
- Type: get_linear_schedule_with_warmup
- Gradually increases the learning rate during an initial warmup phase and then decreases it linearly for the remaining training steps.
- Warmup Steps: 0 (no warmup period was used in this case).

Loss Function

- Type: nn.CrossEntropyLoss
- A standard loss function for multi-class classification problems.
- Applied to the logits output by the BERT model and the corresponding target labels.



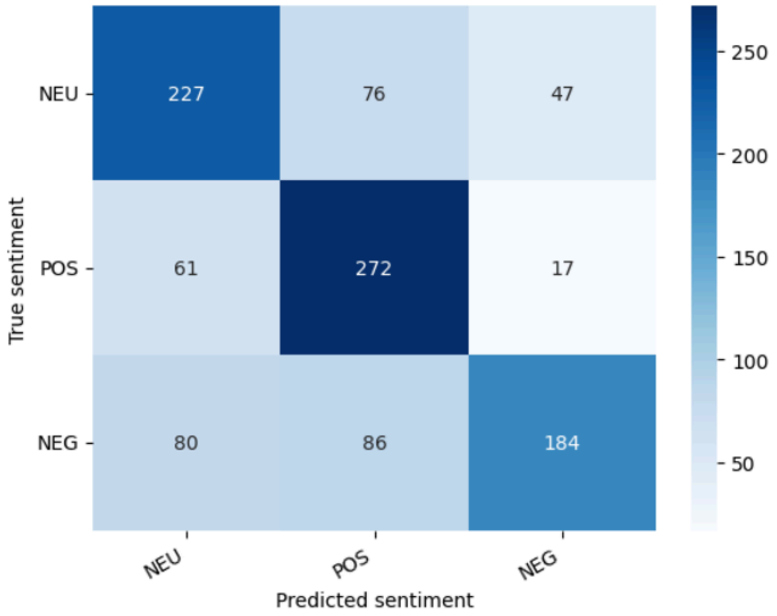



**Result on real test data**

```
Accuracy: 65.05%
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| NEU | 0.62 | 0.65 | 0.63 | 350 |
| POS | 0.63 | 0.78 | 0.69 | 350 |
| NEG | 0.74 | 0.53 | 0.62 | 350 |
| accuracy | | | 0.65 | 1050 |
| macro avg | 0.66 | 0.65 | 0.65 | 1050 |
| weighted avg | 0.66 | 0.65 | 0.65 | 1050 |




Improved Distinction between NEU and POS: Compared to previous iterations, the misclassification between neutral and positive sentiments has decreased significantly. This indicates that the model is improving its ability to distinguish these closely related sentiment classes.

Persistent Weakness in NEG: The negative class still exhibits a noticeable number of misclassifications. The model struggles to correctly identify negative sentiments consistently, especially missing more true negatives (86 misclassified as neutral) than false positives (80 misclassified as neutral).

# 7. Conclusion

In conclusion, the BERT model achieved the highest accuracy at 65.5%, followed by the CNN+LSTM hybrid model with an accuracy of 63%. The standalone CNN model achieved an accuracy of 61.33%, while the LSTM model performed the least effectively with an accuracy of 61.14%.

BERT Model (65.5%):

- BERT excels due to its pretrained transformer-based architecture, which captures deep semantic meanings and contextual relationships in text. This enables it to leverage its vast understanding of language nuances, making it the most effective for this sentiment classification task.

CNN+LSTM Hybrid Model (63%):

- Combining CNN and LSTM leverages the strengths of both:
    - CNN efficiently extracts local features or patterns from the text.
    - LSTM captures sequential dependencies and contextual information.
- This synergy explains why the hybrid approach outperformed standalone models.

CNN Model (61.33%):

- The CNN model performs well in identifying local patterns and short-term dependencies in text. However, it lacks the ability to capture long-term dependencies, limiting its performance on complex language tasks.

LSTM Model (61.14%):

- While LSTM is adept at modeling sequential dependencies, it struggles with learning robust local patterns that are crucial for shorter text or less sequentially dependent features. This limitation likely caused it to underperform compared to CNN and hybrid models.

Overall, the performance hierarchy reflects each model's strengths and limitations in balancing feature extraction, contextual understanding, and generalization for sentiment analysis.