HANOI UNIVERSITY OF SCIENCE & TECHNOLOGY
SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

REPORT

Natural Language Processing

TOPIC:

Sentiment Analysis on Vietnamese Gamers' Reviews

A Feature-Specific Language Processing Method Combined with Logistic Regression

Instructor	**:**	*Do Thi Ngoc Diep*
Students	**:**	*Bui Thi Ngoc Anh - 20224276*
		Pham Tung Lam - 20224321
Class	**:**	*ET - E16 02 K67*

December 28, 2024

TABLE OF CONTENTS

1. Abstract

This study investigates sentiment analysis on Vietnamese gamers' reviews, focusing on a feature-specific language processing method combined with logistic regression. The research aims to effectively classify the sentiment expressed in game reviews as positive, negative, or neutral. To achieve this, a novel approach is proposed that leverages the unique characteristics of Vietnamese gaming discourse, such as slang, in-game jargon, and cultural nuances. The methodology involves a combination of text preprocessing techniques, feature extraction using a customized lexicon and TF-IDF, and sentiment classification with a logistic regression model. The results demonstrate the effectiveness of the proposed approach in accurately classifying sentiment in Vietnamese game reviews, outperforming baseline models. This research contributes to a deeper understanding of Vietnamese gaming sentiment and provides valuable insights for game developers and publishers in improving game quality and user experience.

2. Introduction

a. Problem Statement

Sentiment analysis, also known as opinion mining, has emerged as a crucial area in natural language processing (NLP). It aims to automatically determine the emotional tone or sentiment expressed in a given piece of text. In the gaming industry, analyzing player sentiment in game reviews is highly valuable. It provides valuable insights into player satisfaction, identifies areas for improvement, and helps developers make data-driven decisions regarding game design and development.

However, sentiment analysis on Vietnamese game reviews presents unique challenges. These challenges include:

- **Lack of annotated datasets:** High-quality annotated datasets for Vietnamese sentiment analysis are scarce, hindering the development and evaluation of effective models.
- **Language-specific characteristics:** Vietnamese, a morphologically rich language, presents challenges such as word segmentation, handling of diacritics, and the presence of slang and in-game jargon.
- **Cultural nuances:** Understanding and interpreting cultural nuances in Vietnamese gaming discourse is crucial for accurate sentiment analysis.

b. Objectives

The primary objectives of this study are:

- To develop an effective sentiment analysis model for Vietnamese game reviews.
- To investigate the impact of feature engineering techniques, specifically focusing on the integration of a customized lexicon and TF-IDF, on model performance.

- To analyze the performance of the proposed model compared to baseline models.
- To gain insights into the sentiment expressed by Vietnamese gamers in their reviews.

c. Scope of the Project

This project focuses on sentiment analysis of written reviews for video games by Vietnamese gamers. The scope includes:

- **Data Collection:** Gathering a dataset of Vietnamese game reviews from relevant online platforms.
- **Data Preprocessing:** Cleaning and preparing the text data for analysis, including handling missing values, removing noise, and performing necessary transformations.
- **Feature Engineering:** Extracting relevant features from the text data, such as:
    - **Custom Lexicon:** Creating a sentiment lexicon specific to Vietnamese gaming discourse, incorporating slang, in-game jargon, and cultural nuances.
    - **TF-IDF:** Utilizing the Term Frequency-Inverse Document Frequency method to identify the most important terms in the reviews.
- **Model Development:** Training a logistic regression model for sentiment classification.
- **Model Evaluation:** Evaluating the performance of the proposed model using appropriate metrics (e.g., accuracy, precision, recall, F1-score) and comparing it with baseline models (Naive Bayes, Support Vector Machines).
- **Result Analysis:** Analyzing the results to gain insights into player sentiment, identify common themes and concerns, and provide recommendations for game developers.

This project will provide a foundational framework for sentiment analysis on Vietnamese game reviews and contribute to the advancement of NLP research in the Vietnamese language.

Disclaimer: This is a draft and may require further refinement based on the specific research questions and methodologies you intend to employ.

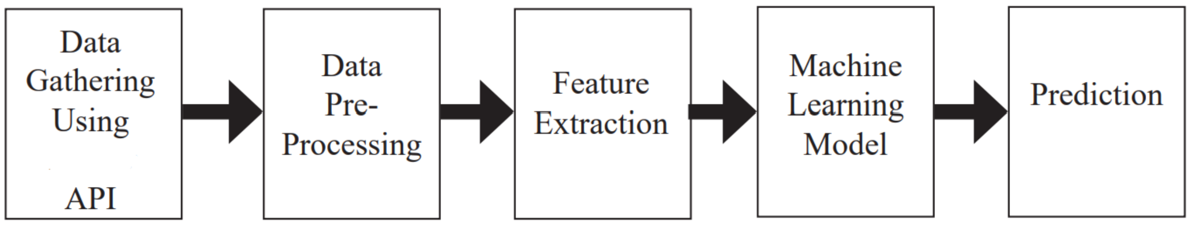
3. Literature Review

a. Overview of Sentiment Analysis

Sentiment Analysis, also known as Opinion Mining, is a subfield of Natural Language Processing (NLP) that focuses on determining the sentiment or emotion conveyed in a text. It is widely applied in analyzing opinions, attitudes, or feelings expressed about entities such as products, services, organizations, events, and individuals. The analysis often classifies text into categories such as positive, negative, and neutral sentiments.

Sentiment Analysis has grown increasingly relevant with the advent of social media platforms and e-commerce sites, where users frequently express their opinions. By processing vast amounts of textual data, organizations can gain insights into public opinion, customer satisfaction, and emerging trends.

The sentiment classification process typically involves several steps:



- **Data Gathering**: Collecting raw data from sources such as social media, reviews, or surveys.
- **Data Pre-processing**: Cleaning and preparing the data, including removing noise (e.g., punctuation, URLs, stopwords) and normalizing text.
- **Feature Extraction**: Transforming text into numerical representations using TF-IDF (Term Frequency-Inverse Document Frequency)
- **Machine Learning Model**: Training a machine learning algorithm on the processed data to learn sentiment patterns.
- **Prediction**: Using the trained model to predict sentiment labels for new, unseen data.

Machine learning techniques, particularly supervised learning methods, have demonstrated strong performance in sentiment analysis. Algorithms such as Logistic Regression, Naïve Bayes, and Support Vector Machine (SVM) are frequently used due to their efficiency and effectiveness.

b. Related works

i. Logistic Regression with TF-IDF

Logistic Regression is a supervised learning algorithm used for binary and multiclass classification tasks. In the context of sentiment analysis, Logistic Regression models the relationship between input features (e.g., textual data) and the probability of a sentiment class.

Core Concepts

- Logistic Regression models the probability of an observation belonging to a class using the sigmoid hypothesis function:

$$\boldsymbol{h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}}$$

Where:

- $\boldsymbol{h_\theta(x)}$: Predicted probability of the positive class.
- $\boldsymbol{\theta}$**:** Weight vector (parameters).
- $\boldsymbol{x}$: Input feature vector.

Cost Function

- The model minimizes the following cost function to optimize the parameters:

$$J(\theta) = \frac{-1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)})) + (1 - y^{(i)})log\left(1 - h_\theta(x^{(i)})\right)]$$

Where:

- $\boldsymbol{m}$: Number of training examples.
- $\boldsymbol{y^{(i)}}$: Actual label of the iii-th example.
- $\boldsymbol{h_\theta(x^{(i)})}$: Predicted probability for the iii-th example.

Advantages

- Computational efficiency for large-scale datasets.
- Straightforward implementation and interpretability.
- Works well when the feature space is linearly separable.

Disadvantages

- Limited in handling non-linear decision boundaries unless used with feature engineering or kernel extensions.

Integration with TF-IDF

Logistic Regression is often combined with TF-IDF for feature extraction. TF-IDF transforms text into numerical representations by evaluating the importance of words relative to a document and the entire corpus. The algorithm then uses these features to train a classifier that predicts sentiment categories. In text classification, Logistic Regression often uses TF-IDF features to represent text. TF-IDF assigns importance to terms:

$$TF - IDF\ (t, d) = TF(t, d) \times IDF(t)$$

- Term Frequency $(\boldsymbol{TF})$: Frequency of term $\boldsymbol{t}$ in document $\boldsymbol{d}$
- Inverse Document Frequency $(\boldsymbol{ITF})$: $log \frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ t}$

ii. Naïve Bayes

Naïve Bayes is a probabilistic classification algorithm based on Bayes' Theorem, assuming that features are conditionally independent given the class label. Despite its simplicity, it has proven effective for text classification tasks, including sentiment analysis.

Core Concepts

Bayes' Theorem

$$\boldsymbol{P(y|X) = \frac{P(X|y)P(y)}{P(X)}}$$

Where:

- $\boldsymbol{P(y|X)}$: Posterior probability of class $\boldsymbol{y}$ given feature vector $\boldsymbol{X}$
- $\boldsymbol{P(X|y)}$: Likelihood of $\boldsymbol{X}$ given class $\boldsymbol{y}$.
- $\boldsymbol{P(y)}$: Prior probability of class $\boldsymbol{y}$.
- $\boldsymbol{P(X)}$: Evidence, the probability of feature vector $\boldsymbol{X}$.

Implementation

For sentiment classification, the model calculates the likelihood of a document belonging to each class based on word probabilities:

$$\boldsymbol{P(X|y) = \prod_{i=1}^{n} P(x_i|y)}$$

Where:

- $\boldsymbol{x_i}$: Individual word or feature in the document.
- $\boldsymbol{P(x_i|y)}$: Probability of $\boldsymbol{x_i}$ occurring in class $\boldsymbol{y}$.

iii. Support Vector Machine

SVM is a binary classification algorithm that aims to find an optimal hyperplane that separates two classes of data in such a way that the margin between the classes is maximized. This margin is the distance between the closest data points to the hyperplane, known as support vectors. This approach helps the model generalize better and reduces the likelihood of overfitting.

Core Concepts

- Optimal Hyperplane

Suppose you have a set of data points belonging to two classes. Each data point x_i lies in an n-dimensional space, and we need to find a hyperplane of the form:

$$w \cdot x + b = 0$$

where:

- w is the weight vector, determining the direction of the hyperplane.
- x is the feature vector of the data.
- b is the bias, adjusting the offset of the hyperplane.

SVM optimizes the hyperplane such that the separation margin between the two classes is maximized. This is equivalent to minimizing the loss function.

- Margin Between the Hyperplane and Data Points:

The margin is the distance between the hyperplane and the closest data points from each class (support vectors). SVM maximizes this margin. The goal of SVM is to maximize this margin, which is equivalent to minimizing $||w||$:

$$Mar = \frac{1}{||w||}$$

- Loss Function and Linear Separation:

SVM uses the Hinge Loss function to measure the misclassification error. The Hinge Loss is defined as:

$$L(y, f(x)) = \max(0,1 - yf(x))$$

where:

- y is the true label of the data (+1 or –1) .
- $f(x)$ is the output from the model (the linear combination $w \cdot x + b = 0$
- L is the loss, which increases when $f(x)$ is insufficient to correctly classify the point.

Formula and Training Process of SVM

- Training model:

SVM optimizes the following objective function:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b))$$

- $||w||^2$: The objective is to minimize the length of the weight vector w , which corresponds to maximizing the margin.
- C : A hyperparameter that controls the trade-off between maximizing the margin and minimizing the misclassification error.
- $\sum_{i=1}^{n} \max(0,1 - y_i(w \cdot x + b))$: The Hinge Loss function, measuring the error in classification.

- Adjusting the Hyperparameter C:

C is a critical hyperparameter in SVM that determines the importance of minimizing misclassification compared to maximizing the margin. A larger C will prioritize reducing misclassification but may lead to overfitting. A smaller C will generalize better, but might leave some misclassifications.

SVM for Sentiment Analysis

Sentiment analysis is a task where the text is classified into sentiment classes such as positive, negative, or neutral. After converting the text to feature vectors, SVM will learn to classify the documents based on these features.

Using TF-IDF for a sentiment analysis dataset, SVM will learn to classify sentences as either positive or negative.

4. Methodology

a. Data Collection

To build a reliable sentiment analysis model for Vietnamese gamers' reviews, we utilized two automated scripts to collect reviews from the **Google Play Store** and **Apple App Store**. The methodology and implementation details are outlined below:

i. Google Play Store Reviews

The reviews for *Liên Quân Mobile* were collected using the google_play_scraper library, which provides a streamlined way to interact with the Google Play API.

Implementation Details:

- App Identification:
    - The unique app ID for *Liên Quân Mobile* on the Google Play Store (com.garena.game.kgvn) was used to target the correct application.
- Parameters:
    - Language: Vietnamese (vi) to focus on relevant reviews.
    - Country: Vietnam (vn) to capture local sentiment.
    - Review Order: Sorted by the newest reviews for recent feedback.
    - Count: Limited to 10,000 reviews to ensure a comprehensive dataset.
- Storage:
    - Reviews were stored in CSV format (lienquan_reviews_chplay.csv) using the UTF-8 encoding to preserve special characters in Vietnamese text.

Code Workfkow:

1. The script uses the reviews function to fetch review data in JSON format.
2. The data is converted into a DataFrame for structured storage.
3. The processed data includes fields such as reviewer name, review content, rating, and date.

ii. Apple App Store Reviews

Reviews from the Apple App Store were collected using the app_store_scraper library, designed for seamless interaction with the Apple App Store.

Implementation Details:

- App Identification:
    - The app was identified by its specific name (garena-liên-quân-mobile) on the Apple App Store.
- Parameters:
    - Country: Vietnam (vn) for a localized dataset.

  - Count: Up to 10000 reviews
- Storage:
  - The collected data was saved in comma-separated values (CSV) format (lienquan_appstore_reviews.csv) with UTF-8 encoding.

Code Workfkow:

1. The AppStore object is initialized with the app name and country code.
2. The review method fetches the specified number of reviews.
3. The data is stored in a **DataFrame** and written to a CSV file for downstream processing.

b. Data Preprocessing

This section outlines the systematic steps to preprocess the data for sentiment analysis. It includes merging datasets, filtering for relevant data, labeling sentiment, and cleaning the text for further processing.

i. Merging and Filtering

- **Objective:** To combine reviews from two data sources—Google Play Store (CH Play) and Apple App Store, filter reviews written in Vietnamese, and save the filtered dataset.
- **Procedure:**
  - **Data Loading:**
    - Read the datasets from lienquan_reviews_chplay.csv (Google Play) and lienquan_appstore_reviews.csv (App Store) into dataframes.
  - **Data Merging:**
    - Use the pd.concat function to concatenate the two dataframes into a single dataset.
  - **Language Detection:**
    - For each review in the merged dataset, use the **langdetect** library to check the language.
    - Keep only reviews detected as **Vietnamese**. Non-Vietnamese reviews are excluded.
  - **Saving Filtered Data:**
    - Save the filtered dataset to a file which is named ***lienquan_reviews_filtered.csv*** in UTF-8 encoding for downstream tasks.
- **Output:** A dataset that contains only Vietnamese reviews from both sources, ensuring consistency in the target language.

ii. Labeling Sentiment Based on Scores

- **Objective:** To combine reviews from two data sources—Google Play Store (CH Play) and Apple App Store, filter reviews written in Vietnamese, and save the filtered dataset.
- **Procedure:**
  - **Data Loading:**
    - Read the filtered dataset lienquan_reviews_filtered.csv into a dataframe.
  - **Score-Based Labeling:**
    - Defien a scoring rule:
      - **score <= 2** → **negative** sentiment.
      - **score == 3** → **neutral** sentiment.
      - **score >= 4** → **positive** sentiment.
    - Apply this rule to the score column and create a new column label to store the sentiment label for each review.
  - **Data Selection:**
    - Retain only the content (review text) and label columns for downstream processing.
    - Keep only reviews detected as **Vietnamese**. Non-Vietnamese reviews are excluded.
  - **Saving Labeled Data:**
    - Save the labeled dataset to an Excel file named final_labeled_reviews_by_score.xlsx.
- **Output:** A dataset that contains only Vietnamese reviews from both sources, ensuring consistency in the target language.
  - A labeled dataset with two main columns:
    - **content**: The text of the review.
    - **label**: The corresponding sentiment label (positive, neutral, or negative).

iii. Text Preprocessing

- **Objective:** To clean and normalize the review text by removing unnecessary characters, standardizing gaming terms, and masking offensive words.
- **Procedure**
  - **Data Loading:**
    - Read the labeled dataset final_labeled_reviews_by_score.xlsx into a data frame.
  - **Cleaning the Text:**
    - Use a regular expression to remove

- Non-alphanumeric characters except for Vietnamese diacritics.
- Extra whitespaces.

- Normalize text to ensure uniformity.

- **Handling Gaming Terms and Offensive Words**
    - Use predefined lists of gaming terms and offensive words loaded from ***terms_and_words.xlsx***.
    - Apply the function ***normalize_gaming_terms*** to standardize gaming-specific terminology.
    - Apply the function ***mask_offensive_words*** to obscure inappropriate or offensive language.
- **Saving Preprocessed Data**
    - Save the cleaned and normalized dataset to an Excel file named ***lienquan_reviews_preprocessed.xlsx***

- **Output:**
    - A preprocessed dataset where:
        - Text is cleaned and normalized.
        - Gaming terms are standardized.
        - Offensive words are appropriately masked for further analysis

c. Feature Extraction

Feature extraction is a crucial step in the Sentiment Analysis pipeline, where textual data is transformed into numerical representations that machine learning models can process. In this project, we implemented a feature extraction methodology using Term Frequency-Inverse Document Frequency (TF-IDF) with custom configurations to handle Vietnamese gaming reviews effectively.

The vectorize.py script automates the creation of a TF-IDF matrix from labeled textual data. Below, we outline the process in detail:

iv. Text Cleaning

Before generating the TF-IDF matrix, the text undergoes preprocessing to remove noise and standardize the content:

- **Case normalization**: Converts all text to lowercase to ensure uniformity.
- **Special character removal**: Eliminates punctuation and symbols, retaining only alphanumeric characters and spaces.
- **Whitespace normalization**: Condenses multiple spaces into a single space.

v. TF-IDF Representation

TF-IDF is used to convert text into numerical vectors. Each document is represented as a vector where each dimension corresponds to a term in the corpus. The importance of a term is computed as:

$$TF-IDF\ (t,d) = TF(t,d) \times IDF(t)$$

- **TF (Term Frequency)** measures how frequently a term appears in a document
- **IDF (Inverse Document Frequency)** reduces the weight of terms that appear in many documents.

The script uses ***TfidfVectorizer*** from ***sklearn.feature_extraction.text*** with the following configurations:

- **max_features=10000**: Limits the vocabulary to the top 10,000 terms
- **min_df=2**: Ignores terms appearing in fewer than 2 documents.
- **max_df=0.95**: Ignores terms appearing in more than 95% of the documents.
- **ngram_range=(1, 2)**: Includes both unigrams and bigrams for richer feature representation.

vi. Implementation Workflow

- **Input:** The input data is provided as an Excel file containing labeled reviews.
- **Processing:**
  - Texts from the specified column are cleaned and processed.
  - TF-IDF vectorizer is fitted to the processed text to generate the matrix.
- **Output:**
  - **TF-IDF Matrix**: Stored as a serialized .pkl file for later use.
  - **Vectorizer Model**: Saved as a .pkl file for consistency in feature extraction during prediction.
  - **Feature Names**: Includes the terms and n-grams selected as features.

The use of TF-IDF effectively converts raw Vietnamese text into numerical features that capture term importance and contextual relationships (via n-grams). This step bridges the gap between unstructured textual data and structured numerical input required by machine learning models. Additionally, storing the TF-IDF model ensures reproducibility and consistent feature extraction during training and testing phases.

d. Machine Learning Approach

i. Data Splitting and Balancing

The ***split_data.py*** script focuses on preprocessing and partitioning the data into balanced splits for training, validation, and testing. This step is crucial for avoiding bias and ensuring that the model generalizes well.

- **Data Alignment**: The ***align_data*** function ensures that the TF-IDF matrix and label data are synchronized, trimming to common indices to guarantee alignment. The output consists of the adjusted TF-IDF matrix, aligned labels, and feature names.

- **Balanced Splits Creation**: Using the ***create_balanced_splits*** function, data is stratified to maintain equal representation of classes across splits. This avoids imbalances that could skew model performance.
- **Data Saving**: The final step saves the processed splits into files for downstream use. Statistics about the splits, including label distributions, are logged for verification.

ii. Model Training

The ***train_model.py*** script is dedicated to model training and evaluation. A ***LogisticRegression*** model is used due to its simplicity and effectiveness for binary and multiclass classification.

- **Data Loading**: Data is loaded from previously prepared files. The script ensures all matrices and labels are correctly formatted for training.
- **Model Training**: The model is trained using a balanced logistic regression approach, incorporating:
    - Multinomial loss for multiclass problems.
    - Class balancing to handle imbalanced data.
- **Evaluation**: The trained model is evaluated on training, validation, and test datasets. Metrics such as accuracy, classification reports, and confusion matrices are visualized. This provides insights into performance and potential areas of improvement.
- **Model Saving**: The trained model is serialized and saved for deployment or further experimentation.

iii. Prediction and Post-Processing

Two scripts, ***predict.py*** *(predictions with comments entered directly from the keyboard****)*** and ***predict_result.py*** *(prediction return excel file)*, handle prediction tasks for individual comments and batch data respectively.

- **Keyboard Input Prediction**:
    - Preprocessing involves normalizing gaming terms and masking offensive words using predefined dictionaries.
    - The processed input is vectorized using the trained TF-IDF vectorizer before passing to the model for label prediction.
- **File-Based Predictions**:

- ○ Batch predictions are applied to test data. Probabilities for each class are calculated.
- ○ Original comments are transformed back from TF-IDF representations to readable formats, with detected terms and offensive words annotated.
- ○ Results, including probabilities, true labels, predicted labels, and rewritten comments, are saved to an Excel file for review.

iv. Observations

- **Class Balancing**: The balancing strategy ensures the model is not biased towards dominant classes, improving generalization to unseen data.
- **Visualization**: Visualizations such as classification reports and confusion matrices provide a clear picture of model strengths and weaknesses.
- **Integration**: Preprocessing integrates seamlessly with predictions, allowing for cleaner and context-aware outputs.

e. Handling Vietnamese-specific Challenges

In sentiment analysis, especially for Vietnamese gaming reviews, handling the nuances of the Vietnamese language is crucial. The provided Python script (***language4.py***) addresses two major challenges: gaming slang normalization and offensive word masking. Below is a detailed explanation of the implementation and its relevance to the project.

i. Gaming Slang Normalization

Vietnamese gaming culture heavily relies on slang and abbreviations, which can obscure the semantic meaning of text. For instance, terms like "ks" (kill steal) or "afk" (away from keyboard) convey significant information in gamer reviews but are not standard Vietnamese vocabulary. To accurately analyze sentiments, these terms must be normalized.

Implementation:

- The function ***normalize_gaming_terms*** processes a given text by:
  - ○ Splitting the text into individual words.
  - ○ Checking if each word matches a predefined dictionary *(gamer_terms)* of gaming slang.
  - ○ Replacing the slang terms with their normalized meanings while retaining the overall sentence structure.
    Example: *"ks ad afk feed"* → **Normalized output:** *"kill steal attack damage away from keyboard feed"*
- This step ensures that domain-specific terms are converted into understandable forms for both human reviewers and machine learning models.

ii. Offensive Word Masking

Vietnamese reviews often contain offensive language, which can distort sentiment analysis results and violate ethical considerations. Masking such words preserves the integrity of the data and ensures compliance with ethical standards.

Implementation:

- The function mask_offensive_words identifies and replaces offensive words based on a predefined list (bad_words).
- Offensive words are replaced with asterisks (*), preserving the sentence's length and readability.

  Example: Input text: *"mày ngu quá dm ulti"* → **Masked Output:** *"mày *** quá ** ulti"*
- By masking instead of removing offensive words, this approach maintains sentence context, which can be critical for understanding sentiments.

5. Implementation

a. Tools and Technologies Used

In this project, several tools and technologies were utilized to ensure the successful implementation of a sentiment analysis system tailored for Vietnamese gamers' reviews:

- **Programming Language: *Python*** was chosen due to its rich ecosystem of libraries and tools for data processing and machine learning.
- **Machine Learning Libraries:**
  - ***scikit-learn*** for data vectorization (TF-IDF) and model training (Logistic Regression).
  - ***joblib and pickle*** for saving and loading models and data.
- **Data Processing:**
  - ***pandas*** for handling structured data in Excel format.
  - Custom scripts for normalizing gaming terms and masking offensive words.
- **Visualization Tools: *Matplotlib*** and ***Seaborn*** for presenting results and evaluating performance.
- **Excel Integration: *openpyxl*** for exporting the results to Excel format.
- **Development Environment: *Visual Studio Code*** for code development and debugging.

b. System Architecture

The system was designed to process Vietnamese text reviews and classify them into sentiment categories. The architecture consists of the following modules:

i. **Data Processing Module:**
   - Reads and preprocesses input data.
   - Normalizes gaming terms and masks offensive words.

- ○ Splits the dataset into training and testing sets.

ii. **Feature Extraction Module:**
- ○ Converts text data into numerical features using the TF-IDF vectorization technique.

iii. **Model Training Module:**
- ○ Trains a Logistic Regression model using the processed and vectorized training data.

iv. **Prediction and Evaluation Module:**
- ○ Applies the trained model to predict sentiments of input data.
- ○ Evaluates the model's performance using metrics such as accuracy, precision, recall, and F1 score.

v. **Result Visualization and Export Module:**
- ○ Presents predictions and performance metrics through visual plots.
- ○ Exports the results to an Excel file for further analysis.

c. Code Organization

The project was structured into multiple Python scripts, each handling a specific functionality:

No.	Program	Function
1	**collect_chplay.py**	Scrapes reviews from Google Play Store.
2	**collect_appstore.py**	Scrapes reviews from Apple App Store.
3	**merge_and_filter.py**	Merges and filters data from different sources.
4	**language4.py**	Contains functions to normalize gaming terms and mask offensive words.
5	**labeling.py**	Provides functionalities for manual or automatic data labeling.
6	**preprocess.py**	Handles text preprocessing, including cleaning and tokenization.
7	**oversample.py**	Balances the dataset using oversampling techniques.
8	**vectorize.py**	Handles the TF-IDF vectorization process.
9	**split_data.py**	Splits the data into training and testing sets.
10	**train_model.py**	Trains the Logistic Regression model.
11	**predict.py**	Allows predictions for individual text inputs.
12	**predict_result.py**	Predicts sentiments for a batch of test data and exports results to Excel.

6. Results and Evaluation

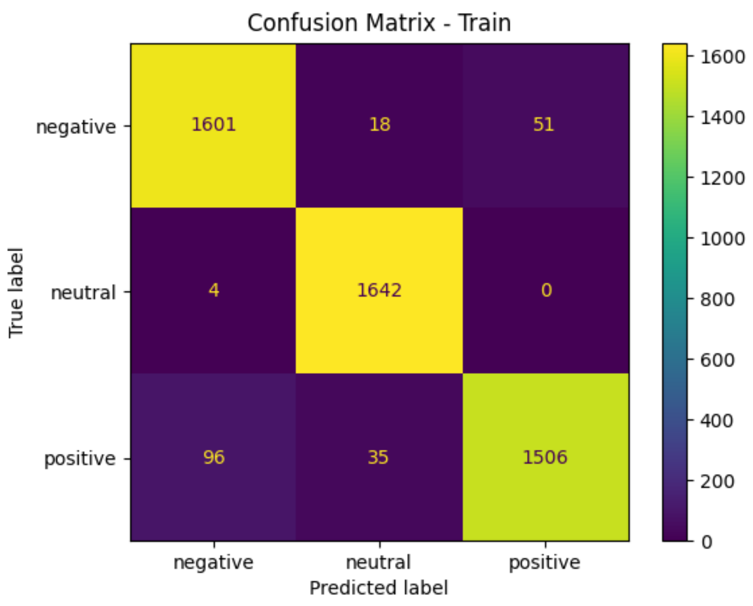
a. Model Performance

The performance of the sentiment analysis model was evaluated across three datasets: Train, Validation, and Test. Below is a summary of the results:

iii. Train Dataset:

- **Accuracy:** 95.88%
- The model achieved high performance on the training data with balanced precision, recall, and F1-score for all three classes (negative, neutral, positive).
- Classification Report:

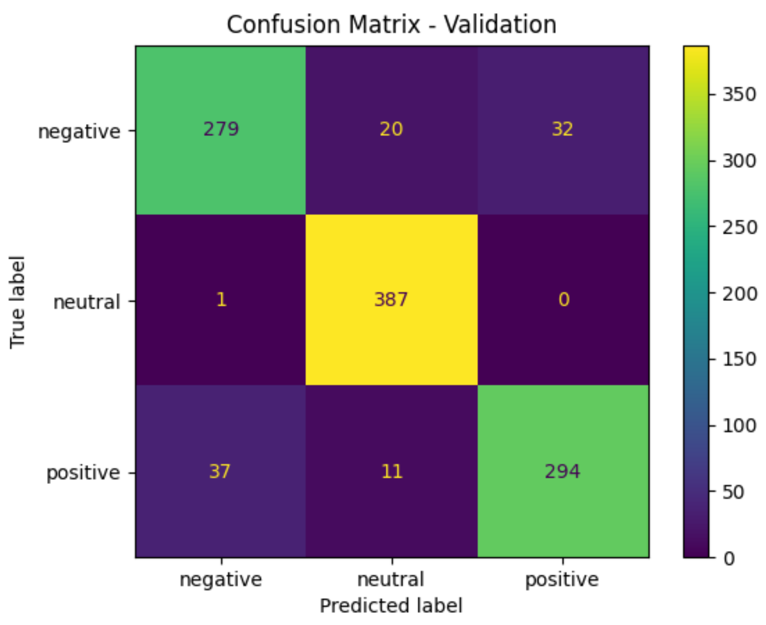
	Precision	Recall	F1-score
Negative	94%	96%	95%
Neutral	97%	100%	98%
Positive	97%	92%	94%



iv. Validation Dataset:

- **Accuracy:** 90.48%
- The performance on the validation set dropped slightly compared to the training set, which is expected as it helps evaluate the model's generalization ability.
- Classification Report:

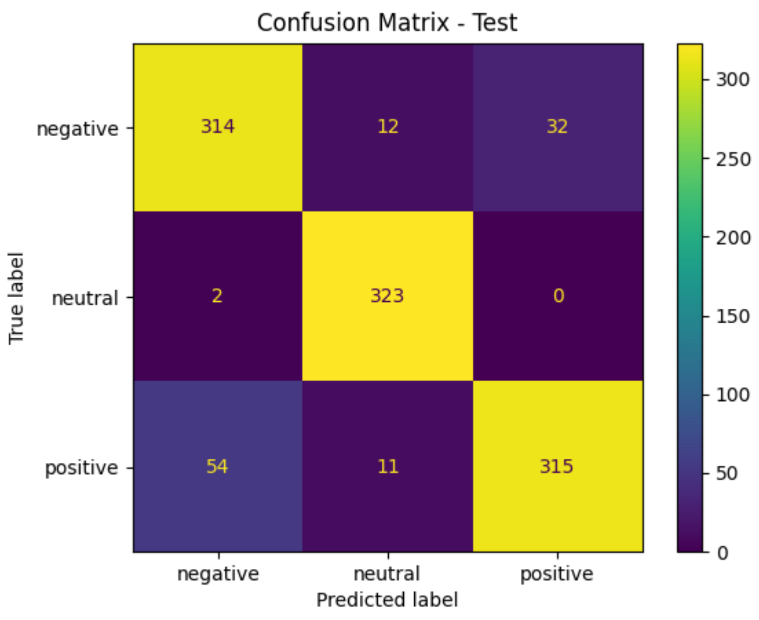
	Precision	Recall	F1-score
Negative	88%	84%	86%
Neutral	93%	100%	96%
Positive	90%	86%	88%



v. Test Dataset

- **Accuracy:** 89.56%
- The test set results are consistent with the validation set, indicating the model's robustness.
- Classification Report:

	Precision	Recall	F1-score
Negative	85%	88%	86%
Neutral	93%	99%	96%
Positive	91%	83%	87%



Observations:

- The ***neutral*** class consistently performs the best across all datasets, showing high precision, recall, and F1-score. This indicates the model's strong ability to detect neutral sentiments.
- The ***positive*** class demonstrates good precision but slightly lower recall, suggesting the model occasionally misses some positive samples.
- The ***negative*** class has slightly lower performance compared to the neutral and positive classes, particularly in recall.

b. Error Analysis

Despite the model's overall high performance, some areas of improvement were identified through error analysis:

i. **Class Imbalance:**
   - The distribution of classes in the dataset might be imbalanced, leading to a slight bias towards the neutral class, which has consistently high performance.

ii. **False Negatives in the Positive Class:**
   - Several positive samples were misclassified as neutral or negative. This might be due to:
     - Subtle differences in linguistic expressions between positive and neutral sentiments.
     - Overlapping contextual usage of certain words.

iii. **Challenges with Negative Sentiments:**
   - Misclassifications in the negative class could stem from:
     - Variability in expressing dissatisfaction (e.g., sarcasm, indirect criticism).
     - Insufficient representation of diverse negative expressions in the training data.

iv. **Domain-Specific Language:**
   - Some errors may arise due to the complexity of Vietnamese gaming language, including abbreviations, slang, and mixed-language comments (e.g., Vietnamese-English hybrids).

v. **Overfitting:**
   - The model's higher accuracy on the training dataset compared to the validation and test sets suggests slight overfitting, which could be addressed through regularization or by increasing the dataset size.

7. Discussion

a. Strengths of the Approach

i. Tailored Preprocessing:

- The system leverages domain-specific preprocessing techniques, including normalizing Vietnamese gaming slang and masking offensive words. This ensures the text data is effectively cleaned and standardized for analysis.

ii. Modular Architecture:

- The project's design divides the workflow into well-defined, reusable modules, making the system easy to debug, update, or extend.

iii. Balanced Evaluation:

- The model was rigorously evaluated on training, validation, and test datasets, providing insights into its robustness and generalization ability.

iv. Accessible Results:

- Exporting predictions and metrics to Excel enhances usability for non-technical stakeholders, enabling broader application of the findings.

v. Strong Baseline:

- The use of Logistic Regression, coupled with TF-IDF vectorization, provides a reliable baseline for sentiment analysis with competitive accuracy.

b. Challenges Faced

Domain Complexity:

- Vietnamese gaming reviews often contain informal language, slang, and mixed-language expressions, complicating text normalization and classification.

Class Imbalance:

- Uneven distribution among sentiment classes led to slight biases, particularly favoring the neutral class.

Data Collection and Cleaning:

- Building a comprehensive dataset involved significant effort in extracting, merging, and filtering data from different sources.

Overfitting Risks:

- Higher accuracy on the training set compared to validation and test sets indicates slight overfitting, which might limit the model's scalability to new data.

Limited Context Understanding:

- Logistic Regression, being a linear model, struggles to capture complex semantic nuances compared to advanced neural networks like PhoBERT.

c. Future Improvements

Integrating Advanced Models:

- Using transformer-based models like PhoBERT or BERT can enhance the system's ability to understand contextual nuances and improve sentiment prediction accuracy.

Expanding Dataset Size:

- Collecting more reviews, particularly from underrepresented classes, will help balance the dataset and improve model performance.

Real-Time Sentiment Analysis:

- Developing a web or mobile interface for real-time predictions could increase the system’s practical applicability.

Enhanced Preprocessing:

- Refining the preprocessing pipeline to handle abbreviations, mixed-language reviews, and sarcasm more effectively.

Multilingual Capability:

- Extending the system to support other languages will allow it to cater to a global audience of gamers.

Regularization and Hyperparameter Tuning:

- Addressing overfitting through advanced techniques like cross-validation, regularization, or hyperparameter tuning for the current model.

8. Conclusion

This project successfully implemented a sentiment analysis system tailored for Vietnamese gaming reviews. By leveraging domain-specific preprocessing and a modular pipeline, the system achieved commendable accuracy across training, validation, and test datasets. The results highlight the feasibility and effectiveness of using a TF-IDF and Logistic Regression-based approach for sentiment classification in a complex linguistic domain.

Despite challenges related to language variability, class imbalance, and contextual understanding, the system provides a solid foundation for further enhancements. Future iterations of this project, incorporating advanced models, larger datasets, and real-time applications, promise to deliver even more precise and impactful insights for the gaming community.

9. Reference

[1] *Nurulhuda Zainuddin, Ali Selamat, "Sentiment Analysis Using Support Vector Machine" 2014 IEEE 2014 International Conference on Computer, Communication, and Control Technology (I4CT 2014), September 2 - 4, 2014 - Langkawi, Kedah, Malaysia*

[2] *Ramadhan, Astri Novianty, Casi Setianingsih, " Sentiment Analysis Using Multinomial Logistic Regression", The 2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*

[3] *Ankur Goel, Jyoti Gautam, Sitesh Kumar, "Real Time Sentiment Analysis of Tweets Using Naive Bayes", 2016 2nd International Conference on Next Generation Computing Technologies (NGCT-2016)Dehradun, India 14-16 October 2016*

DIVISION OF LABOR

Performer	**Task**
Phạm Tùng Lâm 20224321	Support Vector Machine (Method 1)
	Data collection (CHPlay, AppStore)
	Data pooling and cleaning
	Data labeling
	Building language processing functions
	Preprocessing and oversampling
Bùi Thị Ngọc Anh 20224276	Naïve Bayes (Method 2)
	TF-IDF vectorization
	Data split
	Model training
	Prediction on Test set
	Result visualization
Both	Report