# University of Science and Technology of Ha Noi

Natural Language Processing

# Final Project's Tutorial Report

Vietnamese News Classification

| **Student Name** | **Student ID** |
|---|---|
| Bui Dinh Lam | 22BI13234 |
| Le Linh Long | 22BI13262 |
| Nguyen Tuan Khai | 22BI13202 |
| Nguyen Quang Huy | 22BI13195 |
| Pham Thai Son | 22BI13397 |
| Nguyen Hai Dang | 22BI13073 |

**Lecturer in Charge:**

Dr. Nghiem Thi Phuong

**Submission Date: 26/03/2025**

**Abstract**

In this study, we present a novel approach to Vietnamese text classification, focused primarily on an advanced preprocessing pipeline. By refining text normalization, tokenization, and feature extraction methods, our methods were able to improve upon the result of previous studies [2, 8] , demonstrating the efficacy of our enhancements. We used traditional algorithms such as Support Vector Machine and Gradient Boosting algorithms, while also implementing a simple Multilayer Perceptron architecture of our own devising. The results are very promising, both in terms of accuracy and computational efficiency.

***Keywords*** — Text classification, Natural Language Processing, Machine Learning

## 1 Introduction

### 1.1 Project's goals

In today's society, Vietnamese media is rapidly growing, with countless online news sources generating massive amounts of content daily. A machine learning model capable of automatically categorizing news articles becomes crucial not just for media organizations, but for citizens looking for specific types of news in a timely manner. This method can help readers find the needed information faster while also helping media companies in content management and trend analysis.

Furthermore, reliable classification models could provide valuable information for researchers and businesses in analyzing media trends, understanding public opinions, and identifying emerging social topics.

### 1.2 Dataset

Our research utilized a dataset that we gathered from an academic paper [3]. The dataset got its content from four major news distributors that are fairly popular in Vietnam: VnExpress, TuoiTre Online, Thanh Nien Online, and Nguoi Lao Dong Online.

The data collection process is fairly lengthy to ensure high-quality, clean, and accurate class labels. Researchers used web crawling techniques to extract articles, followed by a cleaning process that removed HTML tags and normalizing spelling. This step was crucial to standardize the text and remove any formatting or structural noise that could interfere with model training. To guarantee the highest level of accuracy, five linguist manually reviewed and corrected the documents, carefully verifying that each article was correctly classified

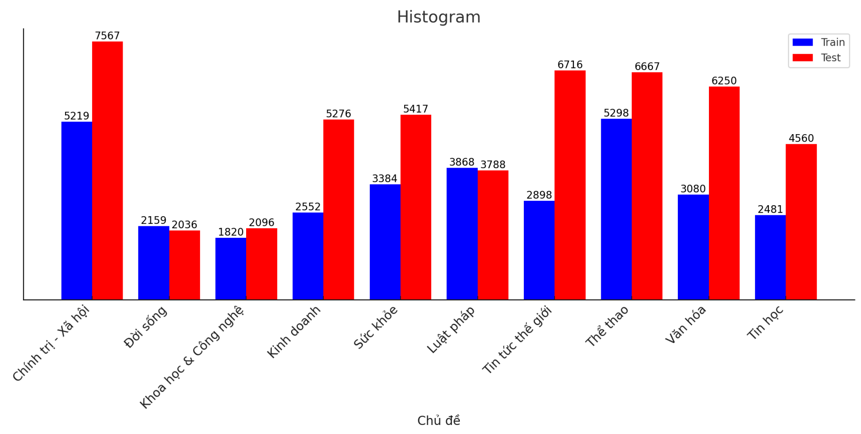

Figure 1: Histogram of document types distribution across train and test sets

The figure 1 showed the frequency of the classes in both the training and test set within the original dataset. Furthermore the train set included 33,000 documents while the test set has 50,373 documents of various classes. Our project's source code can be found here https://github.com/longle0702/nlp-final-usth as well as our Dataset

## 2 Preprocessing

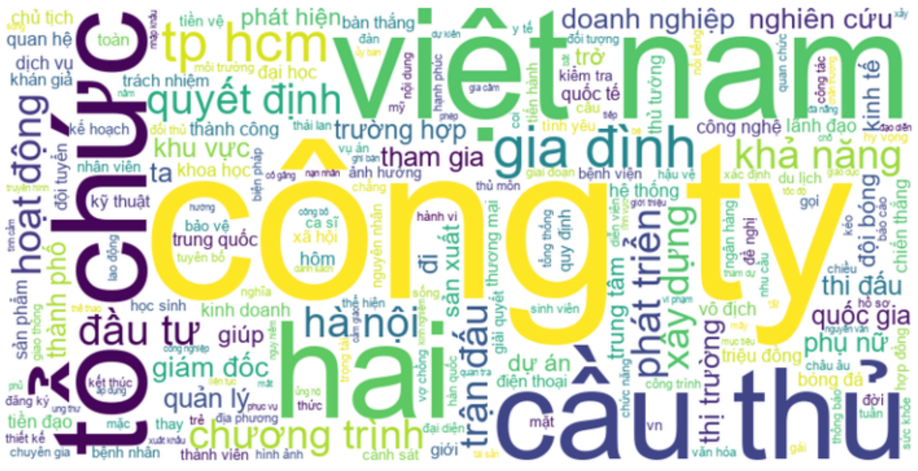

Figure 2: Most frequently occurring words found in articles

One of the most prominent challenges presents itself in the complexity of the Vietnamese language, which varies in terms of compound words (wherein spaces do not always represent word boundaries), tonal distinctions, and most notably diacritics. Unlike English, Vietnamese is a morpho-syllabic language, meaning each syllable in a word corresponds to the smallest unit of language (morpheme) and each syllable is usually a meaningful unit by itself. However, as [2] described, there are many cases where different segmentation of words can carry different meanings due to the presents of compound and fortuitous concurrence words, which can make it very inconsistent at times. Thus, we need to devise a special preprocessing pipeline to handle the textual data, which will be the crux of our methodology.

### 2.1 Preprocessing pipeline

#### 2.1.1 Text standardization

This is the foundational step of our preprocessing pipeline, especially for Vietnamese where the morphology and orthography can introduce significant variability. The main goal of this step is to transform the input text into a consistent and uniform format prior to subsequent steps. This involves multiple sub-steps addressing the textual noise and inconsistency in our documents.

**Unicode Normalization** Essentially, this process handles the diacritics in the Vietnamese language (e.g. high rising tone ´, low falling tone ‘, and vowel marks such as â and ă). In Unicode, a single character can be represented either as a precomposed character or a combination of base characters and diacritical marks. These variations mean that different representations of the same character will result in distinct tokens, causing problems in text processing. To address this, Unicode normalization converts all characters into a single standardized form (dubbed Normalization Form C or NFC) which combines characters and their respective diacritic variants into a unified representation [7]. This step helps ensure that all characters are consistent and that typographical variations will not cause issues down the line.

| Original Text | Normalized Form C |
|---|---|
| Tiếng Việt | T i ế n g V i ệ t |

**Diacritical Normalization** The presence of tonal marks and vowel signs have significant effect on the meaning of a word, and variations in typing conventions including differences in the input method as well as redundant and inconsistent diacritical marks can lead to multiple representation of the same word. For example, the word ’công’ can be typed in several ways that are sometimes slightly different and sometimes altogether incorrect (e.g. cóng, which contains the high rising tone; or cong which has no diacritical mark). To mitigate potential inconsistencies, the diacritical normalization step applies specific rules to map non-standard diacritical combinations to their standard form, thereby eliminating typographical inaccuracy.

**Algorithm 1** Diacritical Normalization

```
Require: Input word $W$
Ensure: Normalized word $W_{norm}$
  for each character $c$ in word $W$ do
      if character $c$ is non-standard diacritic then
          Replace $c$ with its normalized equivalent
      end if
  end for
      return $W_{norm}$
```

**Lowercasing** Simply put, this step involves changing all uppercase letters present in the text to their lowercase counterparts, since the case of characters does not carry any semantic significance to the text itself. The presence of capitalization rules does not affect the core meaning of a given word, therefore we normalize them in this sense to reduce vocabulary space.

**Removal of Extraneous Characters** These characters, which include numbers, punctuation marks, and special symbols that either are irrelevant to the meaning of the word or can be of detriment to the classification task. They do not meaningfully contribute to the identification of the topic of a document, and in the context of our task they only introduce noise and blur the word boundaries. To address this, we elected to remove all non-alphanumeric characters from the text - except for spaces - which allows us to focus on the words themselves, and clear the text for vectorization [6].

### 2.1.2 Linguistic Tokenization and Stopword Removal

**Tokenization** This step extracts the most meaningful features from the text and transforms them into a format suitable for vectorization later on. This is where we split the raw document into discrete units, or tokens, which often constitutes individual words or sub-words. In languages like English where spaces represent clear word boundaries with only a few exceptions, usually relegated to spoken dialect, the task of tokenization is very straightforward. However, for languages like Vietnamese, tokenization can be more complex as the spaces between mono-syllabic components do not represent delimiters between words (such as in cases of compound words). The morpho-syllabic nature of the Vietnamese lexicon, coupled with absence of clear word boundaries, presents challenges that the tokenization algorithm must be able to account for [4]. For this, we used the *ViTokenizer* module, specifically designed for segmenting Vietnamese text. It leverages a rule-based approach combined with statistical models to segment text, able to handle compound words as well as multi-syllabic units and differences in meaning that arise from tone or diacritical marks.

- The most notable advantage *ViTokenizer* has over existing methods for Vietnamese in particular, is that it relies on an extensive lexical dictionary of words and phrases, which would be immediately identified as tokens. For example, the term "công an" would be identified as a single token, even though it consists of two syllables, because it is a common compound word in the Vietnamese language.
- For words that are not found in the dictionary, *ViTokenizer* uses the maximum matching (greedy) algorithm to find the longest possible match. It continuously searches for shorter word matches until a match is found, where the word is treated as a token and removed from the input text.
- Handling uncommon compound words that are not readily available in the dictionary presents another unique challenge, as they cannot be found relying on a single entry. In such cases *ViTokenizer* is also capable of combining syllables based on learned models to accurately identify compound units based on contextual inference. This involves statistical methods like bigram [5] and trigram language models, learned from a large annotated corpus of Vietnamese texts, to segment unfamiliar phrases into tokens that have a high probability of carrying semantic significance.

**Stopword Removal** This step eliminates commonly occurring words, which in themselves do not carry significant meaning. These words are often referred to as "function words", and include articles, prepositions, conjunctions, and pronouns (e.g, "và", "là", "vâng", etc.). This process can be formalized as follows:

Let $T = \{t_1, t_2, ..., t_n\}$ represent the set of tokens obtained from the previous tokenization step, and $S = \{s_1, s_2 ... s_n\}$ as the set of stopwords. The stopword removal function $f : T \to T'$ operates by filtering elements of $S$ from $T$:

$$S(T) = \{Token_i | Token_i \notin S\}$$

where the new set of tokens $T'$ contains only meaningful words, allowing the model to learn the more salient features of the text for more accurate predictions.

**Algorithm 2** Tokenization Process for Vietnamese Text

```
Require: Input text $T$
Ensure: Tokenized text $T_{\text{segmented}}$
1: $T_{\text{cleaned}} \leftarrow$ Clean_Text($T$)    ▷ Pre-process text (remove punctuation, digits, etc.)
2: $T_{\text{normalized}} \leftarrow$ Normalize_Text($T_{\text{cleaned}}$)    ▷ Normalize diacritics, lowercase, etc.
3: $T_{\text{segmented}} \leftarrow$ ViTokenizer.tokenize($T_{\text{normalized}}$)    ▷ Tokenize using ViTokenizer
4: for each token $t$ in $T_{\text{segmented}}$ do
5:     if $t \in$ Stopwords then    ▷ Check if the token is a stopword
6:         Remove $t$ from $T_{\text{segmented}}$
7:     end if
8: end for
9: return $T_{\text{segmented}}$    ▷ Return the tokenized text
```

### 2.1.3 TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization transforms the processed textual data into numerical feature vectors for our machine learning/deep learning models. This technique is based on the intuition that terms that appear frequently in one particular class of documents but infrequently across all documents are more significant and should be attributed higher weights. TF-IDF consists of two components: Term Frequency (TF) and Inverse Document Frequency (IDF). The TF for a given term $t$ in document $d$ is calculated as:

$$TF(t,d) = \frac{\text{Frequency of term t in document d}}{|\text{Terms in d}|}$$

The IDF for a term $t$ is calculated as:

$$IDF(t) = \log\left(\frac{N}{DF(t)}\right)$$

where $N$ is the total number of documents in the corpus and $DF(t)$ is the number of documents that contain the term $t$. With both components calculated, the TF-IDF score is the product of TF and IDF:

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

This metric is used to represent the importance of terms within a document in relation to the entire corpus. Once the TF-IDF has been computed for every word, we choose to keep the top 2500 tokens with the highest TF-IDF value. This step eliminates the need for feature selection steps by limiting to only the most relevant terms, while also reducing dimensionality for computational efficiency.

To handle the imbalance present in our training data, we implement a data augmentation technique called SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic examples of the minority class. Developed by Chawla et al. [1], the process involves generating new training data points via interpolation between two existing instances with the same ground truth label.

**Algorithm 3** SMOTE

```
1: Input: Minority class samples, number of desired synthetic instances , number of nearest
   neighbors (K)
2: Compute Euclidean distance between each minority class sample and all other minority
   class samples to identify the K nearest neighbors for each sample
3: For each sample: Randomly select 1 of K neighbors. Compute the difference vector between
   the minority class sample and the neighbor. Multiply the distance by a random $x \in [0,1]$.
   Add the resulting vector to the minority class sample to create new synthetic instance.
4: Repeat step 3 N times for N synthetic instances.
5: Output: Original minority class now has N synthetic instances.
```

## 2.2 Data analysis post-vectorization

After performing a simple TF-IDF analysis with the dataset, we performed the following visualizations on our dataset for more insights:

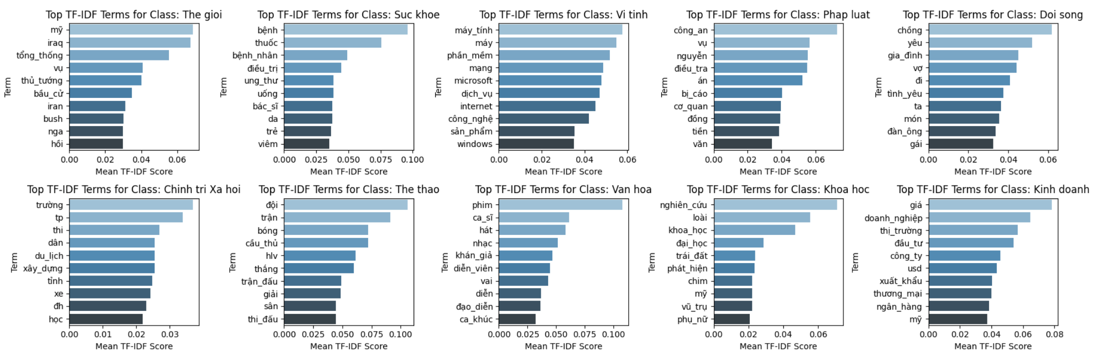

Figure 3: Mean TF-IDF score for different class

From the figure 3, it can be seen that certain words appear much more often in particular classes. Keywords are well-separated, with business terms appearing in the Business category and medical terms appearing in Health, indicating clear topic distinction. The Life category includes more personal and relationship-related words, while Technology and Science are focused on research and digital advancements. Overall, the TF-IDF scores effectively capture key terms for each category and accurately reflect the topic of the article.

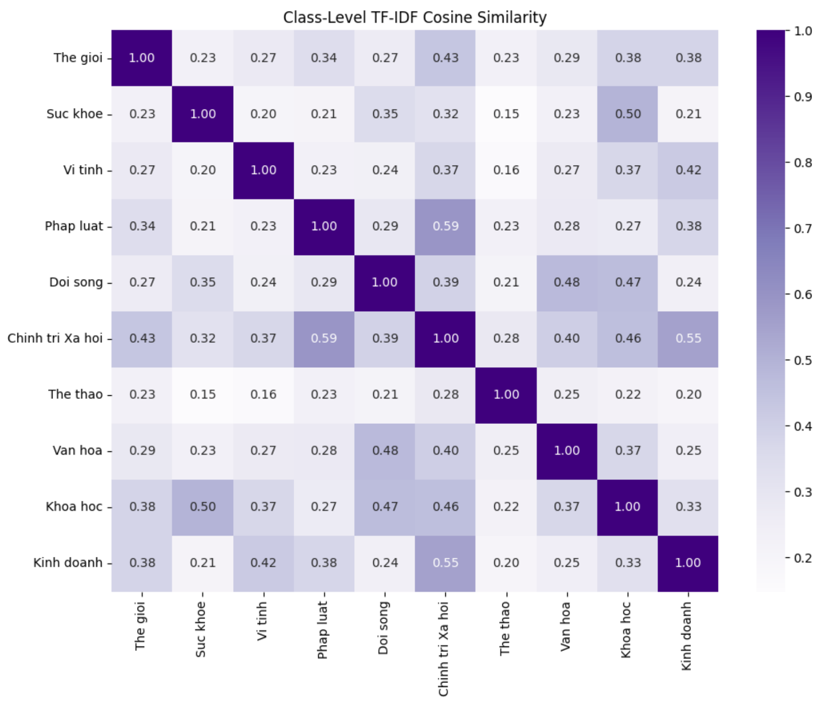

Figure 4: Cosine similarity of different classes

The heatmap 4 shows the cosine similarity between different categories based on their TF-IDF scores. Topics like ”Chinh tri xa hoi” and ”Phap luat” had a higher cosine similarity score, indicating that these two topics are related. Some categories, like The thao (Sports), show lower similarity with others, showing their more domain-oriented vocabulary. The visualization effectively highlights which topics share overlapping terms and which remain more independent.

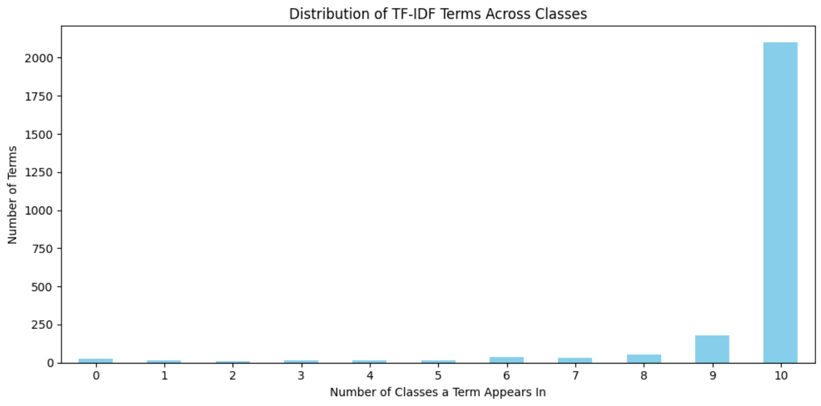

Figure 5: Distribution of TF-IDF Terms across classes

The histogram 5 illustrated the distribution of TD-IDF terms across different classes. This chart shows that a significant proportion of the vocabulary is shared among every category.

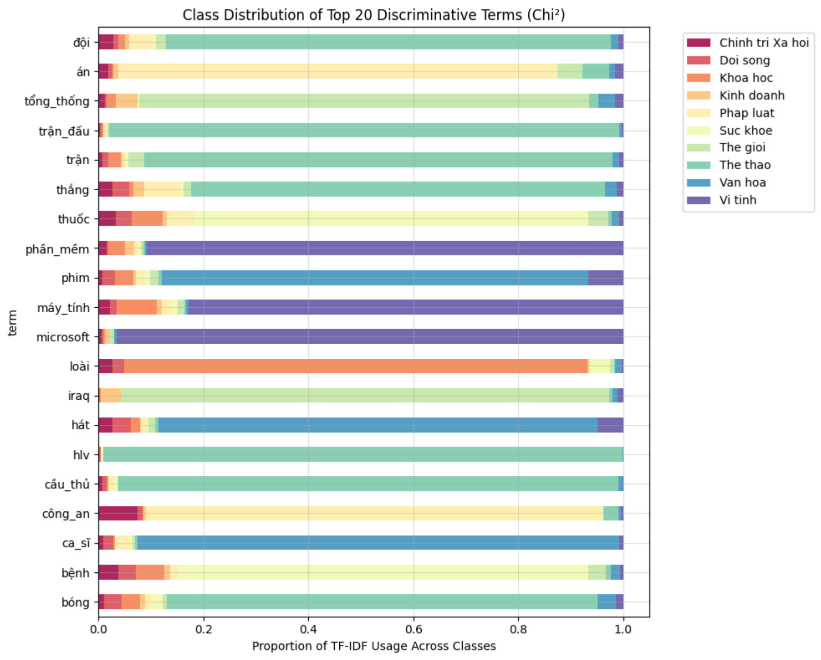

Figure 6: Top 20 most discriminative terms according to Chi-Squared Score

The stacked bar chart 6 visualizes the distribution of the top 20 most discriminative terms across different categories using the Chi-squared statistic. Each term is associated with a specific class, indicating strong topic relevance. For example, technology-related words like microsoft and phan mem (software) are mostly linked to Vi tính, while sports terms like doi and cau thu are concentrated more in The thao (Sports). Some overlap exists, but overall, the terms effectively distinguish between categories.

## 3 Machine Learning Approach

### 3.1 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical technique used for dimensionality reduction. It transforms a large set of variables into a smaller one while retaining most of the original information.

Due to hardware constraints, we must implement PCA to reduce the number of features, which in turn decreases computational cost. Given our dataset with 2,500 features, directly processing such high-dimensional data would be computationally expensive.

To balance dimensionality reduction and information retention, we set the cumulative explained variance ratio threshold to 95%. This ensures that the selected principal components capture the majority of the dataset's variance.

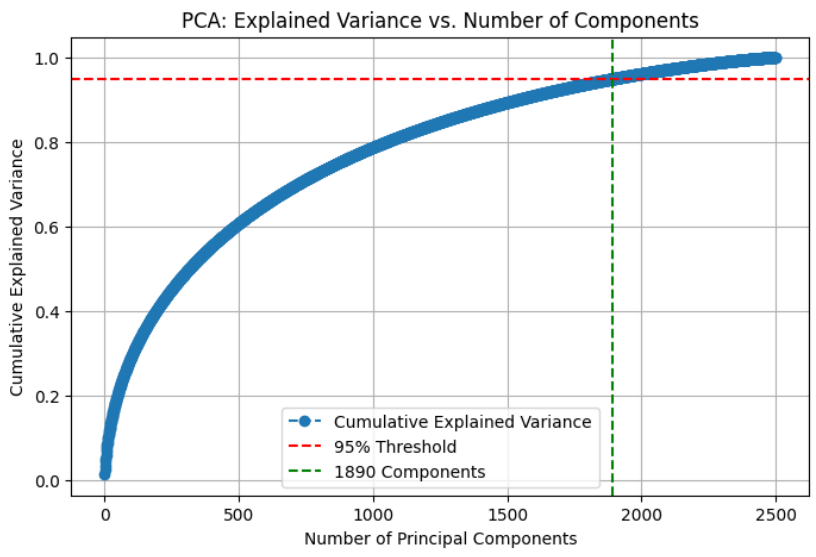

Figure 7: Explained Variance vs Number of Components

### 3.2 Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification tasks. It works by finding an optimal hyperplane that best separates data points of different classes while maximizing the margin between them. The data points closest to this hyperplane, called support vectors, define the decision boundary. If the data isn't linearly separable, SVM can use kernel functions to transform it into a higher-dimensional space where separation is possible.

To find the best model performance, we use GridSearchCV, a technique that systematically searches through different combinations of hyperparameters to find the optimal settings. We tune parameters such as the regularization strength, the choice of kernel (linear or RBF), and the gamma value. This helps improve classification accuracy by selecting the best-performing configuration for our dataset.

### 3.3 Light Gradient Boosted Machine

LightGBM is a gradient boosting framework built upon the GBDT (Gradient Boosting Decision Tree) model. LightGBM improves the concept of ensemble learning to enhance the model's overall performance by combining the prediction results of multiple weak classifiers. Each weak classifier is trained on the residual errors of the previous one, allowing the model to learn new data features in each iteration and refine its predictions. LightGBM offers efficient parallel training with faster speed, reduced memory consumption, improved accuracy, distributed computing support, and the ability of processing large-scale data quickly.

### 3.4 Extreme Gradient Boosting

Extreme Gradient Boosting is a model built based on the gradient boosting framework. The model builds an array of weak decision tree learners sequentially. Each trees tries to learn based on the error of the previous trees. The overall mechanism of the model can be generalized by the following equation:

$$f_t(x) = f_{t-1} + h(t)$$

Where $h(t)$ is the new tree that tries to correct the error of the previous trees and reduce the residuals. Our model uses regularization techniques like $L_1$ and $L_2$ to prevent overfitting. For this algorithm, we also used GridSearchCV to find the best possible hyperparameters so that the model can perform reliably post-training.

## 4 Deep learning approach

The neural network receives a set of input data, denoted by matrix $X$, whose rows represent the sample set and whose columns represent the feature space of each of those samples. The weights $W$ are numerical values associated with the connections between neurons. They determine the strength of these connections and, in turn, the influence that one neuron's output has on another neuron's input. The model calculates the dot product of $X$ and $W$, then feeds the result into an activation function $f(.)$. This function produces non-linearity into the model as well as limits the output in a specific range. This, in turn, ensures that the network is able to learn complex patterns while maintaining gradient stability. The operation is expressed as followed.

$$A = f(W \cdot X + b)$$

For the task of multiclass classification, the final layer is designed such that it contains N neurons, where N is the total number of categories in our data. We obtain the final output probabilities by passing the weighted sum through the softmax activation function.

$$softmax(Z_i) = \frac{e^{Z_i}}{\sum_{j=1}^{N} e^{Z_j}}$$

This function makes sure that the model produces a probability distribution for all classes and that the class with the highest prediction score is selected as the final prediction.

In our approach, we devised a 6-layer perceptron model that uses the ReLU activation function, with the final layer applying softmax. The categorical cross-entropy loss was employed for this model. We define the true label (one-hot encoded) as $y_i$ and predicted probability of class i as $\hat{y}_i$. Categorical cross-entropy punishes predictions based on how confident the model is about the correct class.

$$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

We trained our model for 100 epochs with batch size of 70. We also split the data into train and validation set with the ratio of 80-20 and employed the Adam algorithm for the optimisation process.

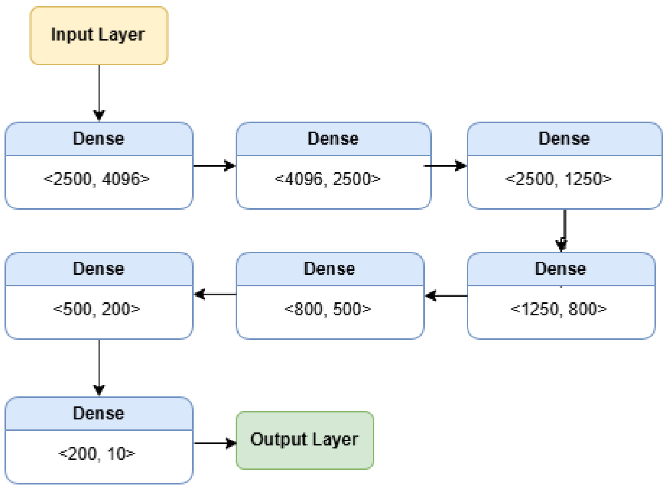

Figure 8: Our proposed method

## 5 Results

In this section, we show the result that we obtain using the proposed pipeline. We utilize the popular classification metrics, including weighted precision, recall, F1 score, accuracy, and balanced accuracy. Weighted precision is calculated by:

$$Precision_{weighted} = \sum_{c \in C} \frac{s_c}{N} * Precision_c$$

where $C$ is the set of classes, $s_c$ is support level of class $c$ which equal to the number of sample in class $c$, $N$ is the total number of samples, $Precision_c$ is the precision when consider $c$ the positive class. Weighted recall is calculated similarly.

Table 1 compares the results between different models.

| | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| SVM | 0.92 | 0.92 | 0.92 | 0.92 |
| LBGM | 0.90 | 0.90 | 0.90 | 0.90 |
| XGB | 0.89 | 0.89 | 0.89 | 0.89 |
| Proposed method | **0.963** | **0.962** | **0.962** | **0.962** |

Table 1: Comparison of different models

Table 2 shows the comparison of our methods with previous works [2, 8]

| Method | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| [2] | - | - | - | 0.9375 |
| [8] | 0.9 | 0.89 | 0.897 | - |
| Ours | **0.963** | **0.962** | **0.962** | **0.962** |

Table 2: Comparison to related works

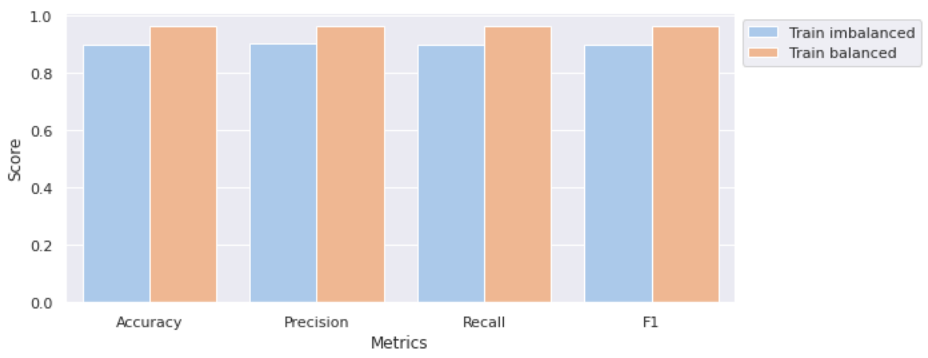

Figure 9: The results of our model when trained on the original data, versus the balanced data

## 6 Conclusion

In conclusion, this study presented a comprehensive approach to Vietnamese news article classification that emphasized effective preprocessing. By handling the unique challenges of the Vietnamese language such as compound words and diacritical marks, we were able to ensure data quality and consistency that showed significant improvement to our results across all metrics. This research has contributed valuable insights into the Vietnamese natural language processing subfield as a whole, and lays the groundwork for further improvements in the future. With greater data availability and hardware capacity, we theorize that it may be possible to achieve significantly better results, but our work here has already shown great promise.

## References

[1] Nitesh V. Chawla, Kevyn W. Bowyer, Lawrence Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. In *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1–7, 2002.

[2] N.T. Hien, B.T. Thoa, and L.N.H. Hoa. A study on deep learning for vietnamese text classification. *Journal of Military Science and Technology*, 95:85–94, 2024.

[3] Vu Cong Duy Hoang, Dien Dinh, Nguyen le Nguyen, and Hung Quoc Ngo. A comparative study on vietnamese text classification methods. In *2007 IEEE International Conference on Research, Innovation and Vision for the Future*, pages 267–273, 2007.

[4] H.P. Le, T.M.H. Nguyen, A. Roussanaly, and T.V. Ho. A hybrid approach to word segmentation of vietnamese texts. *Text, Speech and Dialogue*, 6233:191–198, 2010.

[5] D.Q. Nguyen, D.Q. Nguyen, T. Vu, M. Dras, and M. Johnson. A fast and accurate vietnamese word segmenter. *arXiv preprint arXiv:1709.06307*, 2017.

[6] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[7] Unicode Consortium. Unicode® standard, version 13.0, 2020.

[8] To Nguyen Phuoc Vinh and Ha Hoang Kha. Vietnamese news articles classification using neural networks. *Journal of Advances in Information Technology*, 12(4):363–369, 2021.