FPT UNIVERSITY

FPT UNIVERSITY

# Using Contrastive Learning for Aspect Detection with Vietnamese Dataset

**Lê Phước Cường, Trịnh Hoàng Nam**

Supervised by: **Bùi Văn Hiệu**

**Major: Artificial Intelligence**

**Hà Nội, August 2023**

# ACKNOWLEDGMENTS

We - *Lê Phước Cường*, *Trịnh Hoàng Nam* would like to express our sincere gratitude to the professors and lecturers at the FPT University Hanoi for their dedicated guidance and teaching throughout our study at the university, which has equipped us with the knowledge we have today.

We would like to extend our deep appreciation to Dr. *Bùi Văn Hiệu*, who has provided invaluable guidance, instructions, and shared a wealth of knowledge and experience throughout the process of conducting this thesis.

Lastly, we would like to thank our family, friends, and colleagues, who have always been by our side during difficult times, offering support, guidance, and encouragement, enabling us to complete this graduation thesis.

Once again, we sincerely thank everyone involved.

Hanoi, ngày 10 tháng 9 năm 2023

*Lê Phước Cường*
*Trịnh Hoàng Nam*

# PROTESTATION

We hereby declare that all the presented results in this thesis are accomplished under the guidance of Dr. *Bùi Văn Hiệu*. All the referenced materials and related research used in this graduation thesis are properly cited in the reference section. This graduation thesis does not involve any plagiarism or replication of others' works but represents our own accomplishments.

Hanoi, ngày 10 tháng 9 năm 2023

*Lê Phước Cường*
*Trịnh Hoàng Nam*

# ABSTRACT

Customer reviews for a business are extremely important because they provide valuable insights, and impact the customer's decision-making process and the customer experience after a purchase. Determining the aspect of the reviews, which makes business easier to analyze, is a difficult problem because each review has a different writing style, many grammatical errors, and often contains acronyms. Unsupervised aspect detection (UAD) strives to automatically extract understandable facets and pinpoint segments that are specific to these facets from online reviews. However, because of the difference between syllables and accents in Vietnamese, automatically extracting aspects in reviews is a challenging task. To address aspect detection issues in the context of Vietnamese text, in this thesis, we will propose an approach that combines contrastive learning with aspect detection. Specifically, we generate aspects for similar word clusters followed by the model is encouraged to differentiate between them and capture distinctive features by measuring the similarity between pairs of samples in the Vietnamese dataset. This enables the model to generate high-quality representations for aspects and their corresponding text segments. Furthermore, this thesis builds upon the experimental methods used by previous studies such as Smooth self-attention (SSA) and High-resolution selective mapping (HRSMap) to further enhance the performance of aspect detection in the context of Vietnamese text. We achieved relatively good results for the 4 *"golden"* aspects, averaging around 0.73 for *F1-score*.

# Contents

**Section 1 Introduction** . . . . . **1**

**Section 2 Related Work** . . . . . **2**

**Section 3 Theoretical Basis** . . . . . **4**

3.1 Sentiment classification problems . . . . . 4

3.2 Word embedding models . . . . . 5

3.2.1 Static word embedding . . . . . 5

3.2.2 Dynamic word embedding . . . . . 7

3.3 Clustering methods . . . . . 12

3.3.1 K-Means clustering . . . . . 12

3.3.2 DBSCAN . . . . . 14

3.4 Contrastive learning method . . . . . 16

3.4.1 Contrastive loss . . . . . 16

3.4.2 Some of the contrastive loss . . . . . 16

**Section 4 The proposed method** . . . . . **19**

4.1 Overall method workflow . . . . . 19

4.2 Clustering model . . . . . 20

4.3 Contrastive learning . . . . . 21

4.4 Mapping model outputs to labels . . . . . 21

**Section 5 Experiments** . . . . . **24**

5.1 Dataset . . . . . 24

5.2 Experiments method . . . . . 25

5.2.1 Data overview . . . . . 26

5.2.2 Data processing . . . . . 26

5.2.3 Specify golden aspect . . . . . 27

5.2.4 Data representation . . . . . 28

5.3 Enviroment and Performance measurement . . . . . 31

5.3.1 Enviroment . . . . . 31

5.3.2 Performance measurement . . . . . 31

5.4 Result . . . . . 32

5.4.1 Result of different word embedding methods . . . . . 32

5.4.2 Fine-tuning some additional parameters and factors . . . . . 40

**Section 6 Conclusion and Future work** . . . . . **42**

6.1 Conclusion . . . . . 42

6.2 Limitation and Future work . . . . . . . . . . . . . . . . . . . . . . . . . 43

# List of Figures

3.1 Architecture of Word2Vec with CBOW technique[12] . . . . . . . . . . 6
3.2 Architecture of Word2Vec with Skip N-Gram technique [6] . . . . . . . 7
3.3 Architecture of Transformer [16] . . . . . . . . . . . . . . . . . . . . . . 9
3.4 MLM Sentence Transformer [10] . . . . . . . . . . . . . . . . . . . . . . 11
3.5 Illustration of K-Means clustering algorithm [9] . . . . . . . . . . . . . 13
3.6 Illustration of DBSCAN algorithm [7] . . . . . . . . . . . . . . . . . . . 14
3.7 DBSCAN vs K-Mean Comparison [2] . . . . . . . . . . . . . . . . . . . 15
3.8 Illustrate the operation of the model using the contrastive loss function [11] 17
3.9 Illustration of triplet loss given one positive and one negative per anchor [13] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
4.1 Overall method workflow . . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.2 Illustrate how the Many-to-one mapping works . . . . . . . . . . . . . . 22
4.3 Illustrate how the High-resolution selective mapping works . . . . . . . 23
5.1 Foody Website . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
5.2 Illustrate how the High-resolution selective mapping works . . . . . . . 25
5.3 Number of each aspects . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
5.4 Graph of *distortion score* by KElbowVisualizer with number of clusters *k* 27
5.5 Visualization of aspects from KMeans clustering algorithms . . . . . . 28
5.6 Visualizing FastText model results . . . . . . . . . . . . . . . . . . . . . 34
5.7 Visualizing Word2Vec model results . . . . . . . . . . . . . . . . . . . . 37
5.8 Visualizing Word2Vec model results . . . . . . . . . . . . . . . . . . . . 39
5.9 Result of comparing 3 different word embedding methods . . . . . . . . 39
5.10 Compare the results of the model before and after tuning . . . . . . . . 41

# List of Tables

3.1 Target word and Context words example . . . . . . . . . . . . . . . . . . 7
5.1 Scores of Đồ ăn aspect - FastText . . . . . . . . . . . . . . . . . . . . . . 33
5.2 Scores of Giá cả aspect - FastText . . . . . . . . . . . . . . . . . . . . . . 33
5.3 Scores of Không gian aspect - FastText . . . . . . . . . . . . . . . . . . . 33
5.4 Scores of Phục vụ aspect - FastText . . . . . . . . . . . . . . . . . . . . . 34
5.5 Top 10 most similar words to *đầy_đủ* from 7.1GB aggregated data . . . 35
5.6 Top 10 most similar words to *đầy_đủ* from thesis data . . . . . . . . . . 35
5.7 Scores of Đồ ăn - Word2Vec . . . . . . . . . . . . . . . . . . . . . . . . 36
5.8 Scores of Giá cả - Word2Vec . . . . . . . . . . . . . . . . . . . . . . . . 36
5.9 Scores of Không gian - Word2Vec . . . . . . . . . . . . . . . . . . . . . 36
5.10 Scores of Phục vụ - Word2Vec . . . . . . . . . . . . . . . . . . . . . . . 37
5.11 Scores of Đồ ăn - PhoBert . . . . . . . . . . . . . . . . . . . . . . . . . 38
5.12 Scores of Giá cả - PhoBert . . . . . . . . . . . . . . . . . . . . . . . . . 38
5.13 Scores of Không gian - PhoBert . . . . . . . . . . . . . . . . . . . . . . 38
5.14 Scores of Phục vụ - PhoBert . . . . . . . . . . . . . . . . . . . . . . . . 38
5.15 Compare the results before and after Removing stopwords . . . . . . . . 40
5.16 Hyper-parameters list . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
5.17 Compare the results before and after Removing stopwords + Tuning hyper-parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40

## Section 1. Introduction

Currently, in the era of digital transformation, most human activities generate a vast amount of data. Especially with the development of social networks, e-commerce platforms, and knowledge-sharing websites, users' opinions, comments, and reviews about products and services are being shared more than ever before. This represents a tremendous amount of data. If we can leverage this data, particularly the comments and feedback shared, it can provide valuable insights not only to the owners of the products or services but also to potential customers who are seeking information about them. However, the efficient use of this data in machine learning models is challenging due to the lack of labeled data and the lack of selectivity. While unsupervised methods like clustering have shown limited effectiveness on this type of unlabeled data, various new approaches are being developed to address this issue.

On the other hand, the information that the systems use to analyze is only interested in the rating level, the scale from 1 star to 5 stars. However, these scales do not objectively respond to all customer opinions in written content. As a result, some systems also begin to analyze written customer reviews. For example, *"Dịch vụ spa tại salon A khá tốt, trang thiết bị hiện đại, công nghệ tiên tiến, tuy nhiên nhân viên không nhiệt tình với khách hàng."*, the analysis system will return positive. It can be seen that the above analysis cannot provide a comprehensive analysis. Considering the field of restaurants in particular, we see that to evaluate whether a restaurant is good or not needs a lot of factors, from dishes, prices, space and service, and food safety to many aspects of the restaurant industry. an area of business. Therefore, analyzing the sentence into many aspects of the sentence is an extremely necessary thing, from which it is possible to evaluate the customer's opinion through many aspects, each aspect has its own content. Since then, we need an accurate aspect analysis algorithm for customer reviews to solve this problem. That solution will bring a huge step forward in the analysis of enterprises as they can save a lot of time and analyze more selectively. They do not need to read all of these reviews but only need to look at the aspects of the review sentence after it is retrieved and from there filter the aspects that need to be evaluated.

Some current methods that can be mentioned include using rule-based approaches based on dictionaries and word relationships to determine the aspects of a comment[4]. However, this method requires extensive domain knowledge about the specific product/service being referred to, as well as human understanding. Some attempts have been made to use supervised learning methods based on pre-labeled data, which have shown better results compared to rule-based approaches, but only when a large amount of labeled data is available. In recent years, unsupervised learning methods have achieved significant results based on their own variations and changes. These methods include using Latent Dirichlet Allocation (LDA) models to determine the topics of comments [3], aspect-based auto-encoders, and their variants. However, most of these methods still rely on human knowledge from existing datasets and may encounter issues when applied to new datasets.

In our thesis, regarding the object and scope of the study, we collected customer reviews on [https://www.foody.vn], an online food ordering application. The scope of these comments is issues surrounding the restaurant, the location, the food, and the service. Regarding the objectives and challenges, we first looked at the data. This is a difficult dataset because customer reviews are written in a variety of formats, have many grammatical errors and acronyms, and are often written in regional languages. For example, *“Món này mắc quá, một cái bánh bông lan trứng 21k”*. So we researched and tested several data-cleaning methods. The second is the study of embedding methods. While the difference between Vietnamese syllables and word causes difficulties in word processing *"xe đạp"* is a compound word made up of two syllables, the noun *"xe"* and the verb *"đạp"*. The second is the study of embedding methods. The difference between Vietnamese syllables and words is also a difficult one. For example, *"cánh canh"* is a compound word made up of two syllables *"bánh"* and *"canh"* and they have different meanings. Most pre-training word embedding models are only learned at the syllable level, which can cause the model to misinterpret the meaning of the sentence. Thirdly, we will study incorporating contrast learning into aspect identification. Contrast learning[5] is an approach that depends on the similarity between terms; terms in a particular context are drawn closer together, and represented as similar, while terms related to different contexts are denoted as dissimilar, thus being pushed apart together. This helps the system to analyze the relationship between sentences and aspects easily.

## Section 2. Related Work

Aspect detection is an important problem of aspect-based sentiment analysis. It starts with Named Entity Recognition (NER) problem[1]. The three most commonly used NER systems are supervised machine learning models, rule-based approaches, and dictionary-based methods. Consequently, the limitations of entity recognition tasks often involve the requirement of labeled data with entities corresponding to keywords in the sentences. Even when using rule-based approaches, it can be resource-intensive and challenging to determine the complete set of keywords for entities, as well as encountering cases where words can be used in multiple entities.

Emotional analysis, also referred to as opinion mining or emotion AI, utilizes natural language processing (NLP) and text analysis techniques to systematically identify, extract, and quantify subjective states and information[9]. Transfer learning, on the other hand, is a field in machine learning that focuses on leveraging knowledge gained from solving one problem and applying it to another related problem. In the realm of NLP, recent research has showcased the effectiveness of models that employ pre-training for language modeling tasks. These transfer learning-based models facilitate the comprehension of word and sentence structures, enabling a better understanding of semantics and relationships.

In a study conducted by Ngoc C. Le et al [9]., they proposed the application of BERT, a transfer learning method, in the Vietnamese language to address a text classification problem known as Aspect-based Sentiment Analysis. The experiments were conducted on two datasets, namely Hotels, and Restaurants, and encompassed two specific tasks: (A) Aspect Detection and (B) Aspect Polarization. The obtained results surpassed the performance of some previous systems in terms of accuracy, recall, and F1 measurements. This highlights the effectiveness of using transfer learning, particularly BERT, in enhancing the performance of text classification tasks like Aspect-based Sentiment Analysis in the Vietnamese language. Kiet Vu Van and his friends [15] propose a Convolutional Neural Network architecture for aspect detection for Vietnamese reviews. The aspect detection is to aim to identify the entity E and attribute A pairs expressed in the text.

*Unsupervised learning* approaches do not need labeled data. One of the approaches is using an LDA-based topic model for extracting aspect [3]. This method has shown strong performance when dealing with large amounts of unlabeled data. However, it exhibits some limitations in needing human effort for aspect mapping and difficulty in finding relationships between contexts in data. Therefore, it requires a new approach to this problem and contrastive learning has come into play. By representing words in a sentence as vectors in a vector space, people observe that these vectors have the ability to describe relationships, semantic similarities, and contextual information of the data. This can be beneficial in determining the context of a sentence. And by presenting a simple contrastive learning of sentence embeddings [5], it greatly improves state-of-art sentence embeddings on semantic textual similarity tasks, which helps determine the aspect of the sentence better.

Furthermore, Tian Shi and his friends [14] propose a self-supervised contrastive learning framework and an attention-based model equipped with a novel smooth self-attention module for the UAD task in order to learn better representations for aspects and review segments. However, most of the techniques benefit from pre-trained encoders and overcome the disadvantages of data preprocessing, especially with Vietnamese text.

Therefore, this thesis focuses on unsupervised methods for aspect detection and aims to address the challenges associated with aspect learning and mapping in the context of the Vietnamese dataset.

In this research paper, our focus lies on exploring unsupervised methods for identifying aspects, and we are committed to addressing the difficulties associated with aspect comprehension and alignment and testing the loss functions of the contrast learning model.

# Section 3. Theoretical Basis

In this section, The first objective is to study and test the embedding model that best fits the data and a good clustering algorithm for mapping to aspect. We then show a self-supervised contrast learning framework while testing several loss functions for aspect detection.

## 3.1 Sentiment classification problems

Currently, sentiment classification tasks, which aim to determine the topic of a comment, are quite popular. This is especially true for their applications in classifying user feedback on specific products or in online discussions to assess users' current attitudes. The common characteristic of such tasks is that the majority of the training data used for model training is labeled data. Therefore, supervised machine learning models are constructed to extract information from user comments and make predictions for new comments.

Typically, for natural language processing tasks like the ones mentioned above, neural networks, recurrent neural networks, or upgraded models derived from them are commonly used. With the advancement of hardware and the continuous research and development of models to provide the most accurate predictions, these models often require a large amount of data to achieve the highest level of accuracy. However, this poses a significant challenge as most of the available data is unlabeled. Neglecting such a vast amount of unlabeled data would be a waste. Consequently, recent research has been focused on finding ways for models to leverage the maximum information from this unlabeled data. Contrastive learning has been applied to natural language processing tasks as a means to meet these requirements.

## 3.2 Word embedding models

Embedding textual data is the process of extracting a vector space representation from textual data, such that the vector has the ability to capture relationships, semantic similarity, and contextual information of the data.

In this vector space, words with similar meanings will be located close to each other. For example, consider the following three sentences: *"Nhân viên phục vụ ở đây rất dễ thương"*, *"Nhân viên phục vụ ở đây rất chu đáo"*, *"Đồ ăn ở đây khá đắt so với các quán xung quanh"*. In this vector space, the words *"dễ thương"* and *"chu đáo"* would be represented as vectors that are close to each other, indicating their similarity.

### 3.2.1 Static word embedding

#### TF-IDF

TF-IDF is a method for calculating the weight of words based on their frequency of occurrence in a document, in order to evaluate the importance of words in that document. However, there are certain words that appear frequently across many documents called stop words, such as *"and," "the," "in"*,..., which need to have their impact reduced. Hence, we have the following formula for calculating the weights of words in a document:

$$w_{x,y} = tf_{x,y} \times \log(\frac{N}{df_x}) \tag{1}$$

For $w$ is the weight of word $x$ in document $y$, where $tf_x$ is the term frequency in the document, $N$ is the total number of documents in the sample set, and $df_x$ is the number of documents in which $x$ appears.

#### Word2Vec

Word2Vec is an algorithm that uses a neural network model to create word representations in such a way that words used in similar contexts tend to have similar meanings and are also represented closely in the vector space.

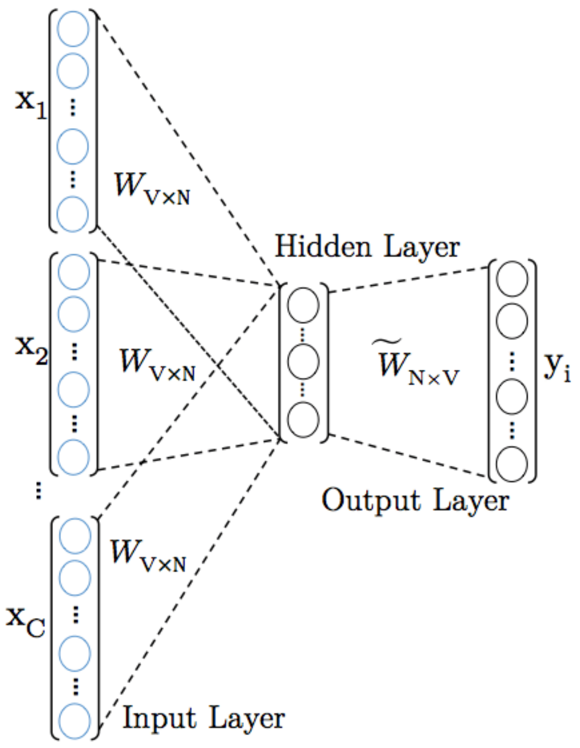
By constructing a neural network and training it on a corpus of text, the model can generate representations of words based on predicting surrounding words given a target word or predicting target words given context words in a sentence. As a result, similar words often have similar representations because they are frequently used in similar contexts.

Word2Vec employs two methods:

**CBOW (Continuous Bag of Words)**

CBOW is a model that predicts target words given the surrounding context words [3.1]. The neural network architecture of CBOW consists of three layers:

- Input layer: It consists of the surrounding context words, and the number of context words is a hyper-parameter typically determined before training.
- Hidden layer: It computes the average vector representation of all the input words to generate a feature vector.
- Output layer: It is a dense layer that applies the softmax function to predict the probabilities of the target word.

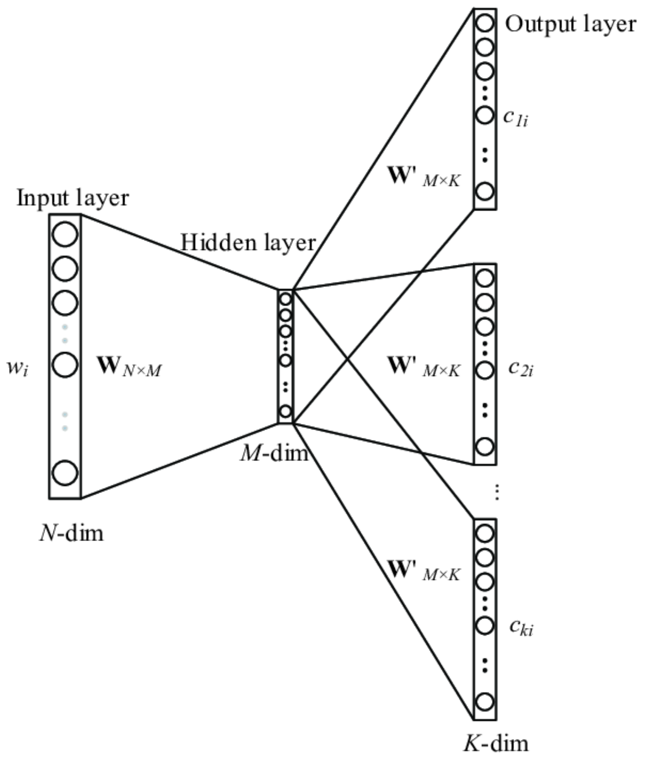


*Picture 3.1. Architecture of Word2Vec with CBOW technique[12]*

**Skip N-Gram**

With the opposite idea to CBOW, Skip N-Gram is built to predict context words based on a n given target word. [3.2]

For example, in the sentence *"Các bạn nhân viên ở đây phục vụ rất nhiệt tình, chu đáo"*, if we choose the word *"nhân viên"* as the target word, the corresponding context words could be *"các"*, *"bạn"*, *"ở"* and *"đây"*.



*Picture 3.2. Architecture of Word2Vec with Skip N-Gram technique [6]*

| Target word | Context words |
|---|---|
| *nhân viên* | *Các* |
| *nhân viên* | *bạn* |
| *nhân viên* | *ở* |
| *nhân viên* | *đây* |

*Table 3.1. Target word and Context words example*

Based on that, predicting the probability of the contextual word given the target word can be represented as follows:

$$p(\text{Các,bạn,ở,đây}|\text{"nhân viên"})$$

To simplify the computation, let’s assume that the occurrences of contextual words given the target word are independent. The formula above can be rewritten as follows:

$$p(\text{Các}|\text{"nhân viên"})p(\text{bạn}|\text{"nhân viên"})p(\text{ở}|\text{"nhân viên"})p(\text{đây}|\text{"nhân viên"})$$

### 3.2.2 Dynamic word embedding

Dynamics embeddings are a powerful technique for capturing temporal patterns in data, utilizing the capabilities of embedding methods. Contextual embeddings, such as ELMo and BERT, go beyond traditional word representations like Word2Vec and have achieved remarkable performance in various NLP applications. These embeddings assign each word a representation based on its context, allowing them to capture word usage across different situations and encode cross-linguistic knowledge. In this section, we will explore the concepts and methods associated with dynamic embeddings, delving into the underlying principles and prominent approaches used to generate them.

Transformers, introduced in the *"Attention Is All You Need"* paper by Vaswani et al., are a powerful architecture used in natural language processing tasks, including word embedding. The core idea behind transformers is the self-attention mechanism, which allows the model to focus on different parts of the input sequence when processing each token. This attention mechanism enables transformers to capture long-range dependencies and contextual information effectively, making them particularly well-suited for word embedding tasks.

The transformer architecture consists of an encoder and a decoder, but for word embedding tasks, we are primarily concerned with the encoder part. The encoder takes a sequence of input tokens (words) and transforms them into meaningful representations, also known as embeddings.

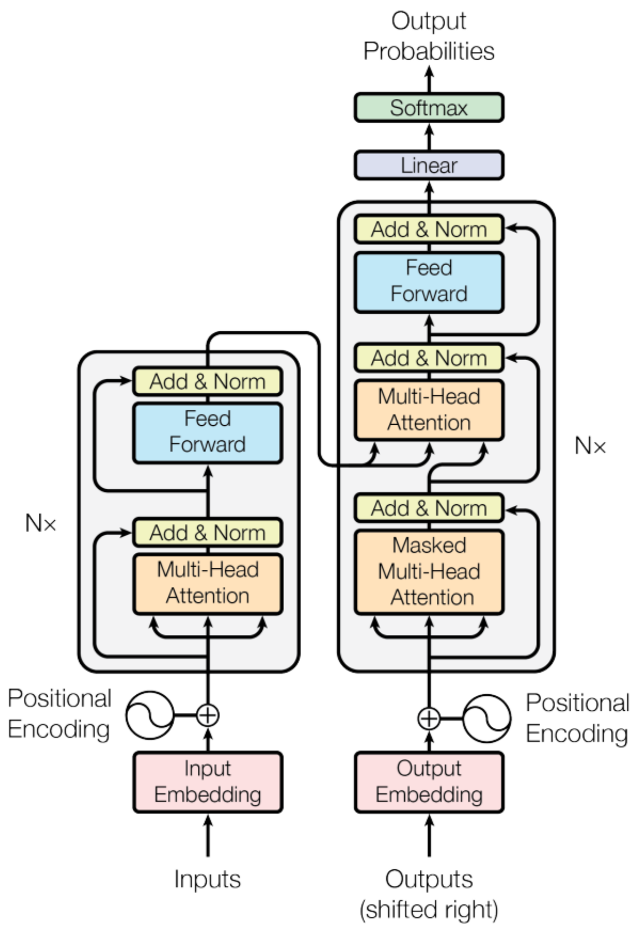
Here's how the transformer's architecture works for word embedding [3.3]:

- Tokenization: The input text is first tokenized, breaking it down into individual tokens (words or subwords). Each token is then represented using an initial embedding, which is typically a randomly initialized vector.

- Embedding Layer: The initial embeddings are passed through an embedding layer. This layer learns a dense representation for each token by mapping the tokens to continuous vector space. This mapping aims to capture semantic similarities between tokens, helping the model to understand the relationships between words.

- Self-Attention Mechanism: The heart of the transformer's power lies in its self-attention mechanism. In this mechanism, the embeddings of all tokens are processed simultaneously, and each token can attend to all other tokens in the sequence. The attention scores determine how much each token should attend to others. Tokens that are more relevant for the current token will receive higher attention weights, allowing the model to consider the context and dependencies between words.

- Feed-Forward Neural Networks: After self-attention, the output goes through feed-forward neural networks. These networks consist of fully connected layers that apply non-linear transformations to the token representations, further enhancing their expressiveness.



*Picture 3.3. Architecture of Transformer [16]*

- Layer Stacking: Transformers use multiple layers of self-attention and feed-forward neural networks. This stacking allows the model to capture different levels of abstraction and context in the embeddings. Each layer refines the embeddings based on learned information from the previous layer, resulting in increasingly informative and contextualized representations.

- Output: The final output of the transformer's encoder is a sequence of contextually rich embeddings for each token in the input sequence. These embeddings effectively capture the contextual information of the words in the text and are commonly used as word representations in various NLP tasks, including sentiment analysis, machine translation, and text generation.

In summary, transformers revolutionized word embedding by leveraging the self-attention mechanism, enabling them to efficiently model complex relationships between words and create meaningful contextual embeddings. This architecture has become the backbone of state-of-the-art NLP models and has significantly advanced the field of natural language processing.

### BERT

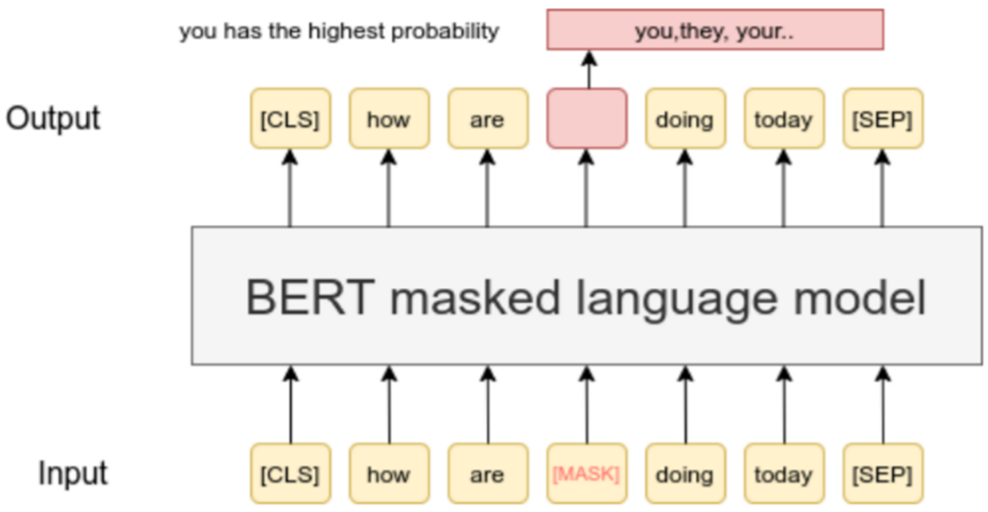
BERT, an acronym for Bidirectional Encoder Representations from Transformers, is a significant advancement in the field of natural language processing (NLP) following ELMO. It has achieved state-of-the-art results in 11 NLP tasks. BERT's main contribution lies in training word embeddings using denoising auto-encoders, rather than relying solely on language modeling. This innovative approach enables BERT to incorporate contextual information from both directions simultaneously.

The concept of denoising auto-encoders originated in computer vision and is used to train compressed representations of images. The process involves encoding an image using linear and convolutional layers and then reconstructing it. The loss function measures the sum of squared differences between the original image and the reconstructed image. By utilizing the weights of the middle layer, the image can be represented in a compact form. To enhance the generalization capability of the representation, various types of noise, such as Gaussian noise, can be added to the input image.

BERT's architecture is an expanded version of the encoder component in the Transformer model, which has been described in detail in a previous post. The BERT base model consists of $L = 12$ Transformer blocks, $A = 12$ self-attention heads, and a hidden size of dimension $H = 768$. On the other hand, the BERT large model comprises 24 Transformer blocks, 16 self-attention heads, and a hidden size of dimension 1024.

In BERT, input sequences are represented using token embeddings, segment embeddings, and position embeddings. Token embeddings involve tokenizing the input sequence and applying a specialized technique called WordPiece embedding. In English, WordPiece is not solely based on space separation but serves as a tool for sentence tokenization and subsequent embedding. The vocabulary size of WordPiece embedding is typically 30K, resulting in an embedding size of $V * H$, where $V$ represents the vocabulary size and $H$ is the hidden size (often chosen to be the same as the hidden size). Segment embeddings capture the relationship between two sentences and are only necessary for tasks involving sentence pairs. Position embeddings function similarly to their description in the Transformer model.

The BERT model consists of two main stages: pre-training and fine-tuning. Pre-training involves two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM randomly masks around 15% of the tokens in a given sequence and trains the model to predict those masked tokens based on their context. Masking can be seen as introducing noise to the input, and predicting the original values of the masked tokens is akin to decoding or recovering the original input. The model is guided to predict the actual tokens using cross-entropy loss. This process enables the model to learn weights that incorporate contextual information and enhance the representation of the masked tokens. The MLM process can be visually represented as shown in the figure below [3.4]:



*Picture 3.4. MLM Sentence Transformer [10]*

To address the discrepancy between the training and testing datasets, where the *[MASK]* token is absent during testing, BERT introduces a strategy. Among the 15% of masked tokens, 80% of the time, the actual token is replaced with *[MASK]*; 10% of the time, it is replaced with a random token; and 10% of the time, there is no alteration made to the original token. The underlying concept behind this approach is to introduce controlled variability, which is believed to enhance the model's ability to generalize and perform well on unseen data.

**PhoBERT**

PhoBERT is an adaptation of the widely known BERT (Bidirectional Encoder Representations from Transformers) model, renowned for its outstanding capabilities in diverse natural language processing undertakings. The foundation of PhoBERT's pre-training strategy draws from [8], an approach that enhances the BERT pre-training process to achieve greater resilience in performance. The training dataset, comprising 20 gigabytes of uncompressed text, exclusively consists of Vietnamese content. *VnCoreNLP* was employed to segment words and sentences. Some proofs show that the model obtains state-of-the-art results in downstream tasks for the Vietnamese dataset.

PhoBERT has two versions, *PhoBERT-base* and *PhoBERT-large* employ identical architectures to BERT-base and BERT-large, correspondingly. *PhoBERT-base* uses a batch size of 1024 across 4 V100 GPUs (16GB each) and the highest learning rate of 0.0004, and *PhoBERT-large* uses a batch size of 512 and the highest learning rate of 0.0002. *PhoBERT-base* and *PhoBERT-large* undergo 40 epochs of training (with a warm-up of the learning rate during the initial 2 epochs), leading to approximately 540K training steps for the former (computed as 13.8M × 40 / 1024) and 1.08M training steps for the latter.

## 3.3 Clustering methods

### 3.3.1 K-Means clustering

With the idea of grouping data based on their representations in vector space, K-Means clustering clusters data by grouping nearby data points into a cluster. [3.5]

Let's assume we have a dataset $X = \{X_1, X_2, \ldots, X_N\} \in R^{d\times N}$ where $N$ is the number of data points and $d$ is the vector space size of each data point. Considering $K < N$ as the number of clusters we need to divide the dataset into.

For each data point $x_i$ let $y_i = \{y_1, y_2, \ldots, y_n\}$ be its corresponding label vector. Therefore, if $x$ is assigned to cluster $p$, then $y_{ip} = 1$ and $y_{ij} = 0\ \forall i \neq j$.

We have a set $\{c_1, c_2, \ldots, c_n\}$ which represents the centroids of the $k$ clusters. From this, when a data point $x$ is assigned to cluster $c_n$, it will have an error/loss of $(x_i - c_n)$. Naturally, like other algorithms, we aim to find the cluster $c_n$ that minimizes the error/loss. Thus, the problem is formulated as the optimization function below:

$$\sum_{j=1}^{K} y_{ij}||x_i - c_j||_2^2 \tag{2}$$

On the entire dataset, the formula above can be equivalently expressed as:

$$L(Y,C) = \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}||x_i - c_j||_2^2 \tag{3}$$

To solve the problem, K-means algorithm sequentially solves two sub-problems:

- Fixing cluster centers and finding labels for data points: When the cluster centers are fixed, the problem of finding label vectors for the entire dataset can be transformed into the problem of finding labels for individual data points $x_i$.

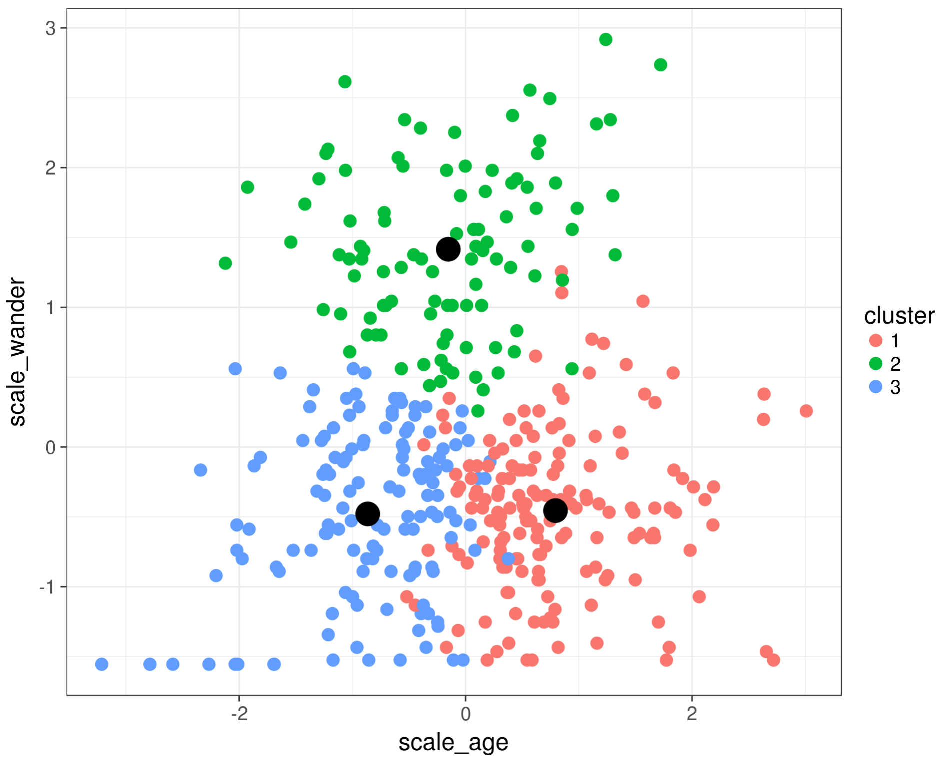
$$y_i = \arg\min_{y_i} \sum_{j=1}^{K} ||x_i - c_j||_2^2 \tag{4}$$

In this case, $j = \arg\min_j ||x_i - c_j||_2^2$ because there is only one cluster where $y_{ij} = 1$. Therefore, cluster $j$ is the nearest cluster center to the data point $x_i$ at the current time.

- Obtaining clusters for data points and updating new cluster centers: Similarly to the previous step, updating the new clusters can also transform the loss function into the following form:

$$c_i = \arg\min_{c_i} \sum_{j=1}^{K} ||x_i - c_j||_2^2 \tag{5}$$

In simple terms, the new cluster will be updated based on the average of the representative vectors of the data points within cluster $j$.

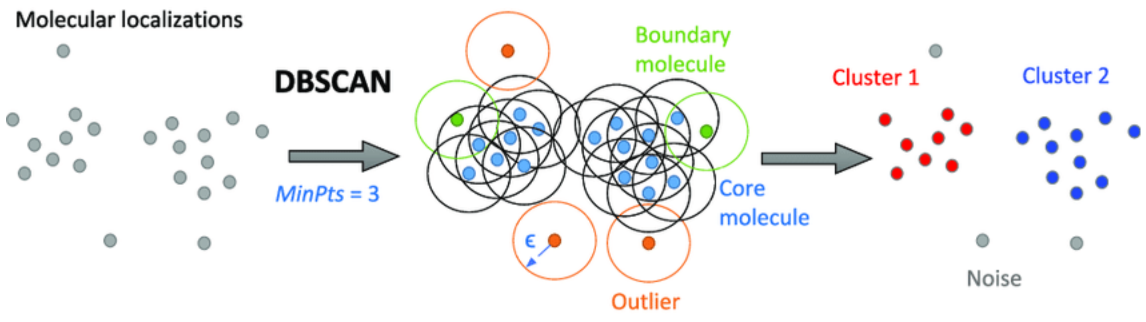


*Picture 3.5. Illustration of K-Means clustering algorithm [9]*

However, a common challenge in clustering problems in general, including K-Means, is determining the appropriate number of clusters.

### 3.3.2 DBSCAN

DBSCAN, also known as Density-Based Spatial Clustering of Applications with Noise, is a clustering algorithm that groups data points in space based on the concept of clusters as maximal sets of points with a density connection. The DBSCAN algorithm is capable of detecting clusters with different shapes and has a good ability to identify noise. [3.6]



*Picture 3.6. Illustration of DBSCAN algorithm [7]*

Based on the given datasets, we can easily determine the clusters and identify the noise values using DBSCAN. This is something that both K-Means and Hierarchical clustering can not achieve.

The underlying idea behind the DBSCAN algorithm is to determine the clusters and noise based on the characteristics of the clusters. In clusters, the data density is always higher compared to the density of noise regions. Within clusters, there is always a neighborhood radius $r$ where a minimum number of data points $k$ are reached. The determination of neighboring points is done using a distance function between points within the cluster.

Building upon this idea, DBSCAN also uses similar inputs, such as an epsilon radius $ept$ and a minimum number of points required to form a cluster $minPts$. We will now introduce some definitions:

- Neighborhood points: Given a dataset $X$, the neighborhood points of a point $p$, denoted as $N_{eps}(p)$, are determined by:

$$N_{eps}(p) = \{q \in X | dist(p,q) \leq eps\} \tag{6}$$

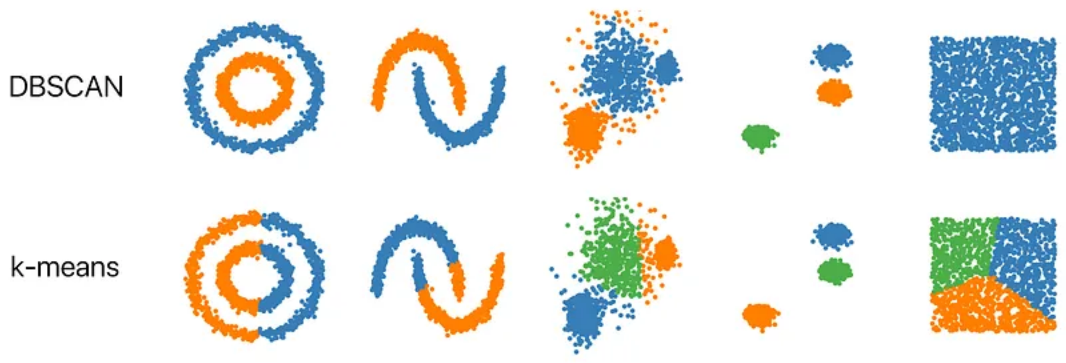
  Where $dist(p,q)$ is the distance between 2 given points $p$ and $q$.

- Direct Density - Reachable: For a point $p$ in $X$ to be considered directly density-reachable from a point $q$, the following conditions must be satisfied:

$$p \in N_{eps}(q) and |N_{eps}(q)| > minPts \quad (7)$$

Specifically, in this context, a cluster can only be defined when it contains a sufficient number of points.

- Density - Connected: A point $p$ is density-connected to a point $q$ if there exists a point $a$ such that both $p$ and $q$ reach the density threshold with point $a$.
- A cluster: A cluster $C$ in the DBSCAN algorithm is a non-empty subset of $X$. For a point $p \in C$ and $q$ reaching the density threshold with point $p$ then $d \in C$. And $\forall p, q \in C$, $p$ is density-connected to $q$.
- Noise: In the DBSCAN algorithm, it is possible to identify noise points, which is not addressed by the clustering models mentioned above. Let $C_1, C_2, \ldots, C_n$ be the clusters of the dataset. Then, the set 0 represents all the remaining points of the dataset that do not belong to any of the clusters $C_1, C_2, \ldots, C_n$.



*Picture 3.7. DBSCAN vs K-Mean Comparison [2]*

In the figure [3.7], we can observe that DBSCAN provides significantly better clustering results than KMeans, especially for data with complex shapes. Additionally, DBSCAN allows us to identify outliers in the data.

However, to achieve such good results, we need to determine two parameters: the minimum number of points *minPts* required for a cluster to be considered, and the radius *eps* around a data point being examined. Finding an optimal parameter set for the DBSCAN model can be challenging because it requires tuning both parameters simultaneously.

## 3.4 Contrastive learning method

### 3.4.1 Contrastive loss

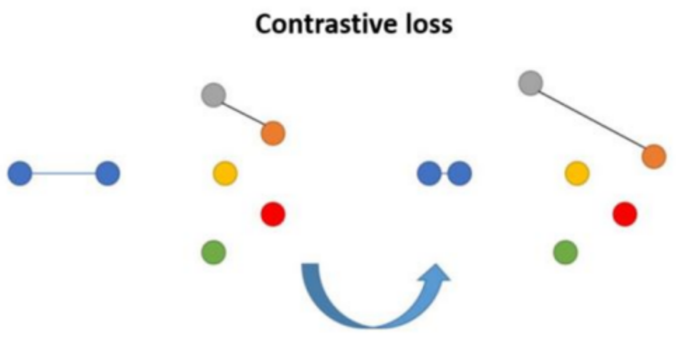
As the name suggests, contrastive learning is a technique that relies on the similarity or contrast of data to *"pull"* similar data closer together and *"push"* contrasting data farther apart. To achieve this, we need to use a similarity measure to calculate the distance between the representation vectors of the data. By combining this with label information or generating variations of the data from the original data, we can consider pairs of similar data and evaluate contrasting data or the remaining parts of the data within the group as contrasting pairs. The goal of contrastive learning is to rely on the invariant features of the data when comparing contrasting pairs of data. From this, the model can learn higher-level features of the data, which are distinctive points that enhance the predictive ability of the model.

Compared to traditional supervised machine learning methods that heavily rely on labeled data, it is worth noting that the amount of unlabeled data currently surpasses the labeled data by a large margin. While acknowledging that there is still a significant amount of labeled data available, the rapid increase in the types of problems and human requirements means that the current amount of labeled data is insufficient. Especially in the field of natural language processing, labeling this data requires substantial resources, time, and accuracy verification. As a result, self-supervised and semi-supervised learning methods have become research topics in many papers. With self-supervised machine learning techniques, we can train models very effectively using unlabeled data. Self-supervised learning has been widely applied in two main directions: generative adversarial networks and contrastive learning. In recent years, the best self-supervised learning methods have shifted from models trained to learn features from unlabeled data, such as rotation and colorization, to contrastive learning. This trend is reinforced by numerous breakthrough studies in the field of natural language processing using contrastive learning methods. Lastly, the success of the contrastive learning method heavily relies on the choice of the loss function, as the model's ability to learn similar and contrasting data representations is greatly influenced by the loss function used by the algorithm.

### 3.4.2 Some of the contrastive loss

#### Contrastive loss

The contrastive loss function is inspired by the idea that if two data points have similar characteristics or the same label, the model should "pull" their representations closer together in the vector space. On the other hand, if the data points are different or have different labels, the model should "push" their representations further apart in the vector space. By doing so, the model learns to distinguish between similar and dissimilar data points, effectively capturing the underlying structure and relationships within the data. This approach encourages the model to create compact clusters of similar data points and increase the separation between different data points in the vector space, facilitating better discrimination and representation learning. [3.9]



*Picture 3.8. Illustrate the operation of the model using the contrastive loss function [11]*

For example, consider a dataset $x = \{x_1, x_2, \ldots, x_n\}$ with $n$ data points. Corresponding to the dataset $x$, we have a set of labels $y = \{y_1, y_2, \ldots, y_n\}$ for the $n$ given data points. We use a projection function $f_\theta(.) : x \to \mathbb{R}^d$ used to encode or map $x_i$ into a vector space. Therefore, suppose we have a pair of data points $(x_i, x_j)$ and $y_i = y_j$ then this two data points will be represented close to each other in the vector space. Hence, in the contrastive loss function, the model takes a pair of input data $(x_i, x_j)$ and aims to *minimize* the distance between their vector representations in the vector space. At the same time, it also needs to *maximize* the distance between data points with different labels, as shown in the figure. From there, the contrastive loss function is defined as follows:

$$L(x_i, x_j) = D_{i,j}||f_\theta(x_i) - f_\theta(x_j)||_2^2 + D_{i,j}\max(0, \varepsilon - ||f_\theta(x_i) - f_\theta(x_j)||_2)^2 \tag{8}$$

With $D_{i,j}$ returns 1 when $y_i = y_j$ and returns 0 otherwise $y_i \neq y_j$.

### InfoNCE Loss (Noise Contrastive Estimation)

This loss function is commonly used in contrastive learning. It aims to maximize the agreement between positive pairs (similar samples) while minimizing the agreement between negative pairs (dissimilar samples). The formula for InfoNCE loss is:

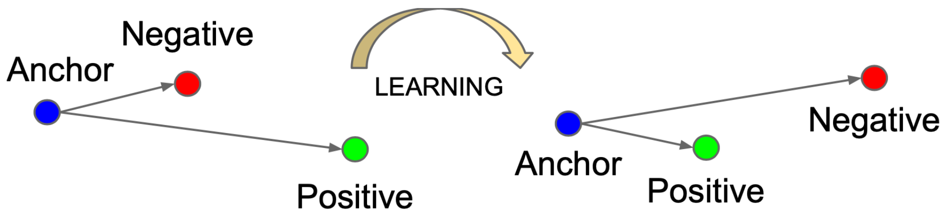
$$L = -E[\log \frac{f(x,c)}{\sum_{x' \in X} f(x',c)}] \tag{9}$$

Where $sim(a,b)$ represents the similarity score between samples $a$ and $b$, $xp$ denotes a positive sample, $xn$ represents negative samples, $T$ is a temperature parameter that controls the sharpness of the probability distribution, and $\sum$ denotes the sum over all negative samples.

## Triplet loss

Triplet loss focuses on triplets of samples: an anchor sample, a positive sample (similar to the anchor), and a negative sample (dissimilar to the anchor). The objective is to maximize the distance between the anchor and negative samples while minimizing the distance between the anchor and positive samples. The formula for triplet loss is:

$$L(x,x^+,x^-) = \sum_{x \in X} max(0, ||f(x) - f(x^+)||_2^2 - ||f(x) - f(x^-)||_2^2 + \varepsilon) \tag{10}$$



*Picture 3.9. Illustration of triplet loss given one positive and one negative per anchor [13]*

Where $d(a,b)$ represents the distance between samples $a$ and $b$, $xp$ denotes a positive sample, $xn$ represents a negative sample, and the margin is a hyper-parameter that sets a minimum desired separation between the positive and negative samples.

## N-Pair Loss

N-Pair loss extends the concept of triplet loss to include more than one positive sample. It constructs $N$ pairs of positive samples for each anchor sample and applies a softmax function to encourage high similarity scores for the positive samples and low similarity scores for the negative samples. The formula for N-Pair loss is:

$$L = -\log \frac{exp(f(x)^T f(x^+))}{exp(f(x)^T f(x^+)) + \sum_{i=1}^{N-1} exp(f(x)^T f(x_i^-))} \tag{11}$$

Where $sim(a,b)$ represents the similarity score between samples $a$ and $b$, $xp1$ denotes the first positive sample, $xn$ represents negative samples, and the sum is taken over all negative samples.

These are just a few examples of contrastive loss functions commonly used in contrastive learning. Different loss functions may be suitable for specific applications and datasets, and researchers continue to explore and develop new variations of contrastive losses to enhance the performance of contrastive learning algorithms.

To sum up, for this section, each method of embedding, clustering, or loss function has its own advantages for each case and problem, we take those basic theories as the foundation for researching, developing, and testing in our downstream task to give the highest accuracy.

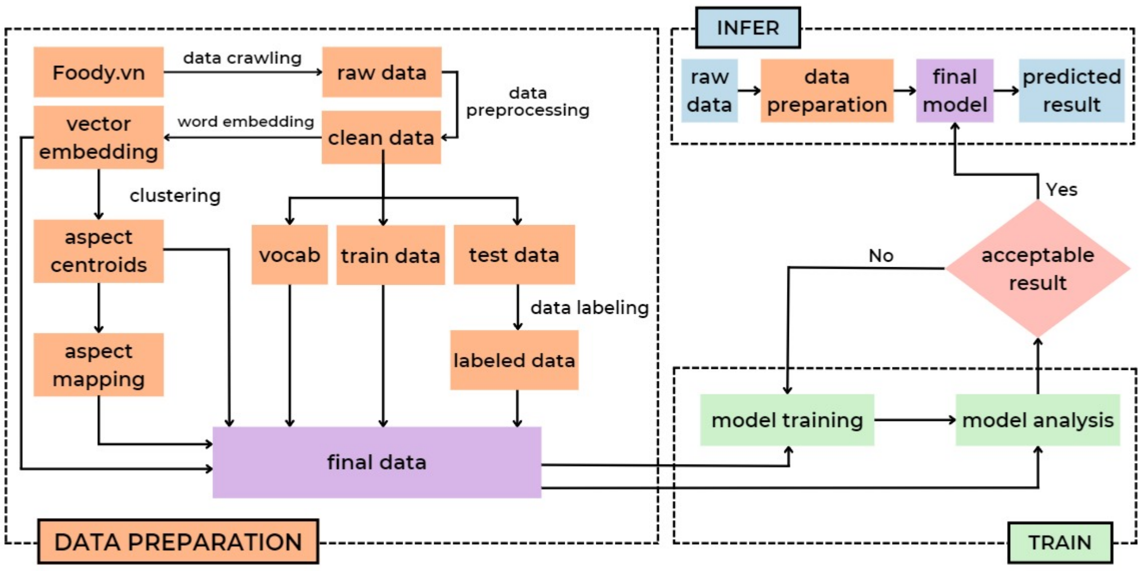
# Section 4. The proposed method

## 4.1 Overall method workflow

Detecting aspects in a comment/review comprises various steps. Therefore, building a *workflow* to fully understand the process of detecting aspects is crucial. After consideration, we decided to build a workflow including three main stages: data preparation collected from an online food ordering website called *Foody*, training model and inferring data for realistic results.

Below is a summary of the workflow of the problem [4.1]:

- *Data preparation*: In this stage, the data is crawled from the website and goes through pre-processing steps such as removing punctuation, emojis, adjusting the format of prices, correcting spelling,... Then, the pre-processed dataset is used to create a dictionary and an embedding vector set for the problem. Next, the embedding vectors are input into the clustering algorithm to generate aspect centroids, which are used to create aspect mapping - name of each aspect and became input for modeling phase. From this phase, we have the final dataset and proceed to split it into training and testing sets to train the model.
- *Train*: In this stage, the input data is divided into batches for the model to learn.



*Picture 4.1. Overall method workflow*

The model will learn by optimizing a loss function in contrastive learning with the goal of minimizing this loss. Then, a test set with labeled data will be used to evaluate the results of the model training. If the results are not satisfactory, we will re-evaluate and retrain the model until we get acceptable result. Conversely, if the results are good, the model will be used for the inference step.

- *Infer*: In this stage, we will simulate the entire workflow of the problem from the data pre-processing stage in order to generate the input for our model. The purpose is to produce the desired aspect output for each input comment/review.

## 4.2 Clustering model

In our thesis, we will be exploring the use of word embedding clustering to identify semantic relationships between words. For that, we will choose *K-Means* is the popular clustering algorithms that can be used for word embedding clustering.

*K-Means* is a **distance-based** clustering algorithm, which means that it clusters points based on their distance to each other. The underlying idea is that the *word2vec-generated vector* representations of words are designed to arrange words with similar meanings in close proximity within a two-dimensional space. Leveraging an appropriate clustering method, we can effectively gather words with related semantics into coherent clusters. These clusters then become instrumental in guiding subsequent analytical steps. Therefore, *K-Means* can work well for word embedding clustering, as the **distance** between two words in vector space can be a good proxy for their semantic similarity.

However, *K-Means* can be sensitive to noise and outliers, and it can be difficult to determine the optimal number of clusters.

## 4.3 Contrastive learning

In the above contrastive loss functions, they can not be directly applied to our problem due to the nature of unlabeled data. Therefore, it is necessary to determine a more suitable loss function, especially when we utilize the clustering aspect mentioned above.

Similar to [14], our contrastive learning model requires pairs of similar and dissimilar data, in this thesis, we consider the pair of embedding vectors for comment/review $s_{x_i,E}$ and the corresponding embedding vector for the comment on the standard context $s_{x_i,A}$ as a similar pair. The remaining pairs, $s_{x_j,E}$ and $s_{x_i,A}$ are initialized as dissimilar pairs.

From there, we construct a contrastive loss function as follows:

$$l_i = -\log \frac{\exp\left(\frac{sim(s_{i,E}, s_{i,A})}{\mu}\right)}{\sum_{j=1}^{N} [j \neq i] \exp\left(\frac{sim(s_{j,E}, s_{i,A})}{\mu}\right)} \tag{12}$$

With $[j \neq j]$ returns 0 when $j = i$ and returns 1 when $j \neq i$.

Furthermore, here we also utilize the cosine similarity function

$$sim(s_{i,E}, s_{i,A}) = \frac{(s_{i,E})^T s_{i,A}}{||s_{i,E}|| \times ||s_{i,A}||} \tag{13}$$

to calculate the similarity between two vectors, $s_{i,E}$ and $s_{i,A}$.

## 4.4 Mapping model outputs to labels

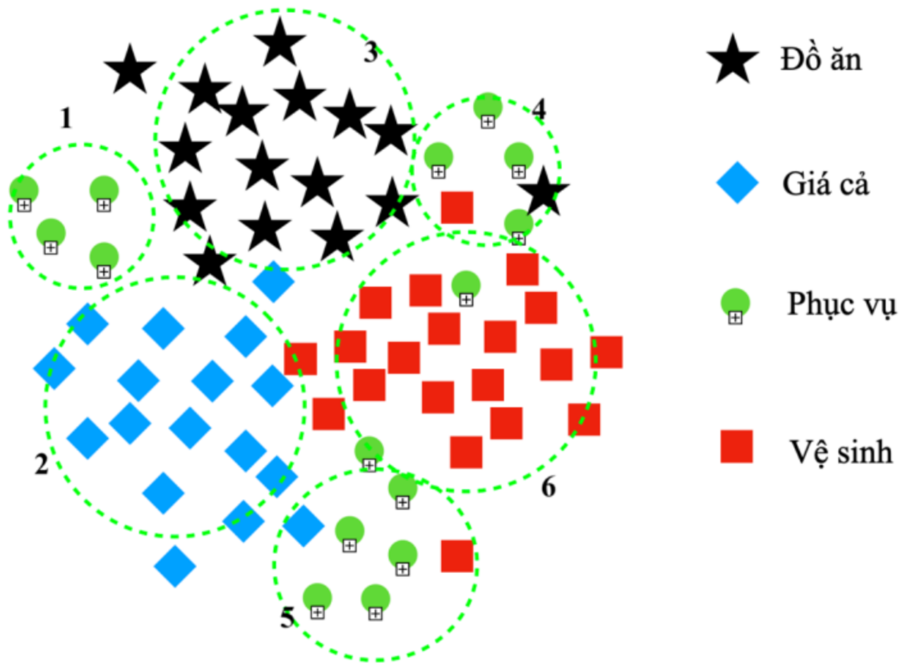
Using the predicted context set as a soft label can help the model determine similar and dissimilar pairs of data. However, to obtain the final result, the model needs a mapping rule from this soft label set to the standard context set in order to determine the comment context.

During the training process, this thesis utilizes the representation space of the comments/reviews to determine the representation vector of the predicted context set based on the semantic similarity of words. Specifically, with the vector representation space of the context set $A$ and the vector representation space of the comment/review $X$, we have computed the similarity matrix:

$$G = A \times E^T \tag{14}$$

Let $G \in \mathbb{R}^{N\times V}$ with $N$ is the number of predicted contexts and $V$ is the size of the data dictionary. Afterwards, with vector $G$, we take the *top-k* keywords to represent and explain the predicted contexts of the model.

When mapping the predicted context set of the model to the standard context set, most unsupervised machine learning models focus on the cohesion and meaningfulness of the predicted contexts. Therefore, it is proposed to map the predicted contexts of the model back to the standard context set. Typically, this approach requires the number of predicted contexts to be larger than the number of standard contexts but not significantly different. This method is called *Many-to-one mapping*.

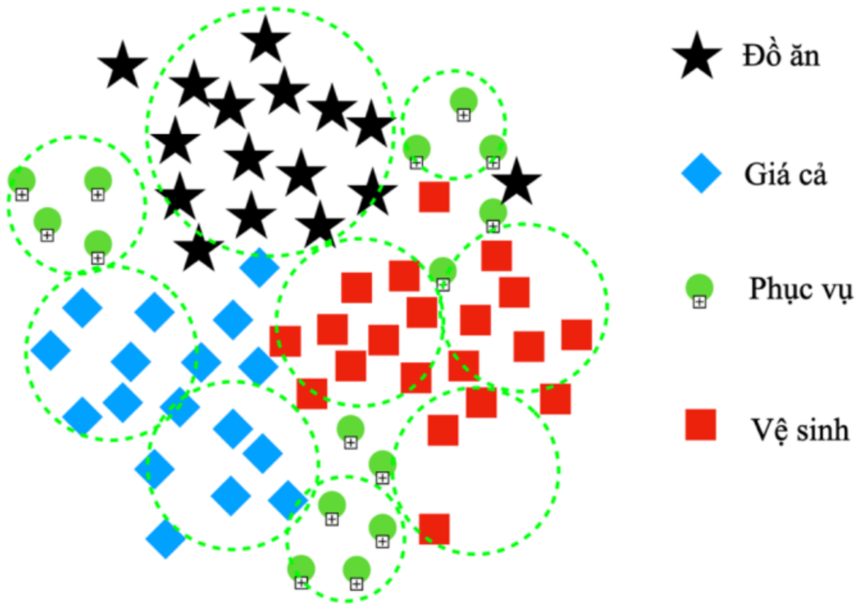


*Picture 4.2. Illustrate how the Many-to-one mapping works*

In the one-to-many mapping, each predicted aspect is mapped to the *"golden"* aspects - the set of final standard aspects for the thesis. For example, cluster 3 is mapped to the *"Đồ ăn"* aspect, and clusters 1,4, and 5 are mapped to the *"Phục vụ"* aspect, and so on for the remaining clusters. However, this mapping approach has some issues. In the case of clusters 4 and 5, their data might contain noise related to words that could be used to describe *"Vệ sinh"* or *"Phục vụ"* attitude. In such cases, these clusters might be considered as noise and not used for mapping. This is because these words can occur together with other words, providing a more specialized description for *"Vệ sinh"* and *"Phục vụ"* attitude. Additionally, there are cases where these words are used in comments but not in the context of *"Vệ sinh"* or *"Phục vụ"*, affecting the prediction results.

Furthermore, for some weakly supervised machine learning methods – which means using models instead of using fully labeled datasets – actual limited, noisy, or inaccurate labeled datasets are utilized. The goal is to enable these models to learn from the labeled data, hoping that they can identify representative words for aspects, and thereby determine suitable standard aspects. This method is called *one-to-one mapping*.

However, in this iteration of the model, based on [14], unlike the two methods mentioned above, we employed a *High-resolution selective mapping* approach. This was achieved by using a larger number of predicted aspects compared to the *"golden"* aspects, multiple times over. The aim was to enhance the coverage capacity of the model's predictions.



*Picture 4.3. Illustrate how the High-resolution selective mapping works*

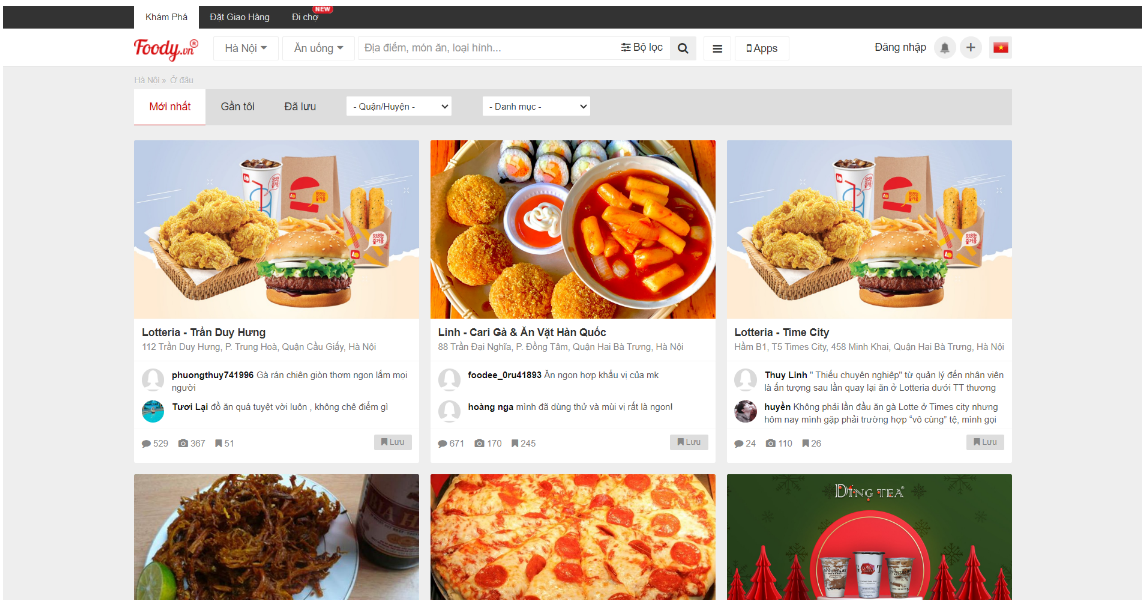
The *High-resolution selective mapping* works well when the number of predicted aspects is sufficiently large. This leads to more detailed representation of words that correspond to specific aspects, resulting in more accurate representations of those words. Additionally, noisy keywords are excluded from the predicted aspect sets, thereby avoiding cases where representative words have multiple meanings and can affect the predictions, similar to the *many-to-one mapping* approach.

In this case, we employed 30 predicted aspects, based on the number of *"golden"* aspects, which are 4 and considering the effort required to map data from the predicted aspect set to the *"golden"* aspect set.

## Section 5. Experiments

### 5.1 Dataset

The dataset for the thesis comprises comments from various restaurants across 12 provinces and cities in Vietnam, including *Ho Chi Minh City, Hanoi, Da Nang, Can Tho, Hai Phong, Lam Dong, Hue, Khanh Hoa, Dong Nai, Dien Bien, Binh Duong, and Binh Dinh*. As a result, the dataset contains both Vietnamese (predominant) and other languages (such as English, Korean, etc.). Furthermore, the dataset contains a lot of noise, such as spelling errors, emojis, hashtags, etc., necessary to pre-processing the data before being input into the model.



Lotteria - Trần Duy Hưng
112 Trần Duy Hưng, P. Trung Hoà, Quận Cầu Giấy, Hà Nội
phuongthuy741996 Gà rán chiên giòn thơm ngon lắm mọi người
Tươi Lại đồ ăn quá tuyệt vời luôn , không chê điểm gì
529 367 51

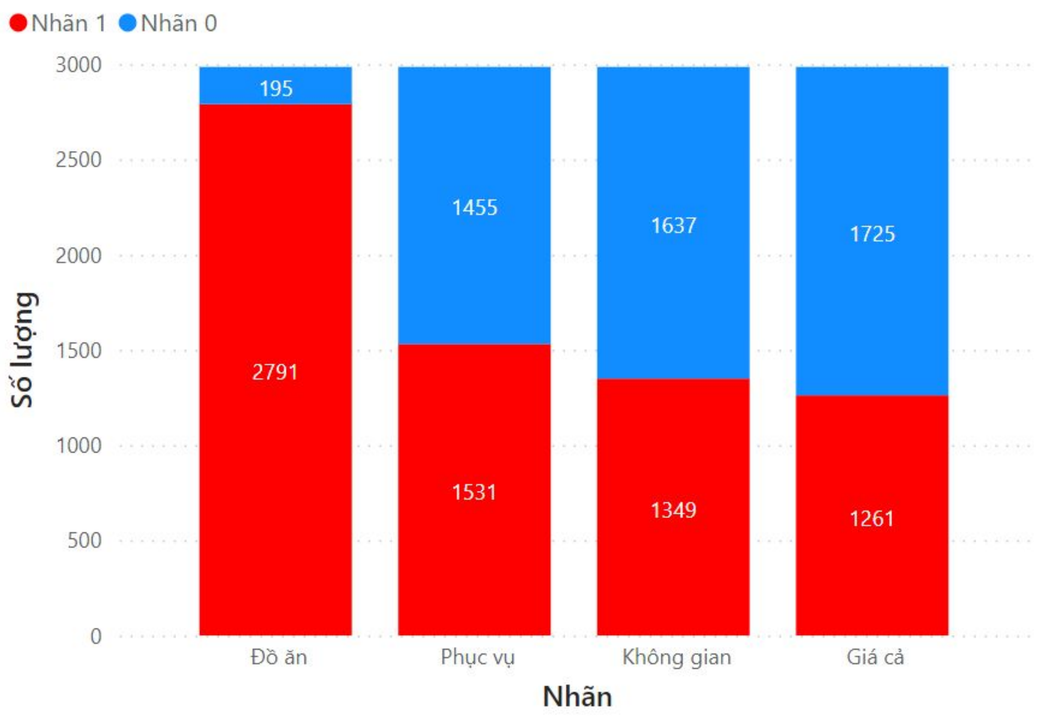
Linh - Cari Gà & Ăn Vặt Hàn Quốc
88 Trần Đại Nghĩa, P. Đồng Tâm, Quận Hai Bà Trưng, Hà Nội
foodee_0ru41893 Ăn ngon hợp khẩu vị của mk
hoàng nga mình đã dùng thử và mùi vị rất là ngon!
671 170 245

Lotteria - Time City
Hầm B1, T5 Times City, 458 Minh Khai, Quận Hai Bà Trưng, Hà Nội
Thuy Linh " Thiếu chuyên nghiệp" từ quản lý đến nhân viên là ấn tượng sau lần quay lại ăn ở Lotteria dưới TT thương
huyền Không phải lần đầu ăn gà Lotte ở Times city nhưng hôm nay mình gặp phải trường hợp "vô cùng" tệ, mình gọi
24 110 26

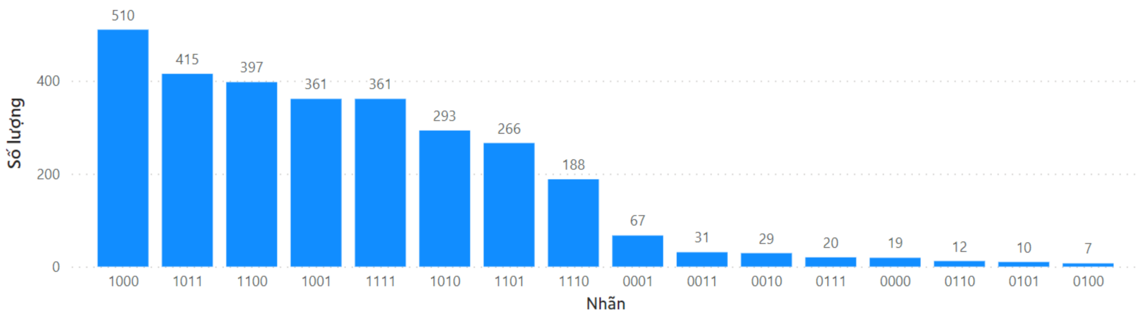
*Picture 5.1. Foody Website*

The data used for evaluation is a small portion extracted from the dataset retrieved from the *foody.vn* website [5.1].

The process involves selecting comments/reviews that ensure diversity in aspects for the comments/reviews in the test set. Manual labeling is then conducted on nearly 3000 comments to create an evaluation set for assessing the model's performance.



*Picture 5.2. Illustrate how the High-resolution selective mapping works*



*Picture 5.3. Number of each aspects*

## 5.2 Experiments method

- Step 1: Preprocess the raw data that has been retrieved, pre-process, perform normalization, and separate a portion of the dataset for labeling to be used in the testing step.
- Step 2: Run the embedding model on the training dataset.
- Step 3: Cluster the data and identify *golden aspect* within the dataset.
- Step 4: Calculate the embedding vectors for comments along with the corresponding aspects expressed in those comments/reviews. Utilize a self-attention mechanism with a smoothing factor to adjust the proportion between aspects in the representation of a comment/review.

- Step 5: Construct the contrastive loss function and optimize the model parameters.

The purpose of this experiment is to evaluate the results of the contrastive learning approach in unsupervised machine learning for the *Vietnamese* language dataset.

### 5.2.1 Data overview

The data in this thesis consists of user comments/reviews left to evaluate the quality of food and restaurants. These comments/reviews often cover various aspects of their experiences rather than focusing on a single topic. However, there will be a common set of aspects around which the content of user comments revolves.

We define each comment $x = \{x_1, x_2, \ldots, x_T\}$, where $x_t$ represents the words in the sentence, and the goal of the model is to predict the aspect that the comment refers to in the standard aspect set $y \in \{y_1, y_2, \ldots, y_K\}$, where $K$ is the number of aspects considered in the problem.

### 5.2.2 Data processing

The data for this thesis was scraped from the *foody.vn* website, consisting of over $200,000$ user comments/reviews, which were collected before pre-processing. After retrieving the data, it was normalized by standardizing abbreviations, correcting basic spelling errors, removing special characters, and formatting price information.

It can be observed that in the Vietnamese language, compound words often require to be used together to convey a complete meaning or even to have a specific meaning. For example, the word "dễ" on its own can be understood as *"dễ dàng"*, and the word *"thương"* can means *"yêu thương"*. However, when these two words are combined, we have the phrase *"dễ thương"* which has a completely different meaning.

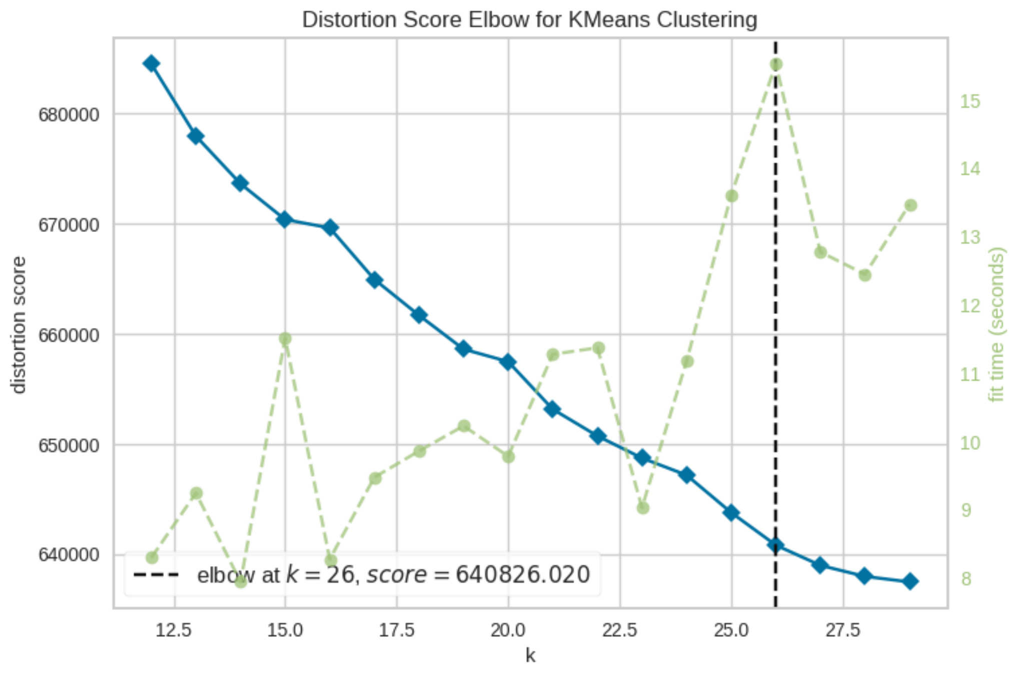
Therefore, it is important to group compound words together to ensure the intended meaning for individual words as well as for the entire sentence. In this thesis, we performed word grouping using the *UnderTheSea* library. Through this pre-processing step, compound words such as *"niềm nở"*, *"nhiệt tình"*, and *"dễ thương"* would be represented as *"niềm_nở"*, *"nhiệt_tình"*, and *"dễ_thương"* respectively. This ensures that these keyword phrases are always treated as a whole during subsequent processing steps. This step is commonly referred to as word segmentation.

After pre-processing, around 3000 data samples were extracted to be labeled for use as a test dataset. 3 individuals were involved in labeling this dataset, following common guidelines to ensure consistent results among all 3 labelers. Once the data was initially labeled, I conducted another round of data inspection and made necessary label revisions if there were unclear cases in the classification process.

### 5.2.3 Specify golden aspect

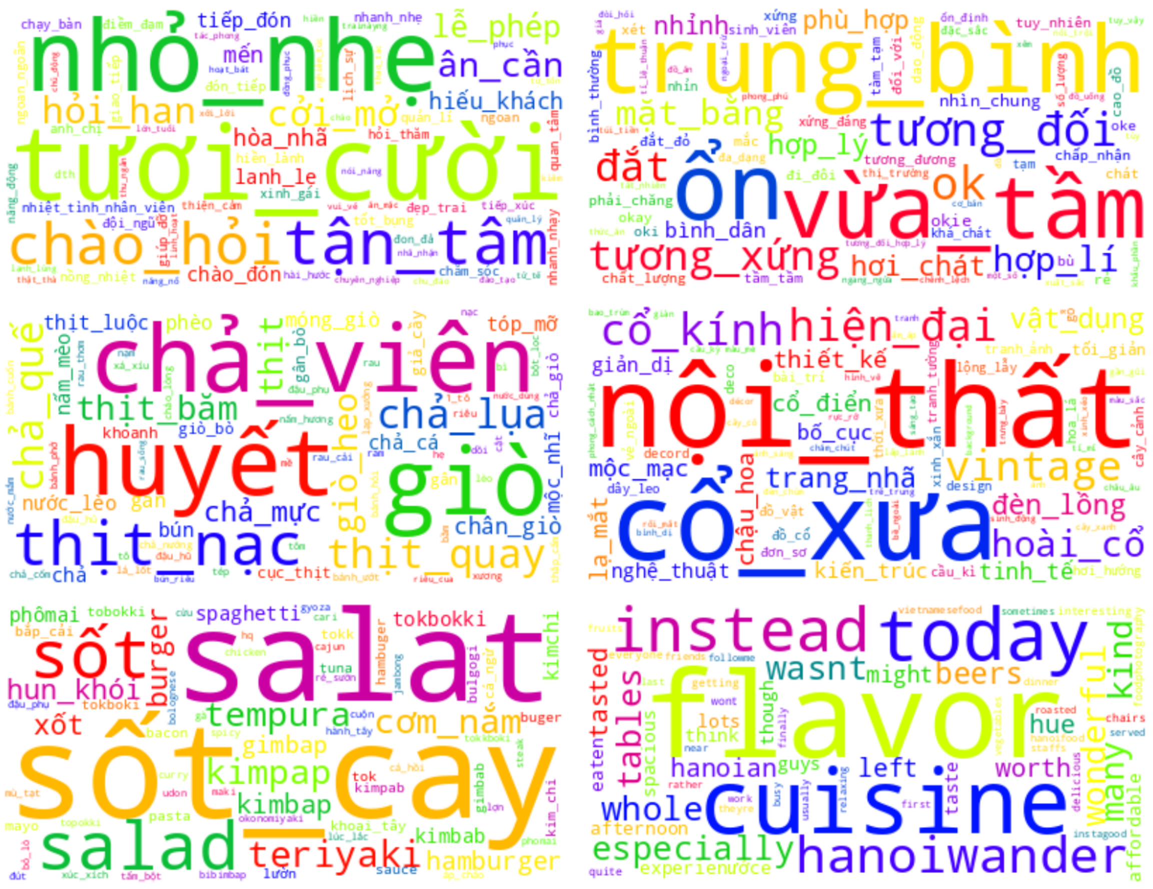
To generalize the dataset and determine the *"golden"* aspect - the set of final standard aspects for the thesis $y \in \{y_1, y_2, \ldots, y_K\}$, we used the K-Means clustering model on the dictionary of the data, which is the collection of all unique words compiled from the dataset. This approach aims to identify prominent word clusters to determine the content of comments related to specific topics based on the grouped data clusters.

By constructing random cluster centers and clustering the representation vectors of words based on their proximity to the nearest cluster center, we formed clusters to identify the relevant topics. To determine the appropriate number of cluster centers, We used the *KElbowVisualizer* method to find the optimal value for $K$ based on the *distortion function*'s change. However, with this dataset, it was challenging to find the optimal point for clustering. Therefore, I considered three values for $K = [16, 24, 30]$ and evaluated the clustering results on the dictionary.



*Picture 5.4. Graph of distortion score by KElbowVisualizer with number of clusters k*

However, based on the results [5.4], it was observed that the choice of $K$ values had minimal effect on the final number of golden aspects, which is 4. These aspects include: *Đồ ăn*, *Giá cả*, *Không gian*, and *Phục vụ*. The variations in $K$ values mainly influenced the number of clusters within each aspect, reflecting a more accurate coverage of aspect clusters.



Picture 5.5. Visualization of aspects from KMeans clustering algorithms

Below are some prominent clusters obtained after running the K-Means model [5.5]:

[5.5] is a *Word cloud* chart illustrating the frequency of word occurrences in comments/reviews, where words closer to the cluster center appear larger. From this, it's evident that the comment/review dataset represents a diverse range of aspects related to the restaurant experience, such as food quality, prices, service attitude, ambiance, food location,.... For example, in a cluster related to service attitude, we can clearly see keywords like *"tươi cười", "nhỏ nhẹ", "quan tâm", "nhiệt tình"*,.... These appearing keywords help us assess the quality of clustering algorithms as well as the significance of the created clusters.

## 5.2.4 Data representation

To start representing data, we need to pass the dataset through a word embedding model to represent the words in a vector space. In this task, we have 2 entities that need to be represented in vector space: *the comments/reviews* and the corresponding set of *aspects*.

**Comments/Reviews representation**

For a comment/review $x$, there are usually multiple words used to introduce, describe emotions, and refer to different aspects of the comment/review. However, for the purpose of predicting the aspect of the comment, we can focus on the important keywords in the sentence, so that based on those keywords, we can easily determine the aspect that the user is referring to in that comment/review. For example, with a comment/review like: *"bánh tròn tròn nhìn cưng lắm nha dẻo dẻo nữa bánh ăn lúc lạnh lạnh càng ngon hơn nha bên trong có nhân dâu và dừa dứa có gì đó nữa á mà mình quên rùi một hộp 5 cái 45 nghìn đồng nà"*. With some words in the sentence like *"bánh"*, *"ngon"*, *"dâu"*, and *"45 nghìn đồng"*, we can quickly determine that the sentence mentions at least 2 aspects: *"đồ ăn"* and *"giá cả"*. Therefore, these keywords need to be given higher weights than words that do not indicate any aspect in the comment/review, such as *"lắm nha"*, *"rùi"*, *"có gì"*,... With the above idea, the representation of comments/reviews used in this thesis is also calculated based on the weights of the words in that comment/review. In other words, the words in the comment/review that have a higher similarity with the predicted aspects will have a higher weight in the calculation of the comment's/review's representation vector.

We will calculate the representation vector of a comment/review by summing up the representation vectors of the words in that comment/review. We will use the $softmax$ function to assign weights to the words in the sentence during the calculation.

Let $E \in R \times V$ represents the embedding matrix of the dictionary generated from the dataset, where $K$ is the size of the vocabulary, and $M$ is the dimension of the embedded word vectors.

For each comment/review data $x = \{x_1, x_2, \ldots, x_T\}$, we construct the representation vectors $s_{x,E}$ based on the embedding vectors of each word in the comment/review $\{E_{x_1}, E_{x_2}, \ldots, E_{x_T}\}$, along with the self-attention mechanism defined as follows:

$$s_{x,E} = \sum_{t=1}^{T} \alpha_t E_{x_t} \tag{15}$$

where $\alpha_t$ is an attention weight and is calculated as follows:

$$\alpha_t = \frac{e^{u_t}}{\sum_{\tau=1}^{T} e^{u_\tau}} \tag{16}$$

$$u_t = \lambda tanh(q^T(W_E E_{x_t} + b_E)) \tag{17}$$

Here, we have $u_t$ as the alignment score and $q^T = \frac{1}{T}\sum_{t=1}^{T} E_{x_t}$ as the query vector. With $W_E \in \mathbb{R}^{M \times M}, b_E \in \mathbb{R}^M$ as the parameters trained throughout the model training process, and the smoothing parameter $\lambda$ as a hyper-parameter. In this case, the attention mechanism used is the Smooth self-attention[14], which uses the smoothing parameter $\lambda$ to adjust the influence of words on the representative vector of the sentence. Additionally, with $u_t$ using the *tanh* function, it ensures that a single word does not have an excessively large weight in computing the representation vector of the comment, while still considering the importance of keywords in the sentence.

**Aspect representation**

In this thesis, the data used is *unlabeled* data, so we do not have the initially assigned aspects along with the representative vectors to determine the similarity and dissimilarity pairs of data.

By clustering the data into words in the sentence, this thesis defines a number $k = 30$ as the number of clusters in the dictionary based on the determined standard aspects, which is 4, and the resources needed to map these clusters to the standard aspects.

The input to the model uses these 30 predicted aspects as a set of predicted aspects, which can be considered as subsets of the standard aspects, as multiple predicted aspects can be mapped to one of the four standard aspects.

At this point, the current comments/reviews do not have accurate information about which aspect they are referring to, whether it's the standard aspects or the predicted aspects. However, by clustering the words in the dictionary based on their vector representations so that words within a cluster share a common theme, we can use the relationship between the keywords in the predicted aspects and the comments/review represented by the keywords in the sentence to determine which aspect the comment/review is referring to.

Therefore, in this thesis, we construct a representation vector $s_{x,A}$ for comment/review $x$ using the embedding vectors of the aspects we built $A_1, A_2, \ldots, A_N)$ with $N$ being the number of predicted aspects of the model through another attention mechanism represented below:

$$s_{x,A} = \sum_{n=1}^{N} \beta_n A_n \tag{18}$$

With attention weights $\beta$ calculated as follows:

$$\beta_n = \frac{e^{(}v_{n,A}^T s_{x,E} + b_{n,A})}{\sum_{i=1}^{N} e^{(}v_{i,A}^T s_{x,E} + b_{i,A}})\tag{19}$$

With $v_{n,A} \in \mathbb{R}^M$ and $b_{n,A} \in \mathbb{R}^M$ as the trained parameters. Using the $softmax$ function to determine the proportion of the comment belonging to the corresponding aspects, the model will then *"pull"* the comment closer to the regions of the corresponding aspects[14]. Here, $\beta$ is considered as a soft label (probability distribution) of the comments on the set of $N$ aspects.

At this point, the problem can be viewed as a multi-label problem, with the number of labels being 30, which is the number of clusters determined above.

## 5.3 Enviroment and Performance measurement

### 5.3.1 Enviroment

The programming language used for this thesis is Python, and the code environment is *Colab*. The *Word2Vec* and *FastText* model were implemented using the *gensim* library, while *PhoBERT* utilized the Transformer library for algorithm implementation.

### 5.3.2 Performance measurement

The results of the models on each dataset were evaluated using a classification task, and the most appropriate metric chosen for this task was the *precision, recall and f1-score* for *class 1*. Those of metrics indicate the classification quality of the models just for the class we care about (class 1).

*Precision* measures the accuracy of the positive predictions made by the model. It indicates the ratio of true positive predictions to the total number of positive predictions. In the context of our thesis, *Precision* would reflect how accurately your model identifies each aspect in the comments. A high *Precision* indicates that the identified aspects are indeed relevant and reliable.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{20}$$

*Recall*, on the other hand, measures the model's ability to correctly identify all relevant instances. It is the ratio of true positive predictions to the total number of actual positive instances. In the context of our thesis, *Recall* would show how well your model captures all instances of a specific aspect. A high *Recall* suggests that the model is effectively detecting relevant aspects even if it occasionally misclassifies other instances as relevant.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negative} \tag{21}$$

*F1-score* is a harmonic mean of *Precision* and *Recall*, providing a balance between the two metrics. It is particularly useful when there is an uneven class distribution, as in your case where some aspects might be more prevalent than others. The F1-score takes both false positives and false negatives into account, making it a comprehensive metric for overall model performance.

$$F1 - score = \frac{Precision \times Recall}{Precision + Recall} \tag{22}$$

The objective of the proposed methods in the thesis is to maximize those metrics, aiming for a value as close to 1 as possible.

## 5.4 Result

### 5.4.1 Result of different word embedding methods

#### FastText

Utilize the pre-trained *FastText* model from the *gensim* library and retrain it on the dataset of the thesis with the parameter set:

- Size: 256 is the dimensionality of the embedding vector space.
- Min_count: 10 for the FastText model to discard words with a frequency of less than 10 occurrences in the text corpus.

- Window: 7 is the number of context words considered both before and after the target word. Here, 7 context words before and 7 after the target word are used.
- Max_n: 3 is used to break down the original words into subwords of 3 characters.
- Worker: 4 is set to accelerate the processing speed of the model by utilizing multiple CPU resources for computation.

After obtaining the text corpus's embedding data, the next step is to feed it into the training model based on the defined contrastive loss function.

Below are the results of the model:

**Đồ ăn**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.07 | 0.97 | 0.14 | 196 |
| 1 | 0.99 | 0.14 | 0.24 | 2790 |
| accuracy | | | 0.19 | 2986 |
| macro avg | 0.53 | 0.56 | 0.19 | 2986 |
| weighted avg | 0.93 | 0.2 | 0.23 | 2986 |

*Table 5.1. Scores of Đồ ăn aspect - FastText*

**Giá cả**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.59 | 0.98 | 0.73 | 1725 |
| 1 | 0.6 | 0.06 | 0.11 | 1261 |
| accuracy | | | 0.59 | 2986 |
| macro avg | 0.61 | 0.51 | 0.42 | 2986 |
| weighted avg | 0.61 | 0.59 | 0.47 | 2986 |

*Table 5.2. Scores of Giá cả aspect - FastText*

**Không gian**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.52 | 1 | 0.7 | 1637 |
| 1 | 0.87 | 0.04 | 0.08 | 1349 |
| accuracy | | | 0.57 | 2986 |
| macro avg | 0.74 | 0.52 | 0.4 | 2986 |
| weighted avg | 0.72 | 0.54 | 0.43 | 2986 |

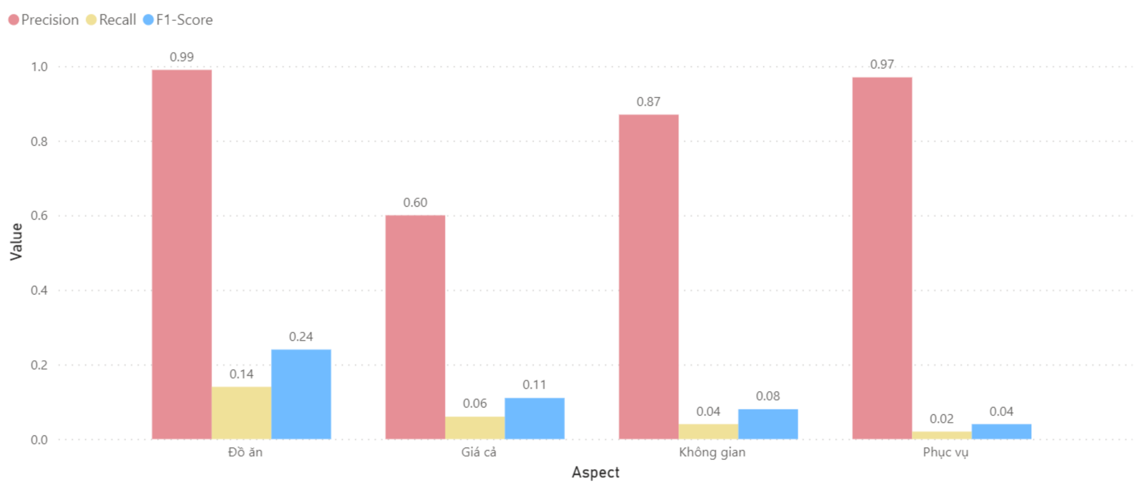
*Table 5.3. Scores of Không gian aspect - FastText*

**Phục vụ**

|  | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.49 | 1 | 0.66 | 1455 |
| 1 | 0.97 | 0.02 | 0.04 | 1531 |
| accuracy |  |  | 0.5 | 2986 |
| macro avg | 0.73 | 0.51 | 0.35 | 2986 |
| weighted avg | 0.74 | 0.5 | 0.34 | 2986 |

*Table 5.4. Scores of Phục vụ aspect - FastText*

Below [5.6] is the visualization of those results above:



*Picture 5.6. Visualizing FastText model results*

Looking at the table and the figures above, it can be observed that despite the high *Precision* of the FastText model, reaching up to 0.99 for the *"Đồ ăn"* aspect, the *Recall* is quite low, with the lowest value being 0.02 for the *"Phục vụ"* aspect. The *F1-scores* for the four aspects *"Đồ ăn", "Giá cả", "Không gian", and "Phục vụ"* are **0.24, 0.11, 0.08**, and **0.04**, respectively. Based on these results, we assess that the *FastText* model performs **poorly** with the word embeddings used in this thesis.

**Word2Vec**

Utilize the pre-trained *Word2Vec* model from the *gensim* library and retrain it on the dataset of the thesis with the parameter set:

- Size: 256 is the dimensionality of the embedding vector space.
- Min_count: 10 for the Word2Vec model to discard words with a frequency of less than 10 occurrences in the text corpus.
- Window: 7 is the number of context words considered both before and after the target word. Here, 7 context words before and 7 after the target word are used.

- Worker: 4 is set to accelerate the processing speed of the model by utilizing multiple CPU resources for computation.

In the *Word2Vec* model, we utilized a pre-trained model in Vietnamese with approximately 7.1GB of data with $1,675,819$ unique words from a corpus of $974,393,244$ raw words and $97,440$ documents [17], then compared the results with a model trained on the dataset specifically for this thesis. The results indicated that, despite having only about 200 comments/reviews segments, due to the distinctive nature of the data, employing the model trained on the thesis dataset yielded significantly better outcomes than using a pre-trained model.

**7.1GB aggregated data**

| | |
|---|---|
| hoàn_chỉnh | 0.48144 |
| đủ | 0.4436 |
| sát_thực | 0.41785 |
| trung_thực | 0.411 |
| cần_thiết | 0.40325 |
| chính_xác | 0.39552 |
| đồng_bộ | 0.39307 |
| nghiêm_chỉnh | 0.3916 |
| phù_hợp | 0.38891 |
| toàn_diện | 0.3785 |

*Table 5.5. Top 10 most similar words to đầy_đủ from 7.1GB aggregated data*

**Thesis dataset**

| | |
|---|---|
| đầy_ắp | 0.61389 |
| đầy_đặn | 0.56808 |
| thập_cẩm | 0.55298 |
| ú_ụ | 0.509261 |
| bao_gồm | 0.49898 |
| ụ | 0.48030 |
| 25_nghìn_đồng | 0.468 |
| 35_nghìn_đồng | 0.45924 |
| ngập_mặt | 0.4496 |
| 30_nghìn_đồng | 0.44265 |

*Table 5.6. Top 10 most similar words to đầy_đủ from thesis data*

Taking an example of the keyword *"đầy_đủ"*, it's evident that with the thesis dataset used, which consists of comments/reviews from customers, keywords like *"đầy_đủ"* are well represented in the model. The top 10 most similar words are also relevant, and the similarity scores among these words are consistently high, ranging from *0.61* to *0.44*.

However, with the dataset aggregated from *Wikipedia*, which is more comprehensive and covers various fields of knowledge, including fewer instances of words like *"đầy_đủ"* and its synonyms, the top 10 most similar words are not well represented, and the similarity scores are not as high as in the thesis dataset.

After obtaining the text data representation, it's further used in the training of the model based on the defined contrastive loss function.

Below are the results of the model:

**Đồ ăn**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.55 | 0.58 | 0.56 | 196 |
| 1 | 0.96 | 0.44 | 0.61 | 2790 |
| accuracy | | | 0.46 | 2986 |
| macro avg | 0.62 | 0.31 | 0.38 | 2986 |
| weighted avg | 0.59 | 0.6 | 0.58 | 2986 |

*Table 5.7. Scores of Đồ ăn - Word2Vec*

**Giá cả**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.82 | 0.35 | 0.59 | 1725 |
| 1 | 0.64 | 0.34 | 0.45 | 1261 |
| accuracy | | | 0.64 | 2986 |
| macro avg | 0.71 | 0.35 | 0.59 | 2986 |
| weighted avg | 0.68 | 0.38 | 0.61 | 2986 |

*Table 5.8. Scores of Giá cả - Word2Vec*

**Không gian**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.99 | 0.59 | 0.78 | 1637 |
| 1 | 0.91 | 0.47 | 0.62 | 1349 |
| accuracy | | | 0.74 | 2986 |
| macro avg | 0.82 | 0.63 | 0.71 | 2986 |
| weighted avg | 0.78 | 0.66 | 0.72 | 2986 |

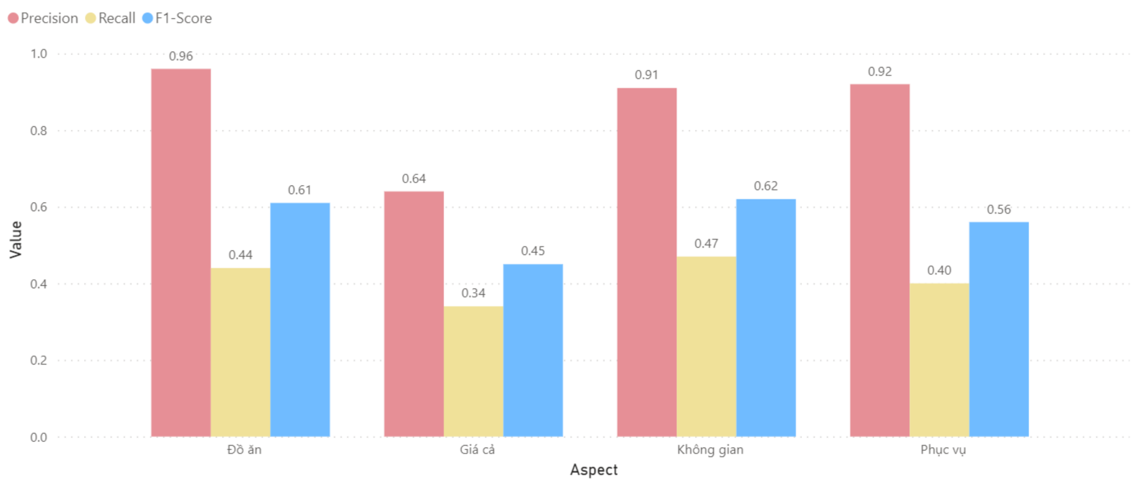
*Table 5.9. Scores of Không gian - Word2Vec*

**Phục vụ**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.68 | 0.01 | 0.02 | 1455 |
| 1 | 0.92 | 0.4 | 0.56 | 1531 |
| accuracy | | | 0.68 | 2986 |
| macro avg | 0.68 | 0.6 | 0.65 | 2986 |
| weighted avg | 0.63 | 0.67 | 0.65 | 2986 |

*Table 5.10. Scores of Phục vụ - Word2Vec*

Below [5.7] is the visualization of those results above:



*Picture 5.7. Visualizing Word2Vec model results*

Looking at the table and figures, similarly to the *FastText* model, we can observe that the *Precision* of the *Word2Vec* model is also quite high, reaching up to 0.96 for the *"Đồ ăn"* aspect. However, unlike the *FastText* model, the *Recall* of the Word2Vec model achieves a relatively decent value, averaging around 0.41, with the highest being 0.47 for the *Không gian* aspect. Consequently, the F1-scores for the four aspects *"Đồ ăn", "Giá cả", "Không gian", and "Phục vụ"* of the Word2Vec model are higher than *FastText*, with values of **0.61, 0.45, 0.62**, and **0.56** respectively. Based on these results, we evaluate that the *Word2Vec* model performs relatively **well** with the word embeddings used in the thesis.

## PhoBert

To verify the impact of different data embedding models on the model's outcomes, we employed a pre-trained BERT model on Vietnamese text data. Subsequently, we re-trained this model using the thesis dataset to represent the embedding vectors of the words.

Below are the results of the model:

**Đồ ăn**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.07 | 1 | 0.12 | 196 |
| 1 | 1 | 0 | 0 | 2790 |
| accuracy | | | 0.07 | 2986 |
| macro avg | 0.53 | 0.5 | 0.06 | 2986 |
| weighted avg | 0.94 | 0.07 | 0.01 | 2986 |

*Table 5.11. Scores of Đồ ăn - PhoBert*

**Giá cả**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.58 | 1 | 0.73 | 1725 |
| 1 | 1 | 0 | 0 | 1261 |
| accuracy | | | 0.58 | 2986 |
| macro avg | 0.79 | 0.5 | 0.37 | 2986 |
| weighted avg | 0.76 | 0.58 | 0.42 | 2986 |

*Table 5.12. Scores of Giá cả - PhoBert*

**Không gian**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.55 | 0.99 | 0.7 | 1637 |
| 1 | 0.35 | 0.01 | 0.02 | 1349 |
| accuracy | | | 0.54 | 2986 |
| macro avg | 0.45 | 0.5 | 0.36 | 2986 |
| weighted avg | 0.46 | 0.54 | 0.39 | 2986 |

*Table 5.13. Scores of Không gian - PhoBert*

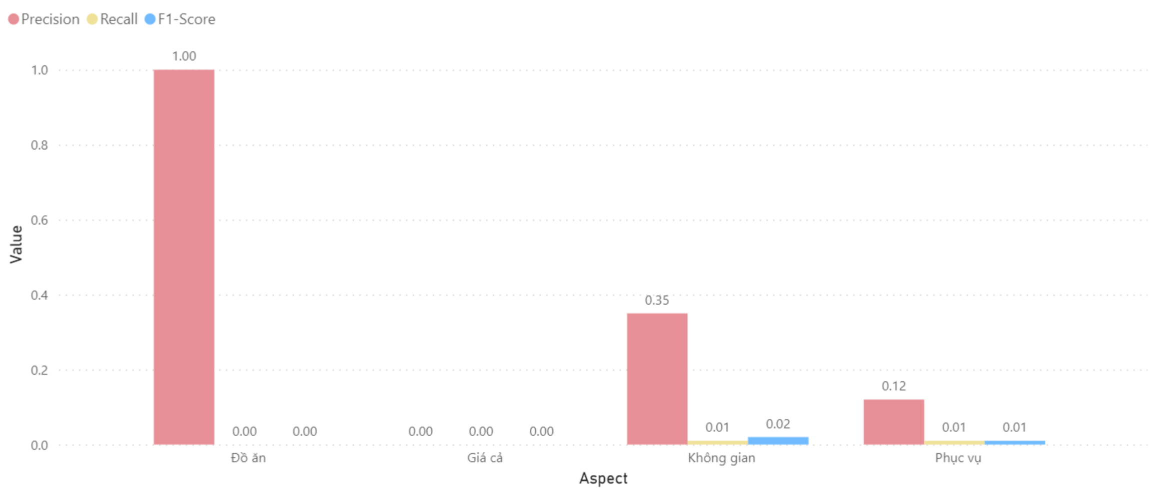
**Phục vụ**

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.49 | 1 | 0.66 | 1455 |
| 1 | 0.26 | 0.01 | 0.01 | 1531 |
| accuracy | | | 0.49 | 2986 |
| macro avg | 0.34 | 0.5 | 0.42 | 2986 |
| weighted avg | 0.35 | 0.49 | 0.42 | 2986 |

*Table 5.14. Scores of Phục vụ - PhoBert*

Below [5.8] is the visualization of those results above:

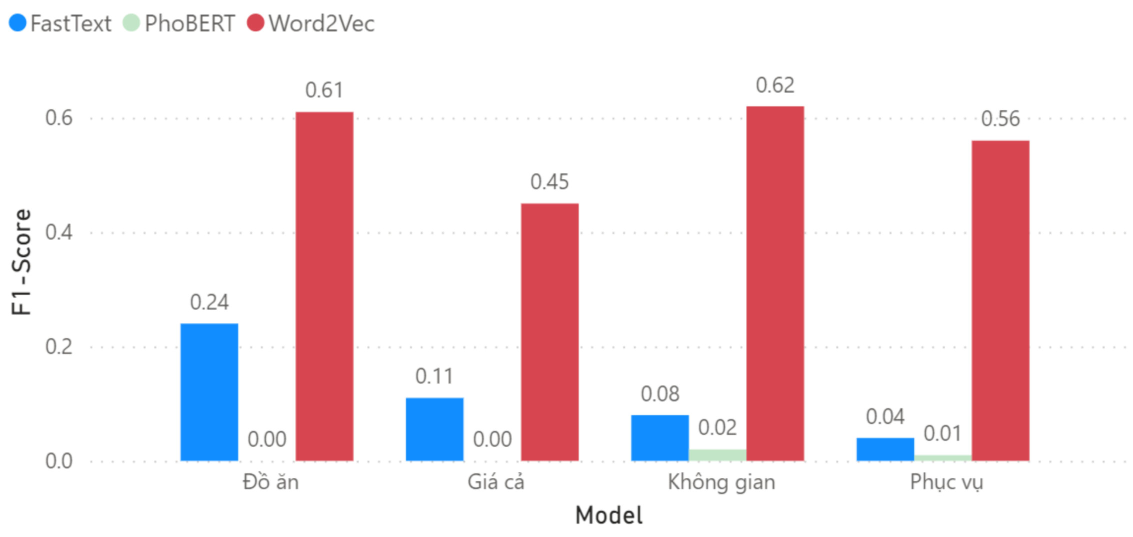
Looking at the table and figures, in contrast to the results of the *FastText* and *Word2Vec* models, the outcomes of the *PhoBERT* model are extremely low, with many



*Picture 5.8. Visualizing Word2Vec model results*

label predictions even falling to a value of 0. This occurs due to the word embedding created by the method utilizing *Masked Language Modeling* (MLM) based on the pre-trained *PhoBERT* model producing very poor results. This directly impacts the training outcomes of the model. Based on these findings, we conclude that using the *PhoBERT* model to generate word embeddings as input for the problem yields **very unfavorable** results compared to the *FastText* and *Word2Vec* models.

Below [5.9] is the sum up visualization of all results above:



*Picture 5.9. Result of comparing 3 different word embedding methods*

After testing all three word embedding methods, *Word2Vec* yielded the best results. Therefore, in the subsequent phase, focusing solely on parameter tuning, the thesis will utilize the *Word2Vec* word embeddings as input for the following steps.

### 5.4.2 Fine-tuning some additional parameters and factors

#### Remove stopwords

During the process of training the model, we observed that for lengthy sentences with a surplus of words, the model's returned result would be *"None"*, indicating that the aspect *"None"* was being assigned more weight, which was not intended. Consequently, we proceeded to remove all Vietnamese stopwords, a step we did not take during the data pre-processing stage. Below are the results of the remove stopwords step:

| **Model** | **Aspect** | **precision** | **recall** | **f1-score** |
|---|---|---|---|---|
| Baseline | Đồ ăn | 0.96 | 0.44 | 0.61 |
| | Giá cả | 0.64 | 0.34 | 0.45 |
| | Không gian | 0.91 | 0.47 | 0.62 |
| | Phục vụ | 0.92 | 0.4 | 0.56 |
| Remove stopwords | Đồ ăn | 0.96 | 0.47 | **0.63** |
| | Giá cả | 0.58 | 0.42 | **0.49** |
| | Không gian | 0.91 | 0.5 | **0.64** |
| | Phục vụ | 0.93 | 0.45 | **0.61** |

*Table 5.15. Compare the results before and after Removing stopwords*

#### Hyper-parameters tuning

For our model, we performed adjustments to the following hyper-parameters:

| **Hyper-parameter** | **Values** |
|---|---|
| Epoch | [2, 5, 10] |
| Batch size | [32, 64, 128] |
| Learning rate | [0.0001, 0.0002, 0.0003, 0.0004, 0.0005] |
| Smooth factor | [0.75, 0.8, 0.85, 0.9, 0.95] |
| Threshold | [0.5, 1, 1.5, 2, 2.5, 3, 3.5] |

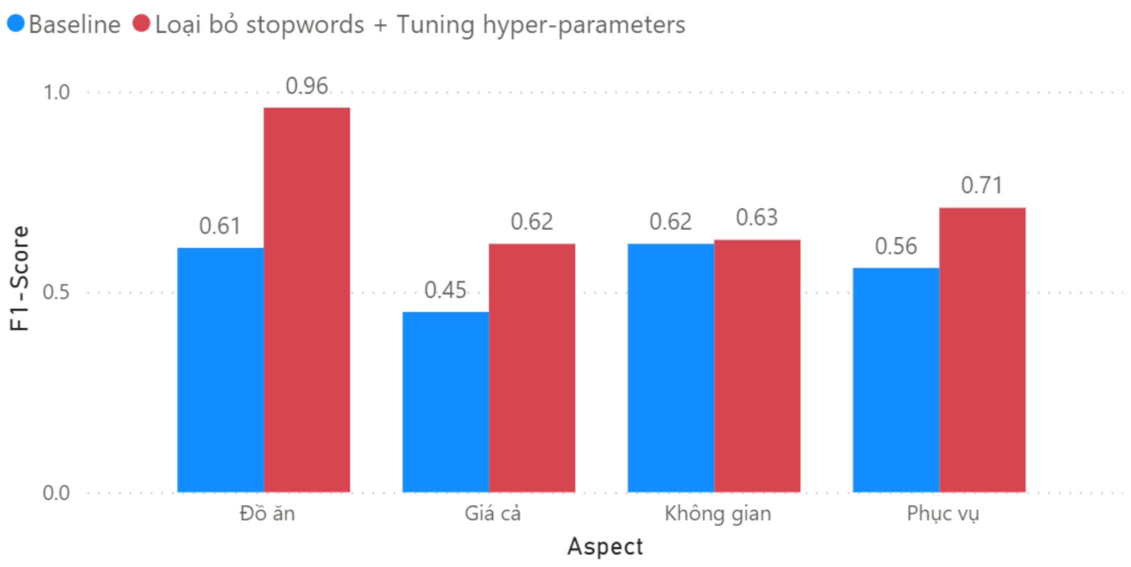
*Table 5.16. Hyper-parameters list*

Below are the results of hyper-parameters tuning:

| **Model** | **Aspect** | **precision** | **recall** | **f1-score** |
|---|---|---|---|---|
| Baseline | Đồ ăn | 0.96 | 0.44 | 0.61 |
| | Giá cả | 0.64 | 0.34 | 0.45 |
| | Không gian | 0.91 | 0.47 | 0.62 |
| | Phục vụ | 0.92 | 0.4 | 0.56 |
| Remove stopwords + Tuning hyper-parameters | Đồ ăn | 0.93 | 0.99 | **0.96** |
| | Giá cả | 0.49 | 0.84 | **0.62** |
| | Không gian | 0.51 | 0.84 | **0.78** |
| | Phục vụ | 0.66 | 0.78 | **0.71** |

*Table 5.17. Compare the results before and after Removing stopwords + Tuning hyper-parameters*

*Table 5.16* demonstrates the best results on the test dataset, with *f1-scores* for each aspect being 0.96 for the *"Đồ ăn"*, 0.62 for the *"Giá cả"*, 0.78 for the *"Không gian"*, and 0.71 for the *"Phục vụ"*.



*Picture 5.10. Compare the results of the model before and after tuning*

Compared to the baseline model, the removal of stopwords and the implementation of hyperparameter tuning lead to significantly improved results. Specifically:

- Regarding the aspect of *"Đồ ăn"*: The *f1-score* is improved by more than *0.35*.
- Regarding the aspect of *"Giá cả"*: The *f1-score* is improved by more than *0.17*.
- Regarding the aspect of *"Không gian"*: The *f1-score* is improved just by *0.01*.
- Regarding the aspect of *"Phục vụ"*: The *f1-score* is improved by more than *0.15*.

Through those results, once again, we observe the effectiveness of the contrastive learning method in handling unlabeled data. Also, it's highlighted that a large model architecture trained on a massive dataset might not necessarily yield superior results compared to a model trained on a specific dataset. The thesis proposes an approach that utilizes feature selection during training to enhance the model's performance compared to conventional models.

# Section 6. Conclusion and Future work

## 6.1 Conclusion

Through the thesis, we can see that understanding customer comments/reviews and the significance of aspect detection play a crucial role in enhancing the quality of products and services. Customer reviews serve as a valuable source of feedback, offering insights into various aspects such as food quality, pricing, ambiance, and service experience. By comprehending these reviews, businesses can identify their strengths and weaknesses, allowing them to make informed decisions for improvement. *Aspect detection*, in particular, holds immense importance as it allows us to automatically extract specific aspects or attributes mentioned in reviews. This process not only aids in categorizing customer opinions but also provides a structured framework for analyzing feedback. By accurately identifying aspects such as *"Đồ ăn", "Giá cả", "Không gian"* and *"Phục vụ"*, businesses can focus their attention on addressing individual areas, thereby optimizing customer satisfaction.

This thesis presents the contrastive learning method, as well as the related techniques required for applying contrastive learning to a problem with unlabeled data. These techniques include embedding algorithms and clustering algorithms. The experimental results on various methods show that embedding the input data significantly impacts the prediction results of the model. This is because the task deals with unlabeled data, and thus, accurate embeddings of words in the dictionary are necessary to assist the model in making accurate predictions. Furthermore, the thesis demonstrates that selecting the top *k* aspects during the model training process helps the model focus on the most relevant aspect groups related to comments, which improves the prediction results of the model.

For this thesis, we conducted the collection and processing of a total of $214,708$ user comments/reviews from real-world scenarios. In the future, we can certainly provide a dataset consisting of these real-life Vietnamese comments to the community interested in aspect extraction for Vietnamese comments.

Subsequently, we performed clustering using various embedding methods, thereby gaining insights into the strengths and weaknesses of each approach in clustering user comments. The best results were achieved with the *Word2Vec* model, which achieved around 0.56 for *F1-score*. Ultimately, we fine-tuned both the *dataset* and *model parameters* during experimentation, leading to significant improvements in results. The average *F1-score* increased from around 0.56 to approximately 0.73.

Additionally, the thesis highlights that the contrastive learning method can be widely applicable to problems with unlabeled data, provided that the input data is sufficiently large to facilitate well-performing embedding models and the construction of loss functions for the model isn't excessively complex.

## 6.2 Limitation and Future work

Despite achieving very good results, it cannot be denied that there are certain limitations in this thesis that could be further improved in the future:

- Lack of experimentation with various clustering methods: The thesis only utilized the *K-Means* algorithm for clustering the dataset. Therefore, there are numerous clustering methods like density-based methods, hierarchical methods, and grid-based methods that could be experimented with and applied in the future.
- Limited diversity in the test dataset: The thesis only selected 4 *"golden"* aspects from the dataset, which indicates a lack of diversity for this aspect classification task. Thus, generating more diverse aspects for testing could be explored in the future.
- Suboptimal activation function tuning: The use of the softmax function reveals a weakness with comments containing many words unrelated to the context. Currently, we mitigate this weakness by removing stopwords from sentences, and this approach demonstrates certain effectiveness. However, we may adjust or change the activation function to achieve the best effectiveness.

## References

[1] Norah AlShammari and Amal AlMansour. Aspect-based sentiment analysis and location detection for arabic language tweets. *Applied Computer Systems*, 27:119–127, 01 2023.

[2] Dingsheng Deng. Dbscan clustering algorithm based on density. In *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, pages 949–953, 2020.

[3] Akhmedov Farkhod, Akmalbek Abdusalomov, Fazliddin Makhmudov, and Young Im Cho. Lda-based topic modeling sentiment analysis using topic/document/sentence (tds) model. *Applied Sciences*, 11(23), 2021.

[4] Geli Fei, Bing Liu, Meichun Hsu, Malú Castellanos, and Riddhiman Ghosh. A dictionary-based approach to identifying aspects implied by adjectives for opinion mining. pages 309–318, 12 2012.

[5] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821, 2021.

[6] Jie Hu, Shaobo Li, Yong Yao, Liya Yu, Yang Guanci, and Jianjun Hu. Patent keyword extraction algorithm based on distributed representation for patent classification. *Entropy*, 20:104, 02 2018.

[7] Ismail Khater, Ivan Nabi, and Ghassan Hamarneh. A review of super-resolution single-molecule localization microscopy cluster analysis and quantification methods. *Patterns*, 1:100038, 06 2020.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[9] Ngoc C. Lê, Nguyen The Lam, Son Hong Nguyen, and Duc Thanh Nguyen. On vietnamese sentiment analysis: A transfer learning method. In *2020 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pages 1–5, 2020.

[10] Baojun Ma, Hua Yuan, and Ye Wu. Exploring performance of clustering methods on document sentiment analysis. *Journal of Information Science*, 43:54–74, 02 2017.

[11] Alfonso Medela and Artzai Picon. Constellation loss: Improving the efficiency of deep metric learning loss functions for optimal embedding, 2019.

[12] Yoan Russac, Olivier Caelen, and Liyun He. *Embeddings of Categorical Variables for Sequential Data in Fraud Context*, pages 542–552. 01 2018.

[13] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015.

[14] Tian Shi, Liuqing Li, Ping Wang, and Chandan K. Reddy. A simple and effective self-supervised contrastive learning framework for aspect detection, 2020.

[15] Dang Van Thin, Vu Duc Nguyen, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. Deep learning for aspect detection on vietnamese reviews. In *2018 5th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 104–109, 2018.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[17] Son Vu Xuan, Thanh Vu, Son Tran, and Lili Jiang. ETNLP: A visual-aided systematic approach to select pre-trained embeddings for a downstream task. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1285–1294, Varna, Bulgaria, September 2019. INCOMA Ltd.