FPT UNIVERSITY

# GRADUATION THESIS

## Enhancing Semantic Search through Domain Adaptation: A Case Study on the Mobifone News Dataset

**Thach Duc Long**

**Nguyen Trieu Ngoc Huyen**

**Nguyen Minh Hieu**

**Bachelor of Artificial Intelligence**

**Supervisor:** Assoc. Prof. Phan Duy Hung _____

**Supervisor's signature**

**Major:** Artificial Intelligence

**University:** FPT University

# TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION....3
1. Abstract....3
2. Problem & Motivation....3
3. Related Works....4
3.1. Lucene and ElasticSearch....4
3.2. Semantic textual similarity....5
3.3. Sentence-BERT....7
4. Contributions....8
CHAPTER 2: METHODOLOGY....10
5. Data Preparation....10
6. Model Training....12
6.1. Multiple Negative Ranking....12
6.2. Model Architecture....13
7. Evaluation Pipeline on Production System Inference Server....15
7.1. System Scenario....15
7.2. Milvus Vector Database....15
7.3. Search pipeline....17
7.4. Evaluation Result....18
CHAPTER 3: CONCLUSION....20
8. Discussion....20
9. Conclusion & Further Works....21
REFERENCES....22

# ABSTRACT

Semantic search is an evolution in both accuracy and flexibility of the search engine. Unlike traditional keyword-based searches, it delves deeper by comprehending the semantics behind words and user queries, resulting in more accurate and relevant search outcomes. In this research, we conduct experiments on a telecommunication domain news dataset to see how its data differs from general model training data and how these differences affect the model feature extraction output and evaluate the model performance in building a search engine.

We will compare different architecture and model used in semantic search task. Furthermore, we conduct investigations of the method to finetune a feature extraction BERT [1] on processed data and the use of Multiple Negative Ranking Loss [2] when the data for the Semantic Textual Similarity training task is not in the ideal format of Premise-Hypothesis-Label.

Finally, we evaluate the model inference performance on a complete pipeline to ensure compelling business requirements. The results show that when data is not ideal, a semantic search model based on Transformers [3] still achieves great retrieval rates on a human-evaluated keywords dataset. We succeeded in creating a highly accurate model while compelling to the required speed on a low-end system with no GPU.

# CHAPTER 1: INTRODUCTION

## 1. Problem & Motivation

Our dataset contains news from a popular telecommunication company in Vietnam. These pieces of news come in the form of promotional SMS, official webpage announcements, internal department reports, etc. All these text data need a solution for customers, support lines, and internal access to search and retrieve in seconds. The existing system is based on a keyword database "LIKE" search. Our mission is to upgrade this system to get a higher search intelligence whilst reducing search time. This can be achieved using an encoder-like pipeline that works by embedding the knowledge base into high-dimensional vectors and retrieving the one with the highest similarity score to the query vector. Some approaches have different types of encoder or data storage options or similarity metrics. In this era of transformers [3], we want to experiment with a bi-encoder based on Sentence-BERT [4] architecture with a modern vector database system. We aim to experiment on how this dataset differs from general model training data and how these differences affect the model's ability to extract semantic features from sentences. We will investigate the method to finetune a feature extraction BERT [1] on processed data and the use of Multiple Negative Ranking Loss [2] when the data for the Semantic Textual Similarity training task is not in the ideal format of Premise-Hypothesis-Label. Furthermore, we address the model performance on different pooling methods. Finally, we evaluate the model inference performance on a complete pipeline to ensure compelling business requirements. The results show that when data is not ideal, a semantic similarity model based on Transformers still achieves great retrieval rates on a human-evaluated keywords

dataset. We succeeded in creating a highly accurate model while compelling to the required speed on a low-end system with no GPU.

## 2. Related Works

### 2.1. Lucene and ElasticSearch

In the old day search system use database “LIKE” search with “%” wildcard to match keyword-based search. The problem of this method was poor flexibility, poor performance and also can not handle Vietnamese accents. To resolve this problem, Apache developed Lucene [5], a search engine based on word one-hotting and TF/IDF technique to find the document with best match at a near realtime speed. And ElasticSearch [6] was developed to ease the problem of interacting with Lucene. Even though Lucene engine and logic was a revolution in text search of both flexibility and performance, the only problem with Lucene was the lack of query understanding. Base off how it works, we can see that when the query contains more stopwords or extra words with low meaning or zero meaning to the query, Lucene can return noisy result. For example, the query “Register account” should return relevant results but the query “I want to Register an account on your website” may brings up some irrelevant news.

### 2.2. Semantic textual similarity

Measuring Semantic Textual Similarity (STS) between words/terms, phrases, paragraphs, and texts is critical in computer science and computational linguistics. STS is required for numerous tasks in natural language processing (NLP), including document summarization, word sense disambiguation, short response grading, information retrieval, and extraction. Advances in natural language processing, distributional semantics, and the availability of large-scale datasets, from early heuristic approaches to sophisticated neural network-based models, have driven the evolution of semantic textual similarity. Early Approaches,

WordNet [7], Distributional Semantics, Zero-shot and Few-shot Learning, Domain-Specific STS, Multilingual STS, BERT and Transformers are some significant milestones and advances in the history of STS. These have increased STS's reach and applicability in various fields and applications.

Furthermore, introducing transformers has had a substantial and favorable impact on the field of STS. Transformers, particularly models such as BERT (Bidirectional Encoder Representations from Transformers) and its successors, have transformed STS in various ways. It enables models to capture contextual information, construct high-quality sentence embeddings, and perform admirably on STS tasks. These models have become the foundation of numerous STS applications, opening up new avenues for analyzing and quantifying semantic similarity in text.

Some well-known datasets commonly used for general purposes include: MultiNLI (Multi-Genre Natural Language Inference) [8] dataset comprises 433,000 pairs of sentences; its scale and data collection approach is designed to resemble SNLI closely and have matched dev/test sets, which are sourced from the same materials as the training set and mismatched sets that are not near similar to anything encountered during training. The key statistics are described in **Figure 1.**

| | #Examples | | | #Wds. | 'S' parses | | | Model Acc. | |
|---|---|---|---|---|---|---|---|---|---|
| **Genre** | **Train** | **Dev.** | **Test** | **Prem.** | **Prem.** | **Hyp.** | **Agrmt.** | **ESIM** | **CBOW** |
| *SNLI* | *550,152* | *10,000* | *10,000* | *14.1* | *74%* | *88%* | *89.0%* | *86.7%* | *80.6 %* |
| FICTION | 77,348 | 2,000 | 2,000 | 14.4 | 94% | 97% | 89.4% | 73.0% | 67.5% |
| GOVERNMENT | 77,350 | 2,000 | 2,000 | 24.4 | 90% | 97% | 87.4% | 74.8% | 67.5% |
| SLATE | 77,306 | 2,000 | 2,000 | 21.4 | 94% | 98% | 87.1% | 67.9% | 60.6% |
| TELEPHONE | 83,348 | 2,000 | 2,000 | 25.9 | 71% | 97% | 88.3% | 72.2% | 63.7% |
| TRAVEL | 77,350 | 2,000 | 2,000 | 24.9 | 97% | 98% | 89.9% | 73.7% | 64.6% |
| 9/11 | 0 | 2,000 | 2,000 | 20.6 | 98% | 99% | 90.1% | 71.9% | 63.2% |
| FACE-TO-FACE | 0 | 2,000 | 2,000 | 18.1 | 91% | 96% | 89.5% | 71.2% | 66.3% |
| LETTERS | 0 | 2,000 | 2,000 | 20.0 | 95% | 98% | 90.1% | 74.7% | 68.3% |
| OUP | 0 | 2,000 | 2,000 | 25.7 | 96% | 98% | 88.1% | 71.7% | 62.8% |
| VERBATIM | 0 | 2,000 | 2,000 | 28.3 | 93% | 97% | 87.3% | 71.9% | 62.7% |
| **MultiNLI Overall** | **392,702** | **20,000** | **20,000** | **22.3** | **91%** | **98%** | **88.7%** | **72.2%** | **64.7%** |

**Figure 1:** Some key statistics of the MultiNLI dataset from the original paper [8]

Microsoft Research Paraphrase Corpus (**MRPC**) [9] is a corpus of 5,801 sentence pairs collected from newswire articles. Each team is labeled if it is a paraphrase or not by human annotators. The whole set is divided into a training subset (4,076 sentence pairs, of which 2,753 are paraphrases) and a test subset (1,725 pairs, of which 1,147 are paraphrases).

### 2.3. Sentence-BERT

Traditional BERT uses a cross-encoder for semantic comparison that accepts two sentences as input and outputs a target value. Due to the massive computations, a cross-encoder is unsuitable in the search engine use case. For example, a search on a knowledge base of n=10,000 documents will take 10,000 model inferences. This number may multiply how the preprocess operation splits each raw document due to the BERT-model maximum accepted sequence length (generally 512 tokens; with popular Vietnamese models, this number is 256).

When first introduced, Sentence-BERT was fine-tuned on NLI data, which is in the format of Premise | Hypothesis | Label. In this process, the model architecture consists of two identical BERT networks with tied weights that produce two output pooled vectors *u*,*v*; a layer concat *u*, *v* and absolute of *u*-*v*; and a layer of Softmax classifier as in **Figure 2**.

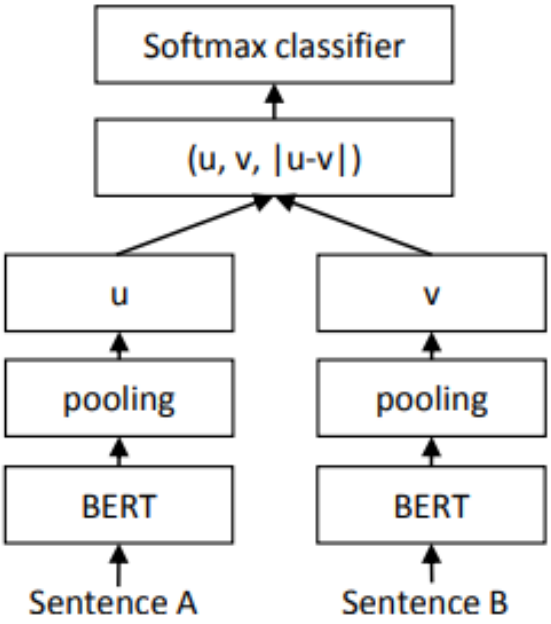


**Figure 2:** SBERT architecture with classification objective function to be fine-tuned on NLI datasets. From the original paper of Reimers and Gurevych [4].

In this paper, we will fine-tune the most famous Vietnamese RoBERTa pre-trained - "vinai/phobert-base" [10] - to an SBERT model on a positive-only NLI-like dataset [11] with a use case-related objective function - Multiple Negative Ranking Loss with cosine-similarity - and evaluate with multiple pooling methods. At inferences, the output of SBERT is two pooled outputs only. A similarity function like cosine-similarity, Euclidean-distance,... will then use these vectors for regression tasks described in **Figure 3**. Using a pre-encoded knowledge base with an optimized index structure can reduce the query time of a search from days to a few milliseconds.

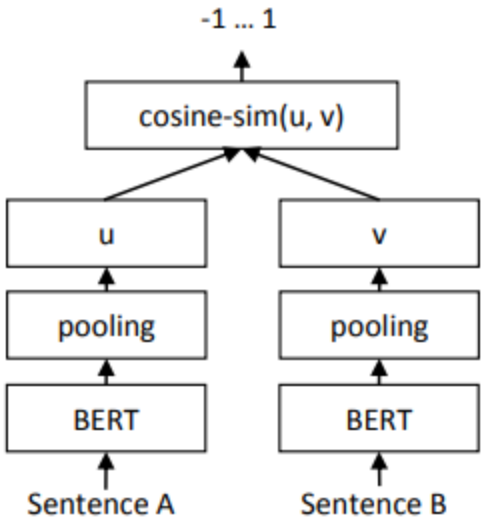


**Figure 3:** SBERT architecture at inference as a bi-encoder. From the original paper of Reimers and Gurevych [4].

## 3. Contributions

Our research shows that using a pre-trained BERT with natural language understanding ability and fine-tuning with the newest sBERT approach is necessary when adapting to unique, distinctive datasets like this telecommunication news dataset. Our model has achieved a raw retrieval rate[1] of ~88% and a human search retrieval rate of 100%. During our workflow, we proposed a data-cleaning pipeline of both manual and automated processes. We fine-tuned a pre-trained RoBERTa architecture with multiple negative ranking set up with a filtered dataset of roughly 5000 examples. During the training process, we conducted optimization methods like hyper-parameters grid search, especially batch size decisions. We balanced the model's natural language understanding and causal ability without needing layers-freezing techniques. The evaluation result proves our confidence in SOTA Transformers architecture on the task of semantic search.

[1] The retrival rate when using test set of training data as search queries

# CHAPTER 2: METHODOLOGY

## 4. Data Preparation

All the data we retrieve is raw data. In the realm of telecommunication company news, the data is a treasure trove of information, encompassing newsletters, promotional announcements, discount programs, incentives, and customer notices. Due to the lack of proper maintaining, there is a lot of news with missing content or description. **Table 2** below shows a brief summary of our dataset.

**Table 2:** A brief summary of our telecommunication news dataset

| Column | Title | Content | Description |
|---|---|---|---|
| News count | 22373 | 4494 | 20259 |
| Average length in characters | 59 | 9371 | 289 |

However, beneath the surface of this textual tapestry lies a chaotic jigsaw puzzle, replete with strange characters, Unicode enigmas, emails, HTML links, Date Time anomalies, and even emojis like in **Figure 4**. While the primary purpose of this data is to convey vital information to customers and consumers, its raw form resembles a complex mosaic of disparate elements, rendering it unsuitable for the intricate demands of business use, especially for sophisticated processing or predictive analysis. The challenge at hand is transforming this raw dataset into a

refined and coherent structure, unlocking its potential as a valuable resource for machine learning education, and ensuring precision in predictions.

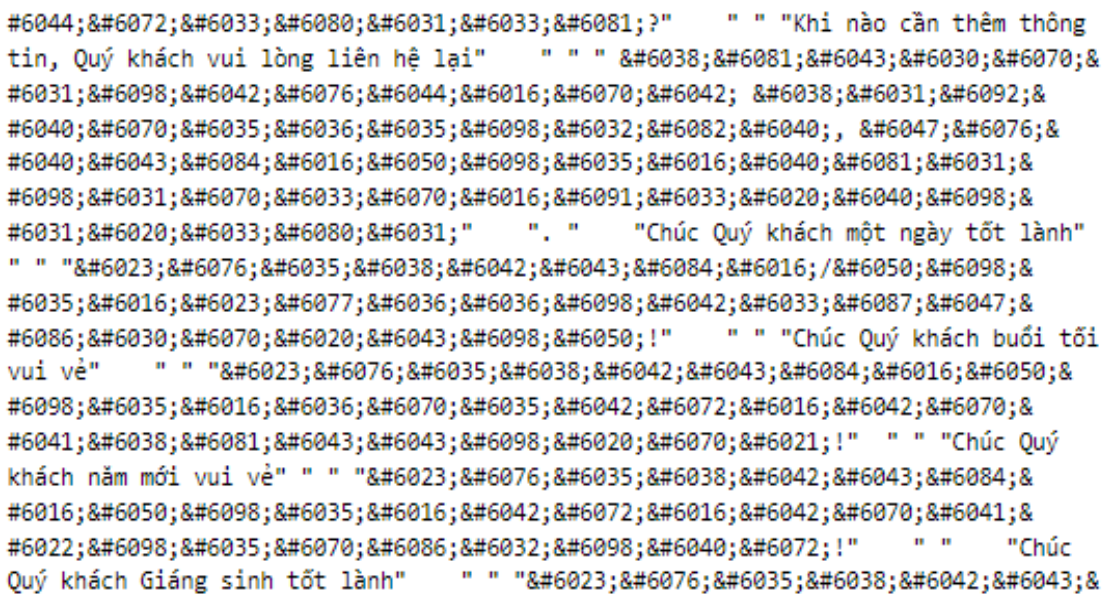


**Figure 4**: An example of original raw data with many spelling errors, confusing characters and Unicode codes

To embark on this transformative journey, Our approach involves a multi-faceted strategy, blending creativity with precision, to tame this unruly dataset. The first step is to aggregate and analyze the raw data comprehensively. This process unveils the underlying patterns and intricacies within the data, acting as a crucial precursor to the subsequent phases. The cleaning process commences with the identification and removal of strange characters, ensuring that the dataset is stripped of unnecessary characters. Unicode characters are harmonized and standardized, transcending language barriers bringing cohesion to the diverse linguistic elements present and removing non-Latin characters. Misspelled Vietnamese words return to correct spelling (wrong position of caret returns to

correct position). Emails and HTML links are carefully parsed, and Date Time entries are refined and then removed to avoid disturbing the data.

The transformative power of this methodology is not confined to merely cleaning the data; it extends to the organization and evaluation of relevant information. By systematically categorizing and structuring the cleansed data, we create a harmonious symphony of information, ready for deployment in a business context or for the rigors of machine learning exploration. The dataset, once a complex puzzle, now emerges as a coherent canvas that simplifies complexity and promotes accessibility.

## 5. Model Training

### 5.1. Multiple Negative Ranking

As our data only contains news titles and content, there will be no contradiction or neutral annotation as in ideal datasets. Therefore, using any classification will halve performance because there will be a class imbalance between positive and negative. We will use Henderson et al. 's idea [2] of batching sentences and marking all non-negative labels with a low score. Henderson's training goal is to minimize the mean negative log probability of the data. In **Figure 6** below, we illustrate the process of adapting our goal using cosine-similarity for inference purposes and cross-entropy as the score function. A scale $S$ is used to emphasize the difference between scores of each $a\_i$ to $p$, and with $n$ as batch size, the training targets to minimize cross-entropy loss. The label is a list with $length=n$, indicating the index of $p$ that $a\_i$ must match.

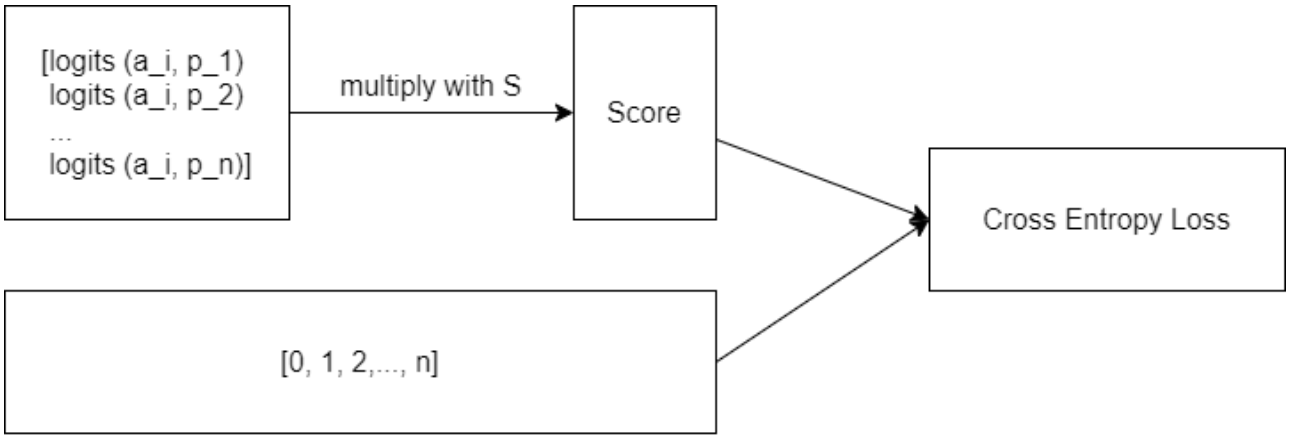


**Figure 6:** Optimization target for training with Multiple Negative Ranking

When using MNR, a larger batch size means improved sample diversity and reduced impact of noise data, which is very helpful since the data we are working on contains many low-meaningful paragraphs (as they have been split down from the main document). In contrast, a larger batch size is also a trade-off between accuracy and training time, as the convergence point will be harder to find, and loss will decrease slower than a small batch size as the model only has to find a set of parameters that suits less negative examples.

### 5.2. Model Architecture

As mentioned in section 1.2, we will use a famous Vietnamese RoBERTa - "vinai/phobert-base" checkpoint to fine-tune our SBERT with the following architecture presented in **Figure 7**.

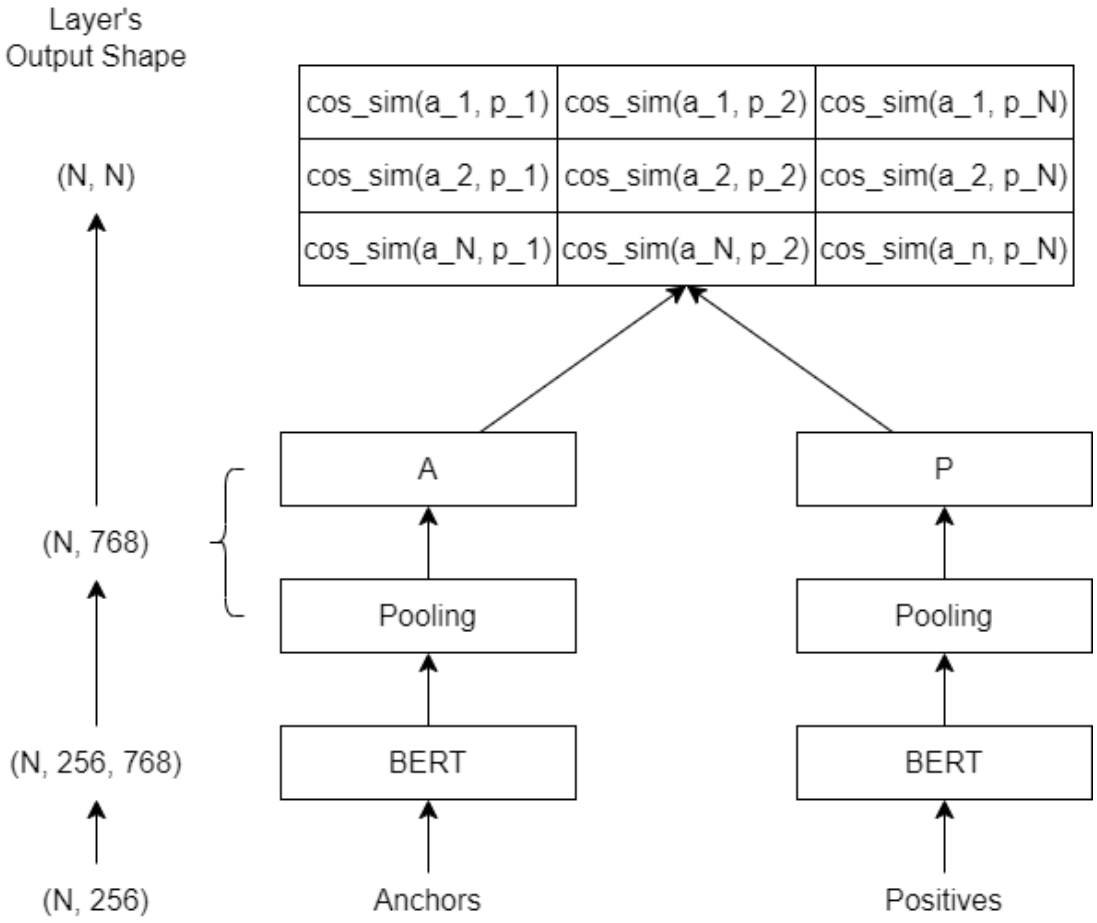


**Figure 7:** Our SBERT fine-tuning architecture with N as batch size. Anchors and Positives are tuples of each batch's *input ids* and *attention masks*.

Our PyTorch implementation loads a single BERT network with pre-trained weights from the HuggingFace model database. This BERT network processes two batches of sentences separately in the training process, creating a "siamese"-like network. Our design guaranteed the identical properties between BERT networks of SBERT as described in **Figure 7**. For further work, this model architecture can improve the model's natural language understanding clarity using a triplet network if the dataset has "negative" annotations.

The benefit of not using classification layers at the end of model architecture helps reduce the computation at inference. It also makes it more convenient to fine-tune the model on downstream tasks, as the final output of the model is a vector embedding.

We set up a PyTorch implementation of the above architecture with Adam optimizer and a simple linear learning rate scheduler with 10% warm-up steps and *dataset length/batch size* total steps.

Additionally, we conducted a naive grid search that we evaluated by loss value. We aim to find an optimal set of parameters with: batch size in [8, 16, 32, 48, 64], number of epochs in [5, 7, 8, 9], and learning rate in [1e-5, 2e-5, 3e-5]. Four models with the lowest final epoch loss value were brought in for further evaluation. We will mention the method below.

The training was conducted on a server with 14 cores 3.30GHZ Intel CPU, and 1xNVIDIA P40 24GB GPU.

## 6. Evaluation Pipeline on Production System Inference Server

### 6.1. System Scenario

The search engine will be running on the following system specifications:

- 4 cores 3.40GHZ Intel CPU with AVX2 support
- 64GB RAM (of which 32GB is available for this search engine)
- Generous volume amount of multiple Gen 3.0 PCI-e NVME SSDs
- NO GPU or GPU turned off

Performance threshold that the search engine must achieve:

- Below 1 second per search query at any system condition.
- Both search engine and API take under 32GB of RAM.

### 6.2. Milvus Vector Database

Milvus [12] is an open-source project introduced in 2019 to store, index, and manage massive embedding vectors. Milvus works with the following flow in **Figure 8**.

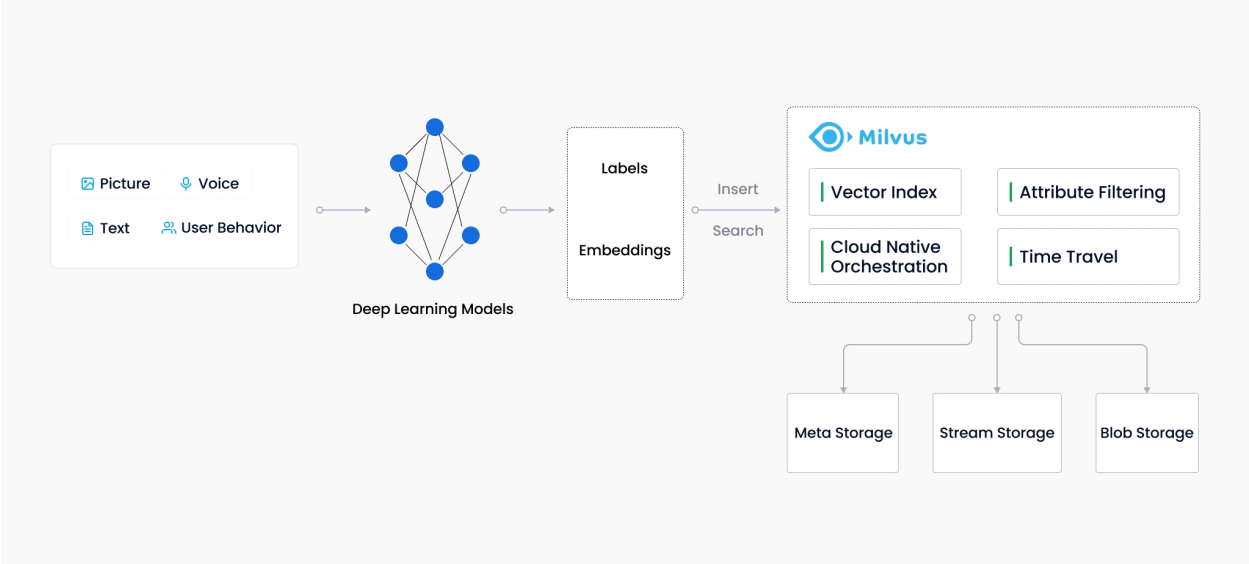


**Figure 8:** Milvus use cases workflow

Milvus provides multiple built-in vector similarity search methods. At the time of this paper, Milvus 2.3.0 released native support for cosine-similarity search. Before this update, users must quantize vectors for the Inner Product metric to use cosine search.

Milvus provides multiple index methods. In this use case, we will take advantage of IVF_FLAT. It uses the approximate nearest neighbor (ANN) [13] search. "IVF_FLAT divides vector data into a number of cluster units (nlist) and then compares distances between the target input vector and the center of each cluster. Depending on the number of clusters the system is set to query (nprobe), similarity search results are returned based on comparisons between the target input and the vectors in the most similar cluster(s) only — drastically reducing query time." When using IVF_FLAT, setting *nlist* at indexing and *nprobe* at

searching is a trade-off between accuracy and speed. The original blog post of Milvus experimented its speed on a dataset of 1 billion 128-dimensional vectors, the result is shown in **Figure 9**.

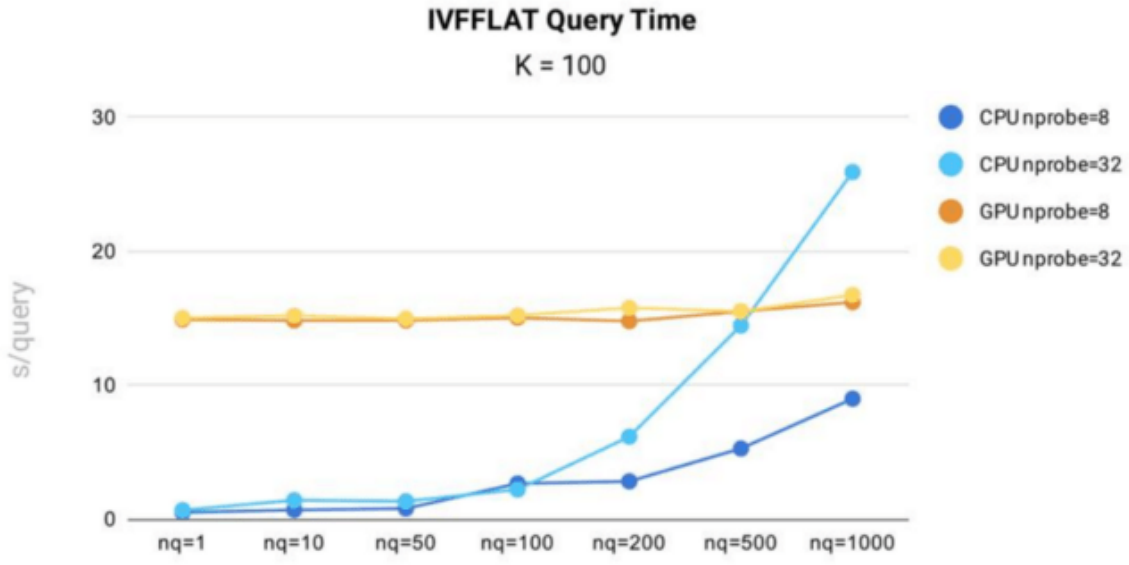


**Figure 9:** Query time test results for IVF_FLAT index in Milvus in Zilliz documentation on a dataset of 1 billion 128-dimensional vectors.

### 6.3. Search pipeline

We first vectorize our entire dataset, insert it into Milvus, and create indexes. As recommended in the Milvus document, we set *nlist* to *4 * sqrt(n)* with *n* as the total number of entities in a segment and conducted experiments to find an optimal *nprobe.*

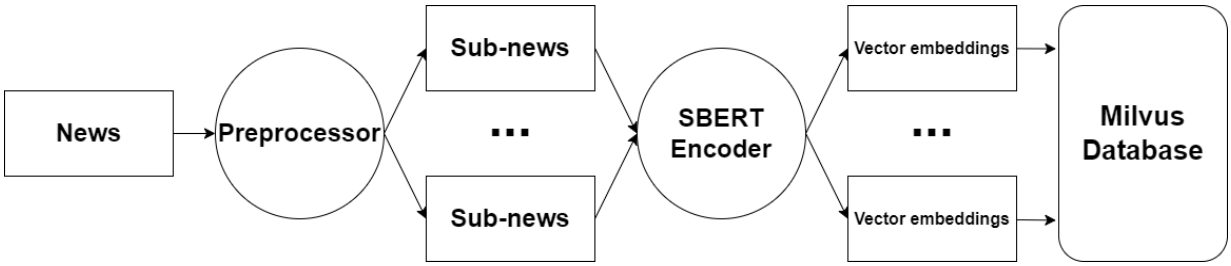


**Figure 10:** The pipeline for inserting knowledge base into Milvus Database

At search phase, the model first vectorizes the input query. Then Milvus helps search that vector embedding in the knowledge base, returning *k* top-ranked results. This k value is often set to 100 for our use case if there are no advanced interference option.

We can set up an evaluation pipeline for model training when we have all the components. We aim to maximize the precision with validation data set aside from the data preparation step. A search is evaluated as a success if the model can retrieve the ground truth label in the top K candidates.

### 6.4. Evaluation Result

Grid search attempts on a similarity search task may be challenging since we finetune a model checkpoint that already has a "fair" knowledge of Vietnamese. When working with causal language models, it is tricky to ensure the model base checkpoint natural language understanding is not lost whilst the ability to learn to predict hypotheses is implemented. This "fitness" can be achieved even before a drastic reduction of loss value (Cross-Entropy Loss in this case) at epoch 1 versus the last epoch. To accomplish our goal, we manually prepared a test set from the same dataset domain but containing completely different topics and target audiences for evaluating NLU and causal ability simultaneously based on our evaluation method.

In **Table 3** below, we set up an ElasticSearch engine, a semantic search system with multiple model and architecture like Word2Vec, pre-trained un-finetuned "phoBERT" [10], pre-trained "keepitreal/vietnamese-sbert" [14]; to compare with our finetuned model. As mentioned above, a search counted as successful if there is the ground truth news within returned results. Due to the fact that database "LIKE" search cannot return any result if using whole title as the query, we did not set it up to evaluation.

**Table 3:** Experiments result on multiple architectures and engines.

| **Model/Engine** | **Retrieval rates** | | | **Average Speed (s/eval)** |
|---|---|---|---|---|
| | **Top 10** | **Top 20** | **Top 50** | |
| ElasticSearch | **72.35** | **75.56** | 75.56 | 35 |
| Word2Vec | 40.28 | 51.65 | 58.92 | 37 |
| vinai/phobert-base | 11.26 | 15.20 | 16.41 | **31** |
| keepitreal/vietnamese-sbert | 22.88 | 28.24 | 35.73 | **31** |
| Our sbert with Batch Size 32 Epoch 8 | 39.98 | 47.67 | 57.68 | **31** |
| Our sbert with Batch Size 32 Epoch 10 | 41.26 | 48.65 | 48.80 | **31** |
| Our sbert with Batch Size 48 Epoch 8 | 48.97 | 58.26 | 72.68 | **31** |
| Our sbert with Batch Size 48 Epoch 10 | 55.25 | 69.20 | **85.21** | **31** |

The average speed of phoBERT and our model can be slow because the tokenizer works base on a word segmenter VNCoreNLP [15] and we did not have the chance to set up a highly optimized pipeline for querying VNCoreNLP. This may affect 20-30% of the model performance since each VNCoreNLP query through a REST API is ~0.3ms. BERT-based models shows better speed due to

lower vector dimension than Word2Vec. The performance of Milvus compare to ElasticSearch achieved a few seconds lower. This may due to the advantage of ANN compare to ElasticSearch inverted index true match.

# CHAPTER 3: CONCLUSION

## 7. Discussion

With the experiments' results, we can argue that trying to fit a model that already has fair language knowledge on a relatively small dataset of a distinct domain is challenging.

First, we simply alter the parameters that affect model "fitness" on the dataset, like the number of training epochs, learning rates, and the optimizer; we did find improvements in loss values. At this time of research, we naively assumed that a better "fitness" means a better model, and we came up with the number of epochs to represent the model "fitness" on the dataset.

Next, we consider the causal reasoning factors. As mentioned above, increasing batch size can help the model's deductiveness because there will be more negative examples to compare. We tried some popular batch size values on a GPU with higher VRAM than our server because 24GB could not exceed 52 as batch size. We found out that the effectiveness of increasing batch size exponentially reduced and accepted the value of 32 or 48. We can argue that this reduction comes from the semantic similarity or representative similarity of training anchors, meaning that when we have similar sentences in the training data, it is impossible, not recommended, inconsequential to make the model learn to positive one over others.

Finally, we evaluate the model's ability to understand queries from natural user language. We set up ~100 examples and evaluated the output news to see if the results matched our expectations through a simple encode and search script with output containing news text.

## 8. Conclusion & Further Works

Even though the result arithmetic values can be below our expectations, our model's performance on human search query results is exceptional. Most of our evaluated search queries based on user history came up with highly relative news. With the architecture of around 135M parameters, we did a good balancing between model retrieval rate and speed (on CPU) after quantizing.

In our research project use case, we fully implemented a search engine that supports semantic search with a good performance with keyword search. The model can be trained once and run for years of new data based on its old knowledge. Although the training process must be run on a high-end GPU system, at inference, the ONNX quantized model [16] can achieve huge performance on just a 4 cores 3.0Ghz CPU.

In the future, we want to conduct some experiments on implementing a learning cycle, creating a base model, and running it on all of our data to create a batching logic to optimize the semantics for anchor sentences in each batch. We will also create a human dataset with queries and news results to evaluate the model's performance thoroughly, and the evaluation pipeline will not be heavily based on arithmetic and mathematical foundations.

# REFERENCES


1. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv.org, Oct. 11, 2018. https://arxiv.org/abs/1810.04805
2. Henderson, M. et al. (2017) Efficient Natural Language Response Suggestion for Smart Reply. arXiv DOI: 10.48550/arxiv.1705.00652
3. A. Vaswani et al., “Attention Is All You Need,” arXiv.org, Jun. 12, 2017. https://arxiv.org/abs/1706.03762
4. Reimers, N. and Gurevych, I. (2019) , Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. , in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Stroudsburg, PA, USA, pp. 3973–3983
5. “Apache Lucene,” lucene.apache.org. https://lucene.apache.org/
6. Elastic, “Open Source Search: The Creators of Elasticsearch, ELK Stack & Kibana | Elastic,” Elastic.co, 2019. https://www.elastic.co/
7. C. Fellbaum, “WordNet,” Theory and Applications of Ontology: Computer Applications, pp. 231–243, 2010, doi: https://doi.org/10.1007/978-90-481-8847-5_10.
8. A. Williams, N. Nangia, and S. Bowman, “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference,” Association for Computational Linguistics, 2018. Available: https://cims.nyu.edu/~sbowman/multinli/paper.pdf
9. W. Dolan and C. Brockett, “Automatically Constructing a Corpus of Sentential Paraphrases.” Available:

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/I05-50025B15D.pdf

10. Nguyen, D.Q. and Tuan Nguyen, A. (2020) , PhoBERT: Pre-trained language models for Vietnamese. , in Findings of the Association for Computational Linguistics: EMNLP 2020, Stroudsburg, PA, USA, pp. 1037–1042
11. Bowman, S.R. et al. (2015) , A large annotated corpus for learning natural language inference. , in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Stroudsburg, PA, USA, pp. 632–642
12. Wang, J. et al. (2021) , Milvus: A Purpose-Built Vector Data Management System. , in Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, pp. 2614–2627
13. W. Li et al., "Approximate Nearest Neighbor Search on High Dimensional Data — Experiments, Analyses, and Improvement," IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 8, pp. 1475–1488, Aug. 2020, doi: https://doi.org/10.1109/tkde.2019.2909204.
14. "keepitreal/vietnamese-sbert · Hugging Face," huggingface.co. https://huggingface.co/keepitreal/vietnamese-sbert.
15. T. Vu, D. Quoc Nguyen, D. Nguyen, M. Dras, and M. Johnson, "VnCoreNLP: A Vietnamese Natural Language Processing Toolkit," 2018. Available: https://aclanthology.org/N18-5012.pdf
16. K. Guo, Y. Xu, Z. Qi, and H. Guan, "Optimum: Runtime optimization for multiple mixed model deployment deep learning inference," Journal of Systems Architecture, vol. 141, pp. 102901–102901, Aug. 2023, doi: https://doi.org/10.1016/j.sysarc.2023.102901.