




Recommendation System for Product Test Failures Using BERT

Xiaolong Sun^{1,2,*}, Henrik Holm^{2,*}, Sina Molavipour^{2,*}^a, Fitsum Gaim Gebre^{2,*}, Yash Pawar²^b,
Kamiar Radnosrati²^c and Serveh Shalmashi²

¹*KTH Royal Institute of Technology, Stockholm, Sweden*

²*Ericsson AB, Stockholm, Sweden*

{sina.molavipour, fitsum.gaim.gebre, henrik.holm, yash.pawar, kamiar.radnosrati, serveh.shalmashi}@ericsson.com

Keywords: Recommendation System, Information Retrieval, Language Models.

Abstract: Historical failure records can provide insights to investigate if a similar situation occurred during the troubleshooting process in software. However, in the era of information explosion, massive amounts of data make it unrealistic to rely solely on manual inspection of root causes, not to mention mapping similar records. With the ongoing development and breakthroughs of Natural Language Processing (NLP), we propose an end-to-end recommendation system that can instantly generate a list of similar records given a new raw failure record. The system consists of three stages: 1) general and tailored pre-processing of raw failure records; 2) information retrieval; 3) information re-ranking. In the process of model selection, we undertake a thorough exploration of both frequency-based models and language models. To mitigate issues stemming from imbalances in the available labeled data, we propose an updated Recall@K metric that utilizes an adaptive K . We also develop a multi-stage training pipeline to deal with limited labeled data and investigate how different strategies affect performance. Our comprehensive experiments demonstrate that our two-stage BERT model, fine-tuned on extra domain data, achieves the best score over the baseline models.

1 INTRODUCTION


A failure record is a document that outlines the details of software tests that have failed during the testing phase. Usually, it contains out-of-range Key Performance Indicators (KPIs), malfunctioning modules, and configuration attributes. To ensure product quality, new features and upgrades are continuously evaluated in test loops at different levels. In this process, it is inevitable to see failures of test cases that require detailed investigations and troubleshooting. However, the number of tests and the complexity of the data pose a big challenge to manual inspections. There has been an increasing demand for improving the efficiency and effectiveness of the testing process.


In light of advances in machine learning, especially in Natural Language Processing (NLP), techniques can be leveraged to identify patterns and trends in failure records. Previous papers (Grimalt et al., 2022; Bosch et al., 2022) proposed solutions to han-


dle duplicate identification in trouble reports, which are more general compared with failure records. In this paper, our goal is to prevent unnecessary replication of test cases and provide better insight into failures for testers. We develop an end-to-end system that can investigate the historical data and recommends a list of similar items given a new failure record. The steps can be broken down as follows:

1. Pre-process the input data with both general cleaning strategies and tailored cleaning strategies adapted to the telecommunication domain;
2. Retrieve a set of top relevant candidates from a large corpus of documents;
3. Reorder the candidates in the previous stage to get a final list of ranked documents.

In order to find a suitable model for our system, we explore both traditional frequency-based models and the state-of-art language model, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). The frequency-based models we use include Term Frequency–Inverse Document Frequency (TF-IDF) (Salton and Buckley, 1988) and BM25 (Sanderson, 2010). The baseline BERT-based

^a <https://orcid.org/0000-0002-6247-1217>

^b <https://orcid.org/0000-0002-6523-1403>

^c <https://orcid.org/0000-0001-9913-3652>

*Contributed equally to this paper.

model we use is TeleRoBERTa (Holm, 2021), a pre-trained RoBERTa (Liu et al., 2019) language model adapted to the telecommunication domain.

TeleRoBERTa is not directly applicable to the text ranking task. Therefore, we build a multi-stage training strategy that incorporates the idea of transfer learning (Torrey and Shavlik, 2010) to improve the performance of our recommendation system. This includes fine-tuning our baseline model on the MS MARCO¹ document ranking dataset (Bajaj et al., 2016), the trouble reports dataset (Grimalt et al., 2022), and our failure records dataset. For this purpose, we use 23.5k trouble reports consisting of internal and external support tickets and incidents with information about fault tags, fault headers, fault descriptions, etc. Failure records, which are the main focus of this work, contain more detailed information about failure incidents during the testing process.

To enhance the interpretability of results, we propose a new metric called Adaptive Recall@K. The final evaluation of our system includes the following aspects:

- The performance of both frequency-based models and BERT-based models in the Information Retrieval (IR) stage.
- The result of models with and without re-ranking in the Re-ranking (RR) stage.
- The effectiveness of different one-stage and multi-stage training strategies.

The rest of the paper is structured as follows. We provide preliminary knowledge of frequency-based models and BERT-based models in Section 2. Section 3 describes our end-to-end recommendation system, including the data, the system architecture, the multi-stage training pipeline, and our adaptive metrics. Then we present our experiments, results, and analysis in Section 4, followed by the conclusion of our paper in Section 5.

2 BACKGROUND

The task of building a recommendation system can be viewed as a text ranking problem. In this paper, we mainly focus on two types of models: frequency-based models and BERT-based neural ranking models.

Frequency-based models utilize classic information retrieval techniques and rely on statistical mea-

¹The dataset has a corpus of 3.2 million documents and 300 thousand queries in the training set, with each query mapped to a positive passage ID that corresponds to its respective document ID within the corpus.

surement to assign weights to each word, whereas neural ranking methods leverage deep neural networks, such as the Transformer-based BERT language model, to produce embeddings of given input sequences. BERT, as a popular state-of-the-art model, offers more precise and nuanced analysis compared to traditional frequency-based models. However, when faced with limited computational resources or training data, frequency-based models like TF-IDF and BM25 still provide a practical and effective approach for text analysis and retrieval tasks. This section provides detailed background information and existing literature for the two models.

2.1 Frequency-Based Models

Frequency-based models determine a term's importance within a corpus by its frequency of occurrence. This is achieved by representing the query and document as vectors, using frequency-based metrics, and assessing their similarity based on shared terms. While these models are valued for their simplicity, efficiency, and interpretability, they typically fail to adequately capture the semantic relationship between words and are sensitive to noise and outliers in the data (El-Din, 2016). Considering these features, it is conventional to use frequency-based models as a baseline for further comparison (Grimalt et al., 2022).

TF-IDF is a widely used algorithm in information retrieval that measures Term Frequency (TF) denoting the importance of each term in a document, and Inverse Document Frequency (IDF) indicating the rarity of the term across the entire corpus of documents.

BM25 is a more sophisticated algorithm that incorporates additional parameters into its ranking formula. Apart from TF and IDF, it also takes other factors into account, such as document length and the average document length in the corpus. BM25 overcomes certain drawbacks of TF-IDF, which include its vulnerability to the influence of rare terms, and its failure to account for variation in the document length. Therefore, BM25 is a more potent method for generating ranking results for specific types of queries and documents.

2.2 BERT-Based Neural Ranking Models

Unlike traditional frequency-based models, neural models learn the relevance of documents to a given query. BERT is one of the most popular models used for neural ranking. It uses a self-attention mechanism (Vaswani et al., 2017) to capture the relationships between words in a sentence or document with two un-

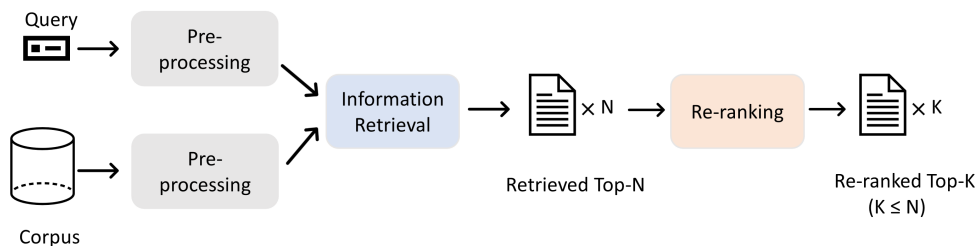


Figure 1: End-to-end recommendation system architecture.

supervised learning tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

In general, a BERT framework consists of two steps: pre-training and fine-tuning (Devlin et al., 2018). With a pre-trained model obtained from the training process above, we can take it as a starting point and fine-tune it for a new task or domain in the transfer learning process (Torrey and Shavlik, 2010). In this paper, we fine-tune our model on a telecommunication dataset using domain adaptation (Farahani et al., 2021) and on a document ranking dataset using sequential learning (Aljundi et al., 2018) to adapt to our use case.

To leverage BERT models in information retrieval systems and semantic search, two architectures have been suggested: the bi-encoder and the cross-encoder. A bi-encoder consists of two encoders that share the same weights and are trained jointly. The first encoder encodes the query sequence, while the second one encodes the candidate sequence in the corpus. After the output embeddings pass the aggregator, usually a pooling layer, each encoder produces a fixed-length vector representation of the corresponding sequence as output. The similarity between the query and the candidate is then calculated by the distance between these two vectors using a similarity function, such as cosine similarity. Sentence-BERT (Reimers and Gurevych, 2019) is one of the most popular bi-encoder models. A cross-encoder encodes both input sentences jointly rather than separately. Mono-BERT and Duo-BERT are examples of the cross-encoder BERT model, where special tokens are added between two sentences to make an input. Besides the BERT model, it includes a final classification layer that takes the output representations and produces a value ranging from 0 to 1 indicating how similar the input sentence pairs are.

Joint encoding allows cross-encoders to capture semantic and syntactic relationships between the two input sentences more accurately (Lin et al., 2021). Therefore, the cross-encoder usually has a better performance than the bi-encoder. However, due to the larger number of parameters and the requirement to encode both input sentences simultaneously, cross-

encoders are typically more computationally expensive than bi-encoders (Choi et al., 2021).

3 END-TO-END RECOMMENDATION SYSTEM

Since both the bi-encoder and the cross-encoder have their disadvantages, a traditional one-stage ranking system usually fails to fully develop their potential. Therefore, a two-stage ranking system can be used to make the best use of their strengths and avoid their weaknesses. Figure 1 presents the architecture of our end-to-end recommendation system consisting of three parts: Pre-processing, IR, and RR. We also introduce a multi-stage training strategy and some improved metrics to enhance the performance and model interpretability. Detailed information is provided in this section.

3.1 Data Pre-Processing

In this paper, we analyze failure records generated during the execution of tests in a Continuous Integration/Continuous Delivery (CI/CD) process. These records contain stored information, including a test case identifier, and logs detailing the faults, activity types, and configurations. We prepare failure records data by concatenating these fields as a data builder and collecting a database with 28k records.

Since the messages in failure records are generated by machines, all detailed information is directly stored, which makes it verbose and contains redundancy or repeated information. However, BERT has a limit on the input sequence length of 512 tokens. Without pre-processing, we risk losing important information as the part exceeding the upper limit is truncated directly. Therefore, our proposed data pre-processing strategy aims to clean the text while preserving the original information as much as possible.

Previous papers looked into the heterogeneity of the log files, which can be a problem to extract useful information with consistent formats (Saneifar et al., 2009). To mitigate such a problem, we develop both

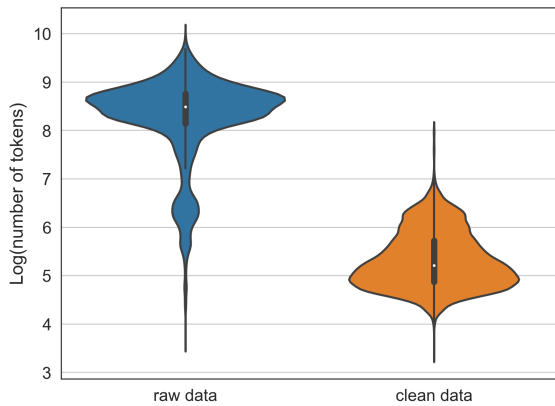


Figure 2: Number of tokens before and after pre-processing. The median values are 4840 and 183 respectively.

general cleaning algorithms and tailored cleaning algorithms:

- In general cleaning, we remove duplicate lines, format whitespace, round decimals, and split different case styles to make them uniform.
- In tailored cleaning, the basic idea is to simplify technical term combinations in the telecommunication domain with the help of experts, but we apply different strategies to different fields to ensure only useful and important information is kept.

For instance, the raw log:

```
[ br />[(SUCCESS), Value of KPI is
5.46459972189E-6 (Criteria $measure >= 1)
[time 2022-10-04 07:22:58], state = 0
[ br />[(SUCCESS), Value of KPI is
5.46459972189E-6 (Criteria $measure >= 1)
[time 2022-10-04 07:22:58], state = 0
```

after pre-processing is converted to:

```
SUCCESS Value of KPI is 5.46E-6;
Criteria: >=1.
```

Figure 2 provides a visual representation of the number of tokens before and after data pre-processing. Note that we apply logarithm to the number of tokens. It's obvious that the average number of tokens among records decreases significantly. In fact, the median value decreases by over 95% after pre-processing, which means our strategy effectively removes many redundant tokens.

3.2 Information Retrieval

In the IR stage, given a search space of size M , an initial set of N documents is retrieved, where $N \ll M$. The primary goal of this stage is to identify the top- N relevant documents related to a user's query while maintaining efficiency and scalability. In other words,

the order of the relevant documents is not the main concern; in fact, the most relevant documents do not necessarily need to be at the top of the retrieved list. Instead, the focus is on quickly processing large volumes of data without significantly sacrificing accuracy. To achieve this, frequency-based and bi-encoder models are employed for the IR stage, given their efficiency. Cosine similarity is used to compare the embeddings and retrieve the top- N relevant records.

3.3 Re-Ranking

In the RR stage, the initially retrieved N documents are re-ordered according to their relevance to the user's query, producing a final list of top- K results, where $K \leq N$ (Grimalt et al., 2022). In contrast to the IR stage, the RR stage places great emphasis on accurately ordering the documents as the model will be evaluated based on both the overall accuracy and specific accuracy, such as top-1. It is worth mentioning that the IR stage successfully narrows down the pool of candidates to the relevant documents, thereby reducing the number of options and facilitating acceptable ranking time.

As explained earlier, cross-encoders are good candidates for re-ranking due to their higher accuracy than bi-encoders. Comparing all pairs of query-candidate for all records is computationally expensive as the inference process requires passing a query and candidate simultaneously through the BERT model. However, as the number of candidates is limited to N , the complexity of the end-to-end retrieval can be controlled by choosing a smaller N . The cross-encoder model is trained with a binary classification and returns a score between 0 and 1, reflecting the similarity between the two documents. Finally, candidates are sorted according to scores computed in this stage.

3.4 Multi-Stage Training

To prepare models, we employ domain adaptation and sequential learning. Domain adaptation transfers a model that was trained on a source domain to a different target domain with different characteristics. Sequential learning focuses on learning and adapting to new tasks without forgetting previously learned information. Figure 3 shows the multi-stage training pipeline, where the plain boxes represent bi-encoders and the dashed boxes represent cross-encoders. For bi-encoders, the proposed approach involves a sequential fine-tuning process for TeleRoBERTa, a pre-trained telecommunications-specific language model that is integrated into both bi-encoder and cross-encoder structures. To further fine-tune the

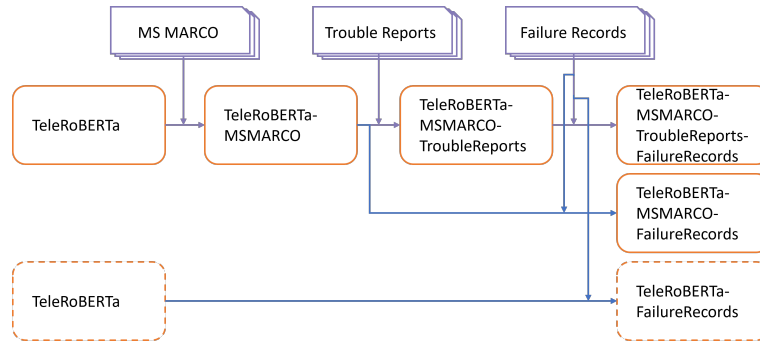


Figure 3: Multi-stage training pipeline. The plain boxes represent bi-encoder and the dashed boxes represent cross-encoder.

bi-encoder model, we use the MS MARCO document ranking dataset for sequential learning. Then, we explore two domain adaptation strategies: one is to directly fine-tune the model on the failure records dataset; the other is to fine-tune the model on the trouble reports dataset and the failure records dataset respectively. For cross-encoders, we investigate domain adaptation of directly fine-tuning TeleRoBERTa on the failure records dataset.

3.5 Model Evaluation Metrics and Interpretability

3.5.1 Adaptive Recall@K

Recall@K is a commonly used evaluation metric in information retrieval and machine learning that measures the fraction of relevant items retrieved in the top K results of a recommendation algorithm. The mathematical formula is:

$$Recall@K = \frac{N_K}{N_{total}}, \quad (1)$$

where N_K represents the number of relevant items retrieved in top K and N_{total} is the total number of relevant items.

However, standard Recall@K is not insightful in our model as the number of relevant records in the ground truth varies widely between samples. As can be seen in Figure 4, certain failure records have more than 100 relevant documents, while other records are linked to only 1 document. To clarify the challenges this creates, consider a sample with 100 related records. Calculating the Recall@10 for this sample can at its maximum return a recall rate of 0.1, even if all 10 retrieved records are included in the ground truth. To mitigate the issues, we propose an adaptive score called Adaptive Recall@K (AR@K). The formula is presented in the next section.

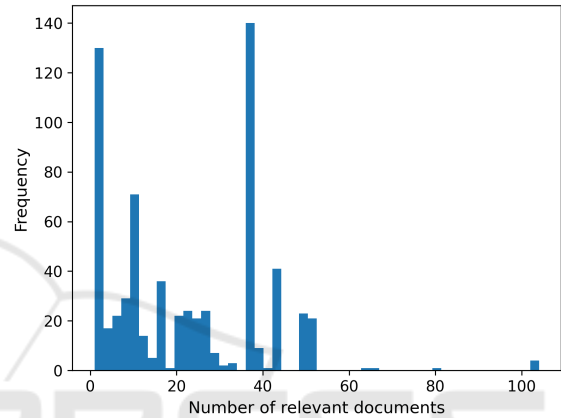


Figure 4: Number of relevant documents for each record.

3.5.2 Improved Model Interpretability

To evaluate our recommendation in an all-round way and improve the interpretability of our model, we introduce the following two matching metrics:

- **Strict Matching:** We want to evaluate the overall accuracy of the model, i.e., what is the percentage of relevant items among the top- K retrieved records? Thus, all K items must be relevant to get the full score. For example, if we have Recall@10 = 0.5, it shows that 50 percent of the top-10 retrieved items are relevant. Mathematically, the recall rate is computed as:

$$AR@K_{Strict} = \frac{N_K}{\min(K, N_{total})}, \quad (2)$$

where the denominator saturates with too many relevant items and avoids the situation of underestimation described above.

- **Single Matching:** We evaluate the accuracy of retrieving from another perspective. Instead of matching all the records, we focus on the possibility of getting at least one relevant item among the top- K retrieved records. Here, only one item needs to be relevant to get the full score. For example, if we have Recall@10 = 0.5, it shows that

we have a 50 percent chance of finding at least one relevant item in the retrieved 10 candidates. Since the matching criterion is a little different, the formula of the recall rate also changes:

$$AR@K_{Single} = \begin{cases} 1, \text{over one relevant in top-}K \\ 0, \text{otherwise} \end{cases}, \quad (3)$$

where we count the recall rate as 1 as long as at least one relevant record is found in the retrieved top- K documents.

For both cases, we calculate the average recall rate over K to get the final result.

Since queries and related target samples are interchangeable, a query can retrieve itself and inflate the scores. Therefore, we detect self-retrieval and do not reward models during evaluation.

4 EVALUATION

In this section, detailed information on the data split and training setups is provided. Then we present and analyze the results under two testing scenarios using our adaptive metrics.

4.1 Dataset Split

In our dataset of about 28k failure records, only 670 records have been labeled by domain experts, meaning that they have been linked to other (similar) records. From these 670 labeled samples, a 50/50 train-test split is created. The number of records in the train and test splits is displayed in Table 1. Moreover, models are evaluated in the following scenarios:

1. **Standard Testing:** In this setting, both query and search records are taken from the test set. In addition to the test set records, we randomly select additional (unlabelled) 1,670 records from the full dataset. These 1,670 records are included in neither the train nor the test split. They are added to the search corpus of the test split to increase its search space, thereby better mimicking a real-world information-retrieval situation. These samples are denoted as *distractors* records as they serve to make the retrieval task more challenging.
2. **Extended Testing:** In this setting, the query records are taken from the pre-defined test set. In this scenario, however, the search records are extended with all unlabelled records in the dataset (i.e., not only the 1,670 distractor records mentioned in the Standard Testing scenario). This introduces more challenge, as the test corpus would

Table 1: Number of records in train and test dataset under standard and extended scenarios.

Dataset	Standard Testing	Extended Testing
Train	336	336
Testquery	334	334
Testcorpus	2,004	27,955

incorporate all available data except for the samples found in the training dataset.

4.2 BERT Training Details

In the IR stage, the bi-encoder TeleRoBERTa model is first fine-tuned on 45k query-document pairs randomly selected from the MS MARCO dataset with 4 epochs. For both the trouble reports dataset and the failure records dataset, the model is fine-tuned with 10 epochs. All fine-tunings use a learning rate of 10^{-5} and the Multiple Negatives Ranking Loss (MNRL) function (Henderson et al., 2017) which is designed for the case when only positive (relevant) pairs are available.

In the RR stage, the cross-encoder TeleRoBERTa model is fine-tuned on the Failure Records dataset with 5 epochs and a learning rate of 2×10^{-5} . For each record, the model is provided together with one positive document sample and three negative document samples. The loss function used is Binary Cross Entropy loss.

4.3 Results and Analysis

The results of the experiments are presented in Table 2, depicting the one-stage and two-stage results using standard testing and extended testing. In general, with an increasing K , the recall rate using single matching increases owing to a wider pool of candidates enhancing the likelihood of retrieving at least one relevant item. Conversely, the recall rate using strict matching decreases since a larger candidate pool makes it more demanding to hit all the relevant items.

4.3.1 Frequency-Based Models vs. One-Stage BERT-Based Models

In the standard testing, the overall recall rates don't show a significant difference between the two kinds of models, while BERT_{MS+FR+TR} has the highest accuracy under both matching scenarios (68.26%). However, as K increases, the difference becomes small, especially when we use single matching. For almost all the models, there is a possibility of more than 80% that at least one of the retrieved 10 records is relevant.

Table 2: Results of frequency-based models and BERT-based models using standard and extended testing scenarios.

Model	Matching	Standard Testing			Extended Testing		
		AR@1 ¹	AR@5	AR@10	AR@1	AR@5	AR@10
TF-IDF	Strict	66.47%	61.63%	58.84%	35.63%	27.87%	27.05%
	Single	66.47%	87.72%	91.02%	35.63%	63.47%	76.65%
BM25	Strict	67.66%	61.54%	59.72%	40.12%	34.84%	33.03%
	Single	67.66%	88.02%	91.02%	40.12%	71.86%	82.93%
BERT ²	Strict	57.49%	48.26%	45.24%	35.03%	28.60%	27.84%
	Single	57.49%	73.95%	79.94%	35.03%	61.98%	70.06%
BERT _{MS}	Strict	65.57%	58.36%	56.23%	42.22%	33.75%	31.78%
	Single	65.57%	85.33%	88.02%	42.22%	68.86%	76.35%
BERT _{MS+FR}	Strict	64.67%	54.47%	49.02%	39.82%	29.30%	26.42%
	Single	64.67%	85.03%	87.13%	39.82%	63.17%	72.46%
BERT _{MS+TR+FR} ³	Strict	68.26%	61.10%	58.06%	39.82%	33.79%	30.85%
	Single	68.26%	87.43%	89.52%	39.82%	69.16%	77.84%
BERT _{MS+TR+FR} + BERT _{FR}	Strict	71.26%	69.28%	64.18%	54.49%	43.54%	39.24%
	Single	71.26%	88.02%	90.72%	54.49%	77.84%	83.23%

¹ AR@K in both strict and single matching modes is averaged over all test queries.

² All BERT models in the table are TeleRoBERTa.

³ MS, TR, FR are respectively short for MS MARCO, Trouble Report, and Failure Record.

We can also observe that the baseline BERT model performs worse than frequency-based models. Similar behavior has been reported in (Cekiç et al., 2022).

In the extended testing, when we try to find the most relevant record, BERT_{MS} outputs the best result on both matching metrics (42.22%). BM25 produces competitive results and surpasses all the one-stage BERT models when we start evaluating the top-5. However, recommending too many documents can be a distraction during the execution of testing, which means we should carefully choose the value for K .

4.3.2 One-Stage vs. Two-Stage BERT-Based Models

We use the cross-encoder TeleRoBERTa fine-tuned on failure records dataset as our second-stage model. The improvement in the recall rate is especially remarkable for a larger search corpus. Compared with the one-stage model, the recall rate of our two-stage model increases by about 3% – 8% in the standard testing and 8% – 14% in the extended testing using strict matching. Even if BERT_{MS+FR+TR} has already achieved satisfactory results in both cases using single matching, we can still observe a gain after the RR stage. All the result proves that the RR stage is able to reorder the retrieved list more accurately.

4.3.3 Comparison of Multi-Stage Models

In the standard testing, the effect of our multi-stage training shows an approximate 11% gain over the baseline BERT model. To be specific, sequential learning makes the greatest contribution with an 8%

increase. In the extended testing, however, BERT_{MS} has the best performance.

We can also observe that BERT_{MS+FR} has a comparatively low increase in the recall rate in both testing cases, indicating that domain adaptation is not as effective as sequential learning. There are two possible reasons accounting for this: 1) high overlap between failure records and telecommunication data makes it difficult to achieve a significant outcome; 2) models tend to forget the previous task and domain when learning the new ones due to catastrophic forgetting (Bosch et al., 2022).

5 CONCLUSION

In this paper, we presented an end-to-end recommendation system for failed test records using a two-stage BERT model, including data pre-processing, IR, and RR. We proposed a multi-stage training pipeline for a pre-trained BERT model to learn different tasks and domains. To enhance interpretability, we introduced an adaptive metric and two test cases. In the IR stage, our BERT model which went through the complete training pipeline outperformed frequency-based models like TF-IDF and BM25, demonstrating the effectiveness of our multi-stage learning strategy. In the RR stage, the two-stage model showed significant improvement over one-stage models, providing highly accurate failure record recommendations. In practice, while BM25-based systems are cost-effective and remain a strong candidate, our end-to-end recommendation system offers a more accurate alternative solution

and is worth pursuing in the long run.

Recent approaches in text ranking attempt to leverage the in-context learning capability of large language models (LLMs). Soft Prompting (Peng et al., 2023) addresses the challenge of insufficient domain-specific training data for dense retrieval by using soft prompt-tuning to generate weak queries and subsequently training task-specific dense retrievers. Pairwise Ranking Prompting (Qin et al., 2023) aims at enhancing the ranking performance of LLMs by reducing the prompt complexities. Exploring the practical implications and potential challenges of these techniques when faced with real-world data remains a promising avenue for future research. Additionally, further benchmark testing can shed light on the comparison between large language models and conventional methods and is considered an extension of this study.

REFERENCES

- Aljundi, R., Rohrbach, M., and Tuytelaars, T. (2018). Selfless sequential learning. *arXiv preprint arXiv:1806.05421*.
- Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., et al. (2016). Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.
- Bosch, N., Shalmashi, S., Yaghoubi, F., Holm, H., Gaim, F., and Payberah, A. H. (2022). Fine-tuning bert-based language models for duplicate trouble report retrieval. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 4737–4745. IEEE.
- Cekiç, T., Manav, Y., Helvacioğlu, B., Dündar, E. B., Deniz, O., and Eryigit, G. (2022). Long form question answering dataset creation for business use cases using noise-added siamese-bert.
- Choi, J., Jung, E., Suh, J., and Rhee, W. (2021). Improving bi-encoder document ranking models with two rankers and multi-teacher distillation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2192–2196.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- El-Din, D. M. (2016). Enhancement bag-of-words model for solving the challenges of sentiment analysis. *International Journal of Advanced Computer Science and Applications*, 7(1).
- Farahani, A., Voghoei, S., Rasheed, K., and Arabnia, H. R. (2021). A brief review of domain adaptation. *Advances in Data Science and Information Engineering: Proceedings from ICDATA 2020 and IKE 2020*, pages 877–894.
- Grimalt, N. M. I., Shalmashi, S., Yaghoubi, F., Jonsson, L., and Payberah, A. H. (2022). Berticsson: A recommender system for troubleshooting.
- Henderson, M., Al-Rfou, R., Strope, B., Sung, Y.-H., Lukács, L., Guo, R., Kumar, S., Miklos, B., and Kurzweil, R. (2017). Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.
- Holm, H. (2021). Bidirectional encoder representations from transformers (bert) for question answering in the telecom domain.: Adapting a bert-like language model to the telecom domain using the electra pre-training approach.
- Lin, J., Nogueira, R., and Yates, A. (2021). Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies*, 14(4):1–325.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pre-training approach. *arXiv preprint arXiv:1907.11692*.
- Peng, Z., Wu, X., and Fang, Y. (2023). Soft prompt tuning for augmenting dense retrieval with large language models. *arXiv preprint arXiv:2307.08303*.
- Qin, Z., Jagerman, R., Hui, K., Zhuang, H., Wu, J., Shen, J., Liu, T., Liu, J., Metzler, D., Wang, X., et al. (2023). Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*.
- Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.
- Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.
- Sanderson, M. (2010). Christopher d. manning, prabhakar raghavan, hinrich schütze, introduction to information retrieval, cambridge university press. 2008. isbn-13 978-0-521-86571-5, xxi+ 482 pages. *Natural Language Engineering*, 16(1):100–103.
- Saneifar, H., Bonniol, S., Laurent, A., Poncelet, P., and Roche, M. (2009). Mining for relevant terms from log files. In *KDIR'09: International Conference on Knowledge Discovery and Information Retrieval*, pages 77–84.
- Torrey, L. and Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.