Fine-Tuning BERT-based Language Models for Duplicate Trouble Report Retrieval

Nathan Bosch *†, Serveh Shalmashi*, Forough Yaghoubi*, Henrik Holm*
Fitsum Gaim*, Amir H. Payberah†
* GFTL GAIA, Ericsson AB, Stockholm, Sweden
† KTH Royal Institute of Technology, Stockgolnm Sweden
nathangerbenbosch@gmail.com, serveh.shalmashi@ericsson.com, forough.yaghoubi@ericsson.com
henrik.holm@ericsson.com, fitsum.gaim.gebre@ericsson.com, payberah@kth.se

***Abstract*—In large software-intensive organizations, trouble reports (TRs) are heavily involved in reporting, analyzing, and resolving faults. Due to the scale of both modern organizations and products, faults are often identified independently by multiple people, leading to duplicate TRs. To mitigate the additional manual effort to identify and resolve these duplicate TRs, prior work at Ericsson focused on developing a 2-stage BERT-based retrieval system for identifying similar TRs when provided a new fault observation. This approach, although powerful, struggled to generalize to out-of-domain TRs. In this paper, we evaluate several fine-tuning strategies to further integrate domain knowledge, notably telecommunications knowledge, into the BERT-based TR retrieval models to (i) attain better performance on duplicate TR retrieval/identification and (ii) improve model generalizability to out-of-domain TR data. We find that adding domain-specific data into the fine-tuning models led to improved results on both overall model performance and model generalizability.**

***Index Terms*—information retrieval, bug reports, trouble reports, neural ranking, catastrophic forgetting, natural language processing, transfer learning, telecommunications**

I. Introduction

In large software-intensive organizations, the reporting, analysis, and resolution of hardware and software faults are crucial to providing stable, high-quality products. Cataloging and sharing these faults are often done through trouble reports (TRs) [1], [2]. However, since faults are often reported independently by different actors, duplicate TRs often arise. Given the high effort required to resolve TRs, identifying duplicates as early in the resolution process as possible is vital. Nevertheless, this process is challenging as TRs mainly consist of natural language text, which may deviate significantly depending on TRs' authors.

To effectively process and perform inference on this text, prior work, BERTicsson [3], leverages two stages of BERT models [4] to perform efficient and effective retrieval of similar TRs (Figure 2). Given a newly written TR, the first stage, which uses a faster but lower accuracy BERT model (Bi-Encoder), identifies the top K most similar TRs. Then, these TRs are re-ranked in the second stage, which uses a more accurate but slower BERT model (Cross-Encoder). The BERT models in these two stages are initially pre-trained on English document ranking data, notably, MSMARCO [5], before fine-tuning on the downstream TR retrieval task.

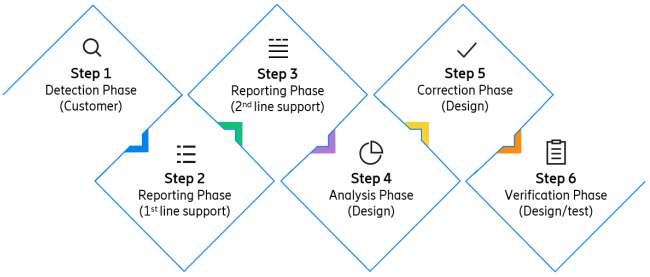


Fig. 1: Process of handling Trouble Reports at Ericsson

In the context of the fault reporting process at Ericsson, outlined in Figure 1, BERTicsson and similar methods can aid both the analysis and resolution phase. Even in the case where a duplicate TR is not present in the data, retrieving TRs with similar faults can still be leveraged to more easily identify software or hardware components in which the fault has occurred and the actors responsible for resolving the fault.

A. Problem Statement

Compared to traditional retrieval approaches, such as BM25 [6], prior work found that BERTicsson [3] performed significantly better when retrieving duplicate TRs. Despite this, BERTicsson has two major drawbacks:

1) BERTicsson does not effectively leverage Ericsson's pre-trained telecommunications-specific language models. These models have been trained on vast amounts of general telecommunications language data. Prior work, [7], has shown that fine-tuning these domain-specific models to other downstream telecommunications-specific tasks has shown strong results. In the existing BERTicsson fine-tuning process, the only telecommunications-specific text provided to the model are the TRs in the final fine-tuning stage, which represents a very small set of the total data available.
2) BERTicsson does not sufficiently generalize to TRs outside the domain of TRs they are trained on, leading to poor performance on TRs written by other customers.

This may be, in part, due to the lack of domain-specific data provided throughout model fine-tuning.

To mitigate the outlined challenges, this research investigates different fine-tuning strategies for training a duplicate TR retrieval model, notably focusing on integrating telecommunications-specific models in the fine-tuning process.

Fine-Tuning Strategies: Figure 9 outlines the different fine-tuning strategies we investigate in this paper, further outlined here:

1) *Domain Adaptation* [8], in which we fine-tune a document ranking model (trained on MS MARCO) to our duplicate TR retrieval task. This is referred to as domain adaptation as we adapt from one domain (English) to another (telecommunications). It is assumed that the task remains the same.
2) *Sequential Learning* [8], in which we fine-tune a general telecommunications language model (e.g., from [7]) to our duplicate TR retrieval task. Another term for sequential learning is task adaptation. In this case, we shift from a general language task to a document ranking task, while the domain (telecommunications) remains the same.
3) Our *Multi-stage approach*, in which we initially fine-tune a telecommunications language model on English document ranking data before fine-tuning on the duplicate TR retrieval tasks. Unlike the prior two approaches, in this scenario we integrate data in both the relevant downstream task and domain prior to fine-tuning on TRs. Note, however, that here the model is sequentially pre-trained on telecommunications data and then English document data. This sequential fine-tuning process can suffer from catastrophic forgetting [9]–[11]. To account for this scenario, we add a catastrophic forgetting mitigation strategy, particularly Elastic Weight Consolidation (EWC) [12], in the first stage of our multi-stage model.

Domain adaptation acts as our baseline fine-tuning strategy, as it is the fine-tuning approach used when training BERTicsson [13]. Sequential learning aims to mitigate the aforementioned challenges, by leveraging a pretrained telecommunications-language model in the fine-tuning process. We hypothesize, however, that by not pretraining on any task-specific (i.e., document ranking) data, we may struggle to attain strong performance on the final ranking task. Hence, our novel multi-stage fine-tuning approach aims to leverage the strengths of both domain adaptation and sequential learning.

Contributions: This paper compares multiple fine-tuning strategies for training language models for a telecommunications-specific task, focusing on how to effectively integrate general (non-task specific) domain-specific data into our model fine-tuning. To our knowledge, we are the first to implement and evaluate a catastrophic forgetting mitigation strategy for effectively integrating tasks and domains in a pretraining strategy for language models. In addition, the insights gained from comparing the approaches outlined in this paper can benefit practitioners.

B. Structure

The remainder of this paper is structured as follows. In Section II, we discuss relevant background research. In Section III, we outline the data, models, and fine-tuning strategies in further detail. In Section IV, we evaluate the performance of the different fine-tuning strategies. Finally, in Section V, we conclude this paper and outline possible future work.

II. Background

In this section, we provide an overview of the existing literature and background research relevant for this paper. This is divided into three sections: 1) Neural Ranking, which is relevant to the models leveraged to attain the results in this paper, 2) Transfer Learning in NLP, and 3) Catastrophic Forgetting.

A. Neural Ranking

Neural ranking refers to ranking (e.g., ranking documents) using deep learning approaches [14], [15]. Before BERT-based methods, this often took the form of representation-based and interaction-based methods. The BERT equivalents are Bi-Encoder and Cross-Encoder models, which are the types of models used in this paper. In Figure 2, we visualize the Bi-Encoder and Cross-Encoder model structure in the context of duplicate TR retrieval.

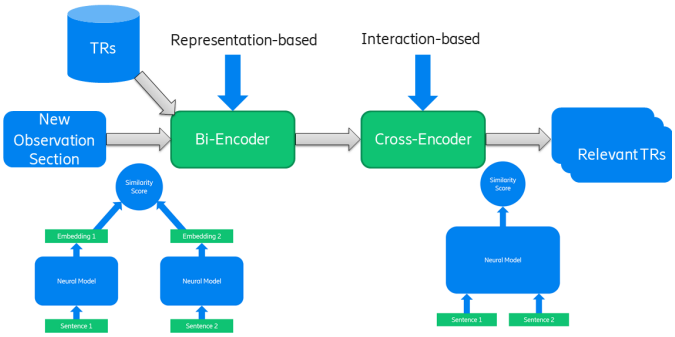


Fig. 2: Multi-stage retrieval process of relevant TRs, including Representation-based and Interaction-based approaches for neural ranking.

Representation-based Retrieval & Bi-Encoders: The **Representation-Based Approach** generally refers to approaches where a neural model produces a high-dimensional embedding of some input text. Metrics, such as cosine-similarity or dot product, can be used to compare the generated embeddings between sentences/documents to identify their similarity. Representation-based approaches can be quite efficient, as once an embedding has been produced for a document it does not need to be computed again.

SentenceBERT [16], is a BERT-based Bi-Encoder model that is very similar to representation-based methods. SentenceBERT can be used to compute the similarity between two documents by 1) generating an embedding using a BERT model and 2) computing the similarity between the embedding

vectors. The model is trained with examples of similar and dissimilar document pairs, and is fine-tuned in a Siamese network training structure [17], where the similarity score between similar documents is maximized, and the similarity score for dissimilar documents is minimized. Other Bi- Encoder models build on or are similar to the sentence-BERT approach, e.g., [18]–[20].

Interaction-based Retrieval & Cross-Encoders: The **Interaction-Based Approach** refers to approaches where the neural model receives two documents (or a query and document) simultaneously and produces a score based on their similarity/relevance. Interaction-based approaches often outperform representation-based approaches [14], [21], [22] as the model will be able to directly focus on the interaction between different tokens in the input query and corpus document.

One of the earliest and most significant Cross-Encoder architectures are the *monoBERT* and *duoBERT* models [23]. In the monoBERT model, the query and document tokens are provided to a BERT model by adding a separator token between them (i.e., "[CLS], query tokens, [SEP], document tokens, [SEP]"). The model is then trained using cross-entropy loss to classify if a query and document are similar or not. Other than monoBERT, there are several other Cross-Encoder architectures, e.g., [24]. This approach can be quite slow, especially as the number of queries and documents increases. Therefore, monoBERT is often only used after an initial retrieval of relevant documents using a faster model. This process is referred to as re-ranking.

B. Transfer Learning in NLP

In traditional machine learning, we generally assume that the domain and task a model was trained for remains static. By domain, we refer to the distribution of the data, e.g., although there is overlap, we consider general English a different domain of text compared to telecommunications or medical text data (different word usage, potentially a shift in grammatical structure). By task, we refer to the structure of model output and its optimization criterion throughout training, e.g., a classification model has very different output compared to a bi-encoder model. We often expect domain and task to remain static for the entire usage of a model. However, in deep learning applications we often see a shift in both the domain and task on which the model is applied.

Notably, in the BERT training framework [4], a model is initially trained on a general language task (with the aim to produce meaningful contextualized token embeddings) before being further fine-tuned to a downstream task/domain. These two stages are generally referred to as pretraining and fine-tuning:

1) *Pre-training:* Unlabelled model training on the base tasks.
2) *Fine-tuning:* Fine-tune all parameters using labelled data associated with the downstream task/domain.

In this framework, a pre-trained model can be fine-tuned to different tasks and domains with minimal architectural changes.

Types of Transfer Learning: [25] and [8] separate transfer learning into two main groups: (i) inductive transfer learning, where a model is fine-tuned on a new task (e.g., fine-tuning a language model trained on masked language modeling to document ranking), and (ii) transductive transfer learning, where a model is trained on a new domain (e.g., fine-tuning a general-domain language model to telecommunications-domain text).

Within transductive transfer learning, [8] distinguishes between domain adaptation, where the model adapts from one domain to another, and cross-lingual learning, where a model is fine-tuned on another language entirely. In addition, in inductive transfer learning, a distinction is set between multi-task learning, where a model is fine-tuned on multiple tasks simultaneously, and sequential learning, where a model is fine-tuned on multiple tasks sequentially.

Transfer Learning in this paper: Besides our multi-stage fine-tuning strategy, this paper focuses mostly on sequential learning and domain adaptation as transfer learning strategies. In domain adaptation, we assume that our source task and our target task are the same but that the domain is different. To fine-tune a model for this scenario, we can essentially just continue training the model as no structural changes are needed [26].

We assume a difference between the source and target tasks in sequential learning. Thus, in most cases some structural change needs to be made to the model (e.g., adding a pooling layer to leverage the BERT token embeddings for the Bi-Encoder or a predictor layer for the Cross-Encoder).

The multi-stage fine-tuning strategy aims to sequentially perform both domain and task adaptation prior to fine-tuning on the final downstream task.

C. Catastrophic Forgetting

When deep learning models are trained on new data, they tend to forget information related to previously seen data in favor of learning new information. This phenomenon, referred to as catastrophic forgetting, represents a major challenge in sequential and lifelong learning (e.g., learning new tasks and domains) [9], [12], [27]. In Natural Language Processing, catastrophic forgetting represents a significant challenge in effectively fine-tuning large language models [8], [28], [29]. Ideally, a model should not lose knowledge of previous domains when fine-tuning on a new domain. For example, a model trained on standard English through masked language modeling should not lose that knowledge when fine-tuned on a smaller subset of medical record data.

D. Elastic Weight Consolidation

There are many strategies for mitigating catastrophic forgetting in deep learning models [12], [30], [31]. These methods aim to find a model parameterization that attains low loss on multiple tasks. We can interpret catastrophic forgetting as the scenario outlined in Figure 3. In sequential training of a model,

TABLE I: Types of Transfer Learning in NLP according to [8]

Class of Learning	Subclass	Description
Inductive Transfer Learning	Multi-Task Learning	Learn tasks simultaneously
Inductive Transfer Learning	Sequential Learning	Learn tasks in sequence
Transductive Transfer Learning	Domain Adaptation	Adapt model domain
Transductive Transfer Learning	Cross-Lingual Learning	Adapt model language

the model shifts from its parameters trained on Task A to new parameters that provide low error on Task B [12].

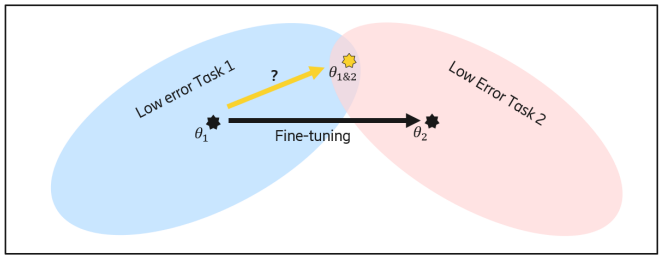


Fig. 3: Outline of what often happens during fine-tuning with and without a catastrophic forgetting mitigation strategy.

In this paper, we leverage Elastic Weight Consolidation (EWC) [12] as the catastrophic forgetting mitigation strategy. In EWC, a quadratic regularization penalty is added to the loss function when training on a new task. This is done to constrain the parameters important for an initial task (e.g., Task A) when learning on a new task (e.g., Task B). In Figure 3, we visualize the shift in model fine-tuning when using a catastrophic forgetting mitigation strategy.

EWC uses the Fisher information matrix, F, [12], [32] to estimate the importance of each parameter in the model. As we see in the following equation, the constraints of the parameters are based on the squared difference between the parameter value on Task A and its current value, multiplied by the importance of that parameter $F_i(\theta_i - \theta^*_{A,i})^2$.

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \quad (1)$$

EWC has been applied to deep language models previously, e.g., [33]–[35]. Although the results of this research does indicate that catastrophic forgetting is mitigated, it does not demonstrate the level of improvement seen in other continuous learning literature.

III. Methods

In this section, we will provide an overview of the dataset used and the preprocessing details involved. After this, we provide an overview of our ranking pipeline, including details on initial retrieval and re-ranking. Finally, we discuss the fine-tuning process of our models, outlining the base models we have available and details regarding our implementation of all of the fine-tuning strategies.

A. Data

A trouble report (TR) is either a primary TR, i.e., ideally the first instance of the fault, or a duplicate TR, meaning that there exists some associated primary TR that described the fault first. We collected a dataset of **23.5K** data points, consisting of **21K** primary trouble reports and **2.5K** duplicate trouble reports. This data was collected from a subset of TRs and spans three years.

TR Structure: Within a trouble report there are several key sections that describe the identified fault. As shown in Figure 4, we look into four sections:

- **Fault Tag:** Outlines in what general region the fault occurred, e.g., it is found to be a radio software issue.
- **Header:** Outlines an initial and short description of the fault
- **Observation:** A much longer section that describes the fault in detail. This often includes log output.
- **Answer:** After the trouble report has been answered, the answer section will be filled with a description of what went wrong and the solution or next steps.

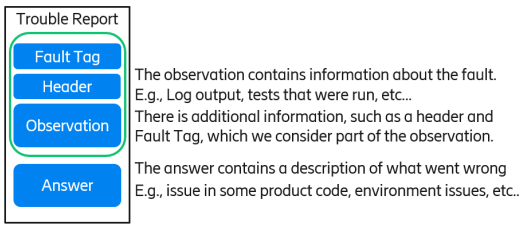


Fig. 4: Overview of the structure of a TR. The key elements are the observation and answer sections. We consider the fault tag and header to be part of the observation section.

The Fault Tag, Header, and Observation section of a TR are provided in the initial fault reporting stages. Hence, we concatenate these sections and simply refer to the combination as the observation section.

Data Exploration: A key challenge when we observe our data, however, is that the length of the observation and answer section may exceed the limit of tokens that our models can take as input. In Figures 5 and 6, we show the number of words in each TR's observation and answer sections, respectively. We find an average of approximately 700 words in the observation section and 200 in the answer section. Although we may expect the answer section to remain within the necessary size for our models, the observation section will often need to be truncated. This runs the risk of losing vital information in our observation section, which may reduce model performance.

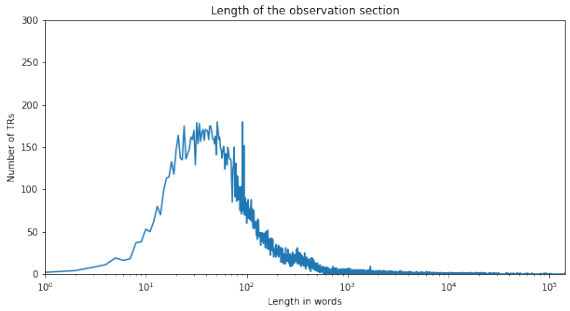


Fig. 5: Number of words in the TR observation sections

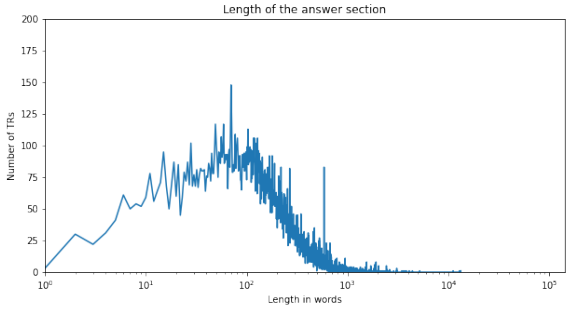


Fig. 6: Number of words in the TR answer sections

Data Preprocessing: We aim to provide the model with as unprocessed text as possible, as we found significant preprocessing (e.g., removing numbers and stop words) reduced the model performance. The preprocessing we did perform was the following (note that this process is performed separately for the observation and answer sections):

1) Extract certain relevant sections of text to reduce the data size. This selection was made based on an existing TR parser. This was necessary to reduce the size of the text for the BERT models.
2) Concatenate the strings of relevant sections from the previous stage.
3) Remove repeating whitespace and newlines.
4) Tokenize the dataset through Spacy [1], a powerful NLP Python library
5) For each token, identify if it matches a known abbreviation (e.g., 5G) and replace it with the original terms.
6) Re-concatenate all tokens to create the final strings.

B. Ranking

As outlined in chapter I, previous work [13] outlined a 2-stage ranking system for duplicate TR retrieval, shown in Figure 2. This ranking approach uses two BERT-based retrieval models: 1) a Bi-Encoder that efficiently retrieves the top 20 most relevant documents and 2) a Cross-Encoder that re-ranks the provided 20 documents. Note that, the top 20 for Bi-Encoder stage is peaked based on experiment conducted on ranking order versus similarity score when the rate of changes in similarity score versus ranking order almost slows down similar to [3], it is $K = 20$.

1) Initial Retrieval with the Bi-Encoder: The initial retrieval process with the Bi-Encoder model can be seen in Figure 7.

The Bi-Encoder model produces an embedding for the TR observation and all texts (TRs) in the corpus. After the embeddings have been produced, the nearest neighbor search is run using cosine similarity as a distance metric to find the 20 closest TRs to the TR observation. We save the IDs of these TRs for the re-ranker stage.

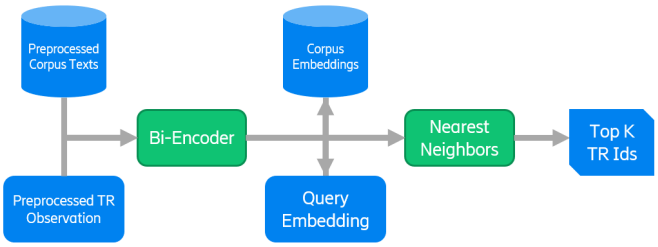


Fig. 7: Initial Retrieval with the Bi-Encoder

2) Re-Ranking with the Cross-Encoder: The re-ranking process is outlined in Figure 8.

Given the top 20 TRs from the initial retrieval, the Cross-Encoder is used to compute a similarity score between each TR and the TR observation query. Finally, each TR is then ranked based on its similarity to the TR observation.

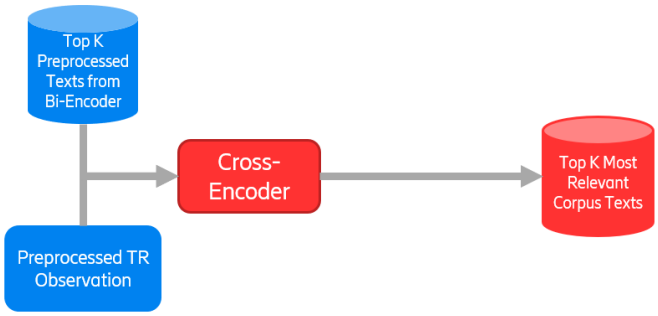


Fig. 8: Re-ranking with the Cross-Encoder

C. Fine-Tuning Strategies

As outlined in Section II, *domain adaptation* and *sequential learning* constitute two common transfer learning strategies in NLP. We refer to the pre-trained model used in the domain adaptation experiments as the MSMARCO model. The pre-trained model used in the sequential learning experiments is referred to as TeleRoBERTa. Both are described below:

- **TeleRoBERTa:** A RoBERTa model originally trained on 160 GB of general-domain English language data [36], but with continued pretraining on 21 GB of general telecommunications data through dynamic masked language modelling [7]. The telecommunications data includes TRs and publicly available 3GPP specifications.

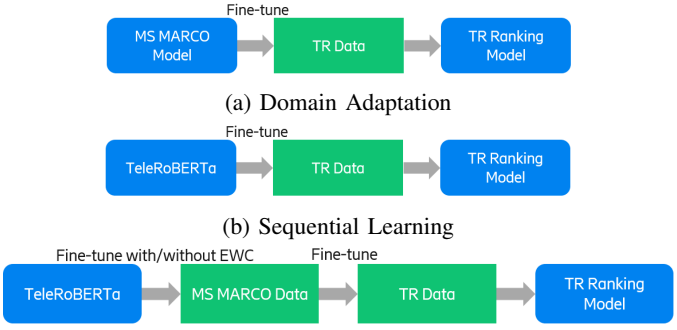
[1] https://spacy.io/

- **MS MARCO model:** A model trained on MS MARCO document ranking data [5]. We have a Bi-Encoder and Cross-Encoder version of the MS MARCO model, both of which are based on the RoBERTa model. The details for how these models are trained is discussed further in section IV-A.

With these models, the *domain adaptation* and *sequential learning* fine-tuning approaches can be seen in Figures 9a and 9b. A challenge with these strategies is that neither takes advantage of both document ranking and telecommunications language data before fine-tuning to the final TR duplicate retrieval task. The third fine-tuning strategy, which we refer to as the *multi-stage fine-tuning approach*, is outlined in Figure 9c. In this scenario, a telecommunications-specific language model is fine-tuned on MS MARCO document ranking data before fine-tuning on the TR duplicate retrieval task, i.e., fine-tuning on both domain and task as part of the pretraining stage.

Multi-Stage Fine-Tuning with Elastic Weight Consolidation: As outlined in Sections I and II, a multi-stage fine-tuning approach can suffer from catastrophic forgetting. To mitigate this, we apply EWC [12] to the initial fine-tuning stage outlined in Figure 9c.

We compute the Fisher information matrix in EWC by collecting the mean squared first order gradients [32] of TeleRoBERTa when we run and evaluate on 7000 randomly selected lines of 3GPP specifications. The gradients are collected when performing the dynamic masked language modeling task.



(a) Domain Adaptation

(b) Sequential Learning

(c) Our multi-stage fine-tuning approach

Fig. 9: The three possible fine-tuning approaches.

In total, we train a Bi-Encoder and a Cross-Encoder model for each of the four scenarios:

- **Domain Adaptation (DA):** Fine-tuning an MS MARCO model on the TR data.
- **Sequential Learning (SL):** Fine-tuning TeleRoBERTa on the TR data.
- **Multi-Stage Fine-Tuning (MS):** Fine-tuning TeleRoBERTa on MS MARCO, then on the TR data.
- **Multi-Stage Fine-Tuning with EWC (MS w/ EWC):** Fine-tuning TeleRoBERTa on MS MARCO with EWC, then on the TR data.

TR Fine-Tuning Process: In accordance with previous work [3], [13], and to fully leverage the vast quantity of primary TRs present in our dataset, we fine-tune the models to produce high similarity scores given a primary observation and a corresponding answer. In other words, provided an observation and an answer corresponding to the same TR, the model is fine-tuned to return a high similarity score. To apply our task to the TR duplicate retrieval task, a new duplicate observation is compared to all primary TR answers or observations, depending on the evaluation scenario.

IV. Evaluation

As outlined in prior sections, we aim to evaluate four different fine-tuning strategies for fine-tuning language models for identifying duplicate TRs. In this section, we outline the experimental setup, evaluation scenarios, and the model results on both initial retrieval and the full re-ranking process (as outlined in Figure 2.

A. Experimental Setup

Dataset Split: All primary TRs with no duplicates present in the dataset are used during training. The remaining primary TRs are split equally to construct the validation corpus and the test corpus. The duplicate TRs associated with the primary TRs are used as queries. i.e., for each primary TR in the validation corpus, we construct a set of queries based on the duplicate TRs that represent a fault already reported in the validation corpus. We refer to these as our validation queries. The same process is done for the test queries. In Table II, we outline the dataset split metrics.

TABLE II: Dataset split sizes

Dataset	Number of Duplicate TRs	Number of Primary TRs	Total
Train	0	18.5K	18.5K
Validation	1.2K	1.2K	2.4K
Test	1.2K	1.2K	2.4K

Evaluation Scenarios: We evaluate the TR duplicate retrieval models under the two following scenarios:

- **Scenario 1:** To evaluate our model's performance at identifying a duplicate TR, we construct the following evaluation scenario: *Given a duplicate TR observation, how well can our model retrieve the primary TR answer?*.
  - i.e., the queries consist of duplicate observations and the corpus consists of the answer section of primary TRs
  - Note that duplicate TRs will inherit the answer from their associated primary TR, hence retrieving primary answers using duplicate answers is not possible.
- **Scenario 2:** We hypothesize that duplicate and primary observations share more information than a duplicate observation and primary answer would. Hence, we also evaluate the models at retrieving the correct TR by using duplicate observations to retrieve primary observations.
  - i.e., the queries consist of duplicate observations and the corpus consists of the observation section of primary TRs

– Note that in this case the models are still trained on producing a similarity score between an observation and an answer section of a TR, we only change how the models are applied.

In both scenarios we use mean reciprocal rank (MRR@K) [37] and recall (Recall@K) [38] to evaluate the models. Recall@K and MRR@K are defined

- **Recall@K:** $\frac{Count(Relevant\ \ Documents\ \ in\ \ Top\ \ K\ \ Retrievals)}{Count(Relevant\ \ Documents)}$. A high recall is a sign that the model is retrieving the necessary documents effectively, but it does not take the rank of documents into account.
- **MRR@K:** The reciprocal rank (RR) is the inverse of the rank of the highest ranked relevant document (e.g., if the highest ranked relevant document occurs as the 4th ranked element, the reciprocal rank is $\frac{1}{4} = 0.25$). Mean Reciprocal Rank (MRR) refers to the mean of the reciprocal rank over a set of queries.

Bi-Encoder Training Details: The MS MARCO Bi-Encoder model is fine-tuned for four epochs on 45k randomly selected query, document pairs from the MS MARCO dataset. The training is done through MultipleNegativeRanking loss function [2], where it is assumed that for all input pairs [(q_0. d_0), (q_1. d_1), ..., (q_i. d_i), ..., (q_N. d_N)], (q_i, d_i) represents a query and document which are similar (i.e., positive) and all other pairs (q_i, d_j) are dissimilar (i.e., negative). The learning rate used is 10^{-5}.

When fine-tuning the Bi-Encoder (TeleRoBERTa or MS MARCO) on the TR data, the model is trained for 10 epochs with a learning rate of 10^{-5}. MultipleNegativeRanking loss is used as well.

Cross-Encoder Training Details: The MS MARCO Cross-Encoder model is fine-tuned for four epochs with a learning rate of $2 * 10^{-5}$ on a subset of the MS MARCO dataset. For 20K queries in MS MARCO, we provide the model one positive document example (i.e., the associated document) and three randomly selected negative documents. This results in 80k positive, negative document pairs. The model is trained through Binary Cross Entropy loss [3]. The same training process is undertaken when fine-tuning the Cross-Encoder to the TR data.

When fine-tuning the Cross-Encoder on the TR data, we again train for four epochs with a learning rate of $2 * 10^{-5}$. Again, we employ a 1:3 ratio for positive and negative samples.

EWC Hyperparameters: The standard MS w/ EWC model is fine-tuned with $\lambda = 10^6$ as we found the best and most stable results under this hyperparameter value.

B. Retrieval Results

Initial Retrieval: The performance of just the initial Bi-Encoder retrieval is outlined in Tables III and IV under the two different evaluation scenarios. We find relatively minimal differences in the performance between models, although it is clear that scenario 2 evaluation leads to much higher performance. This is most likely due to higher overlap in content between the observation sections of TRs, compared to the overlap between an observation and answer section of TRs.

TABLE III: How do the different fine-tuning strategies perform on TR duplicate retrieval (Scenario 1)?

Metrics	DA	SL	MS	MS w/ EWC
Recall@1	10.28%	10.85%	**11.45**%	11.01%
Recall@3	20.77%	19.84%	**21.05**%	20.28%
Recall@5	**26.25**%	25.85%	26.09%	25.12%
Recall@10	35.32%	34.27%	34.76%	**35.73**%
Recall@15	**42.18**%	41.53%	41.57%	40.97%
Recall@20	**46.41**%	45.85%	46.01%	45.24%
MRR@5	16.04%	15.96%	**16.73**%	16.10%
MRR@15	17.78%	17.62%	**18.40**%	17.93%

TABLE IV: How do the different fine-tuning strategies perform on TR duplicate retrieval (Scenario 2)?

Metrics	DA	SL	MS	MS w/ EWC
Recall@1	21.05%	20.69%	**21.33**%	20.60%
Recall@3	32.66%	31.09%	**32.90**%	31.61%
Recall@5	**38.99**%	37.14%	38.06%	38.19%
Recall@10	48.02%	46.77%	47.74%	**48.47**%
Recall@15	**53.75**%	53.15%	53.71%	53.63%
Recall@20	58.23%	58.55%	**58.99**%	57.94%
MRR@5	**27.59**%	26.64%	27.58%	26.99%
MRR@15	29.21%	28.43%	**29.34**%	28.78%

The lack of performance increase of the MS w/ EWC fine-tuning strategy over the MS fine-tuning strategy indicates that catastrophic forgetting may not be a concern for initial retrieval. Despite catastrophic forgetting not severely impacting the performance of multi-stage fine-tuning model, in Table V we do observe catastrophic forgetting take place as we fine-tune TeleRoBERTa on the MS MARCO data. This is mitigated, although not perfectly, by using EWC.

TABLE V: Average loss per batch (over 871 batches) of a model on a subset of the original TeleRoBERTa training data. We show the loss before and after fine-tuning TeleRoBERTa on MS MARCO, with and without EWC.

	Before Fine-Tuning	No EWC	With EWC
Average Loss per Batch	0.649	1.990	1.004

Re-Ranking: As can be observed in Tables VI and VII, the SL and MS w/ EWC cross encoders attained the highest performance. The improvement of MS w/ EWC over the MS fine-tuning strategy also indicated the benefit of leveraging a catastrophic forgetting mitigating strategy when fine-tuning the language models.

Although the improvement over SL is small, when evaluating the models under scenario 2, we find the MS w/ EWC cross encoder to perform the best across the board, indicating that there may be a benefit to fine-tuning on both task and domain before fine-tuning on the final downstream task.

[2]https://www.sbert.net/docs/package_reference/losses.html

[3]https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

TABLE VI: Re-ranker results with MRR@5 using different Bi-Encoders and Cross-Encoders. Evaluation is done under Scenario 1 (i.e., queries consisting of duplicate TR observations and the corpus consisting of primary TR answers). Note that BE stands for Bi-Encoder and CE stands for Cross-Encoder.

Models	CE DA	CE SL	CE MS	CE MS w/ EWC
BE DA	21.28%	**22.98%**	21.19%	22.78%
BE SL	21.60%	**23.80%**	21.54%	23.09%
BE MS	21.87%	23.65%	21.80%	**23.72%**
BE MS w/ EWC	21.49%	**23.73%**	21.77%	23.34%

TABLE VII: Re-ranker results with MRR@5 using different Bi-Encoders and Cross-Encoders. Evaluation is done under Scenario 2 (i.e., queries consisting of duplicate TR observations and the corpus consisting of primary TR observations). Note that BE stands for Bi-Encoder and CE stands for Cross-Encoder.

Models	CE DA	CE SL	CE MS	CE MS w/ EWC
BE DA	35.67%	38.81%	37.99%	**39.48%**
BE SL	35.43%	38.22%	37.47%	**38.62%**
BE MS	36.06%	38.98%	38.06%	**39.21%**
BE MS w/ EWC	36.04%	39.31%	38.50%	**39.45%**

These results indicate that adding prior telecommunications-specific information into the model pre-training process does aid overall performance. The fine-tuning strategy leveraged in prior work [13] attained lower performance than all other strategies.

Generalizability to Out-of-Domain TRs: Thus far, we have evaluated our model on a single dataset in which the trouble reports (train, validation, and test) were sourced from a single domain/group of trouble reports. As outlined previously, however, it is important that the models remain generalizable to other types of TRs. Hence, we aim to identify the performance drop, if any, when evaluating the pretrained models on an out-of-domain TR dataset, notably where we are focusing on TRs created by a different set of operators. We evaluate over 3.4K TRs, of which 1.2K are primary TRs and 2.2K are duplicate TRs.

In Tables VIII and IX, the full re-ranking results are shown for evaluation scenarios 1 and 2, respectively. Although we do observe a decrease in performance compared to prior re-ranking results, this drop is within acceptable bounds. Especially for scenario 2, we only observe a 3.5% drop in MRR@5 performance. Interestingly, in this scenario we observe that the multi-stage re-ranking process (i.e., using both a bi-encoder and cross-encoder fine-tuned in our multi-stage fine-tuning process) achieves equivalent and even slightly better performance than the sequential learning and MS w/ EWC re-ranking approaches under evaluation scenario 2. This could be due to SL and MS w/ EWC overfitting slightly on telecommunications concepts relevant to the TR data it was trained on.

C. Discussion

For initial retrieval, we found no major difference between the performance of the Bi-Encoder models. These findings are in stark contrast to the full re-ranking performance, where a substantial difference between the models could be seen. Notably, the SL and MS w/ EWC Cross-Encoders consistently outperformed the DA and MS Cross-Encoders, both when evaluating under scenario 1 and scenario 2, as can be seen in Tables VI and VII, respectively. This indicates that integrating telecommunications-specific data in the fine-tuning process did provide a substantial benefit in overall model performance.

TABLE VIII: Results on the out-of-domain TR dataset with Scenario 1 evaluation. Note that the models used are bi-encoder/cross-encoder pairs (e.g., DA uses both the domain adaptation bi-encoder and cross-encoder)

Metrics	DA	SL	MS	MS w/ EWC
Recall@1	10.96%	13.63%	12.53%	13.50%
Recall@3	18.70%	21.83%	22.02%	22.43%
Recall@5	23.26%	25.98%	26.02%	26.44%
Recall@10	30.17%	32.70%	31.37%	33.86%
Recall@15	35.15%	36.71%	36.11%	37.36%
Recall@20	37.54%	39.29%	40.40%	40.17%
MRR@5	15.40%	18.16%	17.64%	18.31%
MRR@15	16.69%	19.37%	18.73%	19.59%

TABLE IX: Results on the out-of-domain TR dataset with Scenario 2 evaluation. Note that the models used are bi-encoder/cross-encoder pairs (e.g., DA uses both the domain adaptation bi-encoder and cross-encoder)

Metrics	DA	SL	MS	MS w/ EWC
Recall@1	23.35%	28.42%	28.01%	28.14%
Recall@3	37.95%	42.38%	42.84%	42.47%
Recall@5	43.85%	47.86%	48.83%	48.96%
Recall@10	50.90%	53.11%	54.31%	54.12%
Recall@15	54.72%	56.20%	57.99%	57.35%
Recall@20	56.89%	58.27%	60.29%	59.19%
MRR@5	31.18%	35.83%	36.00%	35.95%
MRR@15	32.45%	36.81%	37.03%	36.92%

The re-ranking results in scenario 2, in Table VII, show that the MS w/ EWC cross encoder marginally attained the strongest results. Although the margins are too small to conclude that the multi-stage fine-tuning strategy improved over a sequential learning fine-tuning strategy, the results do indicate that a multi-stage fine-tuning, in which we sequentially learn relevant elements of the final downstream data (in this case task and domain), can perform well.

We also find that the lowest performing fine-tuning strategy on an out-of-domain dataset is domain adaptation, which is the strategy originally used in prior work [13]. Sequential learning and both multi-stage fine-tuning models maintained more acceptable performance.

Finally, across all models we found that scenario 2 evaluation, in which we produced a similarity score between the observation sections of primary and duplicate TRs, performed the best. This indicates strong generalizability in the models, as the structure of the data provided during inference would deviate from the structure that the model was trained on (most notably for cross-encoders).

V. Conclusion

The trouble reporting process at Ericsson is used to identify, report, analyze, and eventually resolve software and hardware faults. Due to the scale of the organization, however, we often find duplicate trouble reports that, if not identified, represent a significant amount of unnecessary additional effort to resolve. In this paper, we investigate and evaluated how four fine-tuning strategies impacted the performance of RoBERTa-based models for retrieving duplicate trouble reports.

We find that integrating existing telecommunications knowledge through the form of a pretrained telecommunications-specific language model into our fine-tuning strategies allows us to outperform a domain adaptation fine-tuning strategy, achieving state-of-the-art performance on trouble report retrieval. We also attain sufficiently strong generalizability to out-of-domain TR data with all strategies other than domain adaptation.

Although the multi-stage fine-tuning strategy with EWC did not outperform sequential learning by a significant margin, it did demonstrate that catastrophic forgetting mitigation was an effective approach to mix both task and domain in the modelling process.

A. Future Work

There are several future directions for this research. Notably, the work done in this paper is limited to telecommunications-specific language data. Extending this, by investigating the generalizability of the conclusions and methods in this paper, would further contribute to our understanding of deep language models. In addition to this, multi-task learning methods and other lifelong learning strategies could be interesting to explore, to see if similar conclusions hold.

References

[1] L. Jonsson, *Machine Learning-Based Bug Handling in Large-Scale Software Development*. Linköping University Electronic Press, 2018, vol. 1936.

[2] N. Bosch, "Integrating telecommunications-specific language models into a trouble report retrieval approach," 2022.

[3] N. Marzo i Grimalt, "Natural language processing model for log analysis to retrieve solutions for troubleshooting processes," 2021.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] P. Bajaj et al., "Ms marco: A human generated machine reading comprehension dataset," *arXiv preprint arXiv:1611.09268*, 2016.

[6] S. Robertson and H. Zaragoza, *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[7] H. Holm, "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," 2021.

[8] S. Ruder, "Neural transfer learning for natural language processing," Ph.D. dissertation, NUI Galway, 2019.

[9] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[10] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[11] I. Goodfellow et al., "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[12] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[13] N. M. I. Grimalt, S. Shalmashi, F. Yaghoubi, L. Jonsson, and A. H. Payberah, "Berticsson: A recommender system for troubleshooting," 2022.

[14] J. Lin, R. Nogueira, and A. Yates, "Pretrained transformers for text ranking: Bert and beyond," *Synthesis Lectures on Human Language Technologies*, vol. 14, no. 4, pp. 1–325, 2021.

[15] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng, "A deep look into neural ranking models for information retrieval," *Information Processing & Management*, vol. 57, no. 6, p. 102067, 2020.

[16] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[17] D. Chicco, "Siamese neural networks: An overview," *Artificial Neural Networks*, pp. 73–94, 2021.

[18] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.

[19] L. Xiong et al., "Approximate nearest neighbor negative contrastive learning for dense text retrieval," *arXiv preprint arXiv:2007.00808*, 2020.

[20] K. Lee et al., "Latent retrieval for weakly supervised open domain question answering," *arXiv preprint arXiv:1906.00300*, 2019.

[21] L. Pang et al., "Deeprank: A new deep architecture for relevance ranking in information retrieval," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 257–266.

[22] Z. Dai et al., "Convolutional neural networks for soft-matching n-grams in ad-hoc search," in *Proceedings of the eleventh ACM international conference on web search and data mining*, 2018, pp. 126–134.

[23] R. Nogueira, W. Yang, K. Cho, and J. Lin, "Multi-stage document ranking with bert," *arXiv preprint arXiv:1910.14424*, 2019.

[24] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over bert," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 39–48.

[25] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[26] S. Gururangan et al., "Don't stop pretraining: adapt language models to domains and tasks," *arXiv preprint arXiv:2004.10964*, 2020.

[27] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[28] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[29] C. Sun et al., "How to fine-tune bert for text classification?" in *China national conference on Chinese computational linguistics*. Springer, 2019, pp. 194–206.

[30] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3987–3995.

[31] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, 2017.

[32] R. Pascanu and Y. Bengio, "Revisiting natural gradient for deep networks," *arXiv preprint arXiv:1301.3584*, 2013.

[33] Y. Xu, X. Zhong, A. J. J. Yepes, and J. H. Lau, "Forget me not: Reducing catastrophic forgetting for domain adaptation in reading comprehension," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[34] J. Lovón-Melgarejo, L. Soulier, K. Pinel-Sauvagnat, and L. Tamine, "Studying catastrophic forgetting in neural ranking models," *arXiv preprint arXiv:2101.06984*, 2021.

[35] J. Thorne and A. Vlachos, "Elastic weight consolidation for better bias inoculation," *arXiv preprint arXiv:2004.14366*, 2020.

[36] Y. Liu et al., "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[37] E. M. Voorhees *et al.*, "The trec-8 question answering track report." in *Trec*, vol. 99, 1999, pp. 77–82.

[38] D. Harman, "Information retrieval evaluation," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 3, no. 2, pp. 1–119, 2011.