A Token-Prioritization Strategy for Handling Data Imbalance in Network-Change Ticket Classification

Md. Shamim Towhid[*], Nasik Sami Khan[*], Nashid Shahriar[*], Massimo Tornatore[†], Raouf Boutaba[‡], Aladdin Saleh[§]

[*]Department of Computer Science, University of Regina, {mty754, nku618, nashid.shahriar}@uregina.ca
[†]Politecnico di Milano, massimo.tornatore@polimi.it
[‡]David R. Cheriton School of Computer Science, University of Waterloo, rboutaba@uwaterloo.ca
[§]Rogers Communications Canada Inc., aladdin.saleh@rci.rogers.com

***Abstract*—Changes are an integral part of the day-to-day operation of large telecommunications networks as they allow to keep pace with technological advancements, meet growing network demands, ensure scalability, enhance security, improve service quality, and meet customer expectations. Changing configurations, installing devices, and migrating traffic are some examples of these changes. These changes are documented by opening tickets through a ticket management system. Automation in the ticket management system is now becoming highly desirable to manage the large number of submitted tickets. An automated ticket management system supports the management of a ticket by automating several parts of a ticket's life cycle. In this context, ticket classification problem consists in assigning an appropriate label to a ticket to be utilized in the later stages of the ticket management cycle. In this paper, we use a collection of network-change tickets from a real network operator to solve a ticket classification problem. We observe that the network-change ticket dataset is highly skewed in the number of tickets for different possible classes. We address this challenge of classification in a highly imbalanced dataset by proposing two token-prioritization strategies along with other components. We compare three variations of our proposed approach with three methods from the literature and show that the variations of the proposed approach outperform existing methods by up to 7% in terms of F1 score.**

***Index Terms*—ticket classification, imbalanced data, token prioritization, deep learning, automation**

I. Introduction

Tickets describing network changes are commonly used to manage changes in a complex telecommunication network. Opening tickets for network changes enables efficient management, documentation, accountability, collaboration, and compliance within an organization. It supports smooth operations, minimizes disruptions, and ensures the network is maintained effectively. The information recorded in a submitted ticket for a network change may vary depending on the specific processes and systems used by an organization. The most commonly recorded information in a Network-Change Ticket (NCT) is the requestor information, the description of the changes, the level of priority, and the information related to the approval and authorization process. Once a ticket is submitted by the requestor, it goes through an authorization process. If the changes are approved and authorized by the designated employee, the ticket goes to the execution phase.

A large number of NCTs are submitted on a daily basis in a large and complex network, so an automated ticket management system is essential to manage these tickets in an efficient and effective manner. Authors in [1] mention five ways an automated ticket management system can be beneficial to an organization. One crucial aspect of automated ticket management is *ticket classification*, i.e., the process of assigning a label to a submitted ticket. This label can be based on the priority of the changes, required problem resolutions, or possible impact of the suggested change on the network.

In this paper, we leverage a repository of a large number of submitted NCTs from a major telecommunication operator in Canada. The life cycle of an NCT in the existing approach is shown in the upper part of Figure 1. During the submission of an NCT, the submitter assigns a network-impact label to the NCT based on his/her initial assessment and domain expertise. This network-impact label captures the probable impact on the network during the execution of the NCT. There are four types of network impact in our collection of NCTs, namely, "Outage", "Threatened", "Degraded", and "No Impact". The priority of an NCT during the authorization phase is decided based on the assigned network impact label by the ticket submitter. For example, a ticket with "Outage" as network impact is given more priority than a ticket with "No Impact" as network impact. Furthermore, this network-impact label helps to make informed decisions during the execution phase of an NCT. If a ticket submitter predicts the network impact as "Degraded" or "Threatened", but the changes lead to an "Outage", it is critical to the organization, and there can be negative consequences to the implementer of the NCT. Therefore, It is crucial to assign the correct network-impact label to the NCTs during the submission. We develop a new Deep Learning (DL) model to predict the network-impact label by taking the NCT description as input. The model prediction should assist the human assessment of network impact during the submission of an NCT, increasing the confidence of the submitter. Since the model is trained on previously submitted NCT tickets, it learns to predict the network impact based on the description of the changes. The lower part of Figure 1 shows our proposed approach to predict the network impact label using the proposed DL model.

In our dataset, network changes are summarized in two columns: "Headline" for a brief overview and "Description" for details. We concatenate these columns to input our DL model, which must address specific dataset and NLP challenges for accurate network impact prediction.

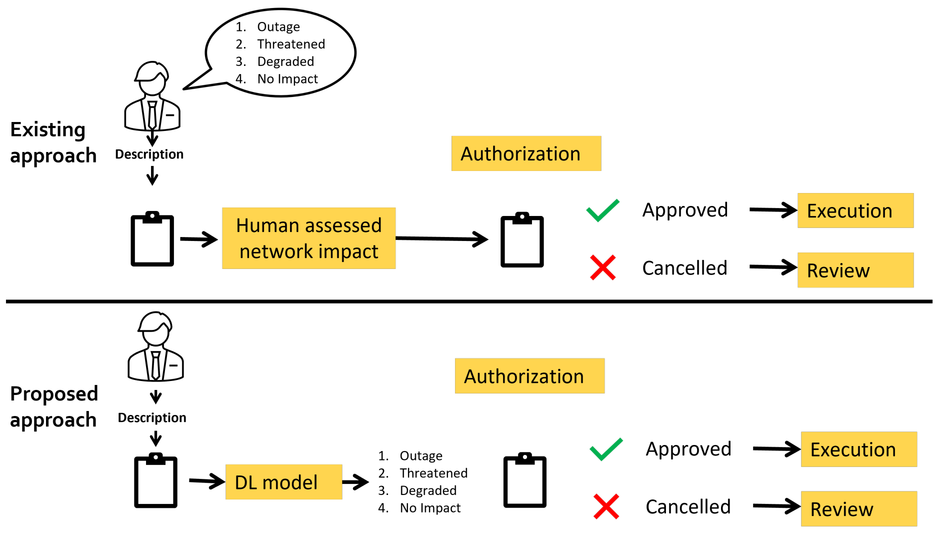


Fig. 1: Existing ticket management process and our proposed approach

- The dataset is highly imbalanced, as shown in Figure 2. As our dataset is from a real network, more than 50% of the NCTs belong to the "No Impact" class because an event like "Outage" is rare in the real network. The DL model will be biased towards the majority class if we train a model on this imbalanced dataset.
- The dataset contains many domain-specific words (e.g., IPRAN, QAM, LTE, etc.). In text classification, a common practice is to use a pre-trained model available in public and fine-tune the pre-trained layers for a specific task [7]. Because of these domain-specific words, pre-trained models do not perform well on our dataset.
- A lot of words in our dataset provide context for the change description but are not relevant for the classification of network impact. Keeping these words may cause the model to overfit the data.
- Since the description of an NCT is written by humans, there will be spelling errors and ambiguity in the data. Non-meaningful characters like repeated punctuations and special characters (***, !!!, &) and HTML-like tags are also present in the dataset. The presence of ambiguity in the data and the inconsistency of punctuations pose challenges in learning patterns and accurately identifying the intended class of a ticket.

In our proposed solution, we leverage a transformer-based state-of-the-art model named BERT, and we propose several new components in our solution to handle each of the mentioned challenges, as outlined below.

- To handle the data imbalance issue, we propose to use augmentation strategies combined with token prioritization. As for augmentation, we apply both textual and numeric augmentation processes mentioned in [5]. Our proposed token-prioritization strategies help to improve the overall accuracy of the model by giving priority to the most important words in a ticket.

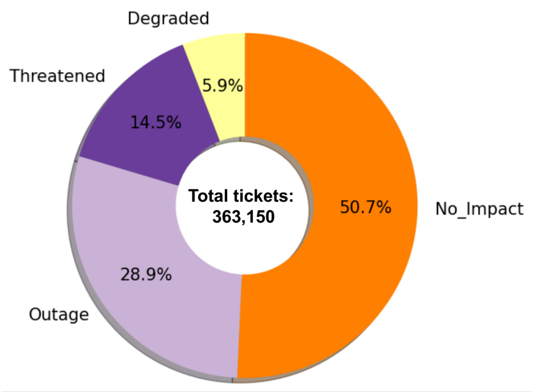


Fig. 2: Class distribution of our NCT ticket dataset

- Our contribution involves fine-tuning BERT's pre-trained tokenizer and transformer layers to effectively handle domain-specific words, addressing the issue of context-related words through the use of volatile tokens [2].
- We use regular expressions and other text processing techniques (removal of stopwords and filtering text) to handle the issues of non-meaningful characters in our dataset [4].
- Through extensive experiments, we show that the proposed approach outperforms existing methods by 7% in terms of the F1 score. Furthermore, our proposed approach is able to achieve higher F1 scores for minority classes compared to existing approaches.

The remainder of the paper is organized as follows. Our literature review is presented in Section II, where we discuss existing works in the literature. Section III provides a detailed description of the components of our method. In Section IV, we present the results of our proposed approach in comparison with three existing approaches from the literature. Section V concludes the paper by providing a summary of our work and discussing future research directions.

II. Related Works

In this section, we discuss state-of-the-art research focusing on research questions that are similar to ours, such as using data augmentation for solving class imbalance in text classification, selecting relevant important features, and adaptation of custom-domain knowledge.

A. Data Augmentation

The work in [2] investigates several data augmentation strategies to solve the data imbalance issue. The authors discuss the effectiveness of sampling a subset of the dataset for augmentation, the efficiency of introducing augmented examples gradually during training, the significance of choosing the right sentence pairs, and the potential of sampling techniques like the Synthetic Minority Oversampling Technique (SMOTE) [16] to address the class imbalance in classification models.

To address the class imbalance, the limitations of a unique oversampling method using SMOTE for high-dimensional binary datasets have been discussed in [3]. In [3], the authors propose a novel distance metric that analyzes various ranking schemes and considers a selection of pertinent features. In high-dimensional space, it performs better than SMOTE and other alternatives.

Authors in [4] compare various open-source libraries and advanced NLP frameworks for text processing. They focus primarily on the NLPAug [15] library and evaluate its usage to enhance the processing of textual datasets for training conversational chatbots and other NLP applications. Similar to NLPAug, authors in [5] discussed another approach to data augmentation, named Easy Data Augmentation (EDA) which uses straightforward textual modifications to produce new training samples. It involves the four fundamental processes of synonym substitution, random word addition, random word removal, and random word switching. On average, the EDA method improved model accuracy by 4.5% across five benchmark datasets, with individual increases ranging from 0.9% to 9.4% based on the dataset and model used. A thorough overview of data augmentation for NLP tasks depending on various scenarios can be found in [6]. The overview examines several augmentation techniques and emphasizes the importance of data augmentation in several NLP tasks. When selecting a suitable augmentation technique, the authors stressed the importance of carefully considering the data quality, the intended result, and the task limits.

B. Feature Extraction

In a text classification problem, selecting the best features from the words in numeric space is a vital task that impacts the overall result of the classification model. Authors in [7] present a method for selecting the first token using the BERT model, which is used to better capture the linguistic nuances and increase performance on a wide range of NLP tasks. The study in [8] employs a combined strategy of numerical augmentation and the BERT model to address the problem of data imbalance. We compare this approach with our proposed approach in this research.

Another common way to concentrate on data imbalance issues is discussed in [9] by employing a weighed loss mechanism, which we combined with BERT in our research. Authors in [10] introduce a three-way model that divides the description space into confirmatory, disconfirmatory, and neutral regions for evaluating confirmation in classifications. However, the proposed method is limited for binary classification in [9]. One of the token prioritization techniques in our research is inspired by this study.

C. Domain Adaptation

To classify domain-specific log messages, the study in [11] enhances the word embedding-based neural network's adaptability by focusing on domain-specific vocabulary, word-level and character-level information, training coverage, and various approaches of word embedding. The approach divides log messages into templates based on contextual similarity and uses volatile tokens to generalize similar words with minor differences by masking the keywords in place of volatile tokens. This helps analyze and effectively categorize log messages by reducing feature dimensionality. Another method for fault localization that employs word embeddings to convey semantic relationships in IT infrastructure event data is discussed in [12]. The method uses transfer learning to cluster extracted vectors based on semantic similarity, enhancing online fault detection by gradually adding domain-specific token sequences to generic word embeddings, resulting in high efficiency without recreating knowledge bases.

Authors of [13] propose a multimodal deep-learning framework for the classification of short texts into multi-classes using an imbalanced and extremely small dataset, which achieved an F1-score of 85% and an accuracy of 86.65%, surpassing existing models. It uses a five-layer architecture consisting of a DistilBERT [13] model for word embeddings, LSTM network layers for deep semantic information, and SoftMax and max-pooling layers for multiclass classification. This efficient method is competitive for small datasets and lightweight for mobile device deployment. In this paper, we leverage both down-sampling and up-sampling strategies using numeric and textual augmentation to tackle data imbalance. We explore the use of a probabilistic relevancy score based method which performs well with imbalance data, and the Term Frequency-Inverse Document Frequency (TF-IDF) method for token selection, finetuning pre-trained model, which are different from this existing research.

III. Methodology

In this section, we discuss our proposed approach. The overview of all the steps in our proposed approach is shown in Figure 3. Our contributions to address the challenges in our dataset are categorized into three subsections: Pre-processing, Domain adaptation, and Token prioritization.

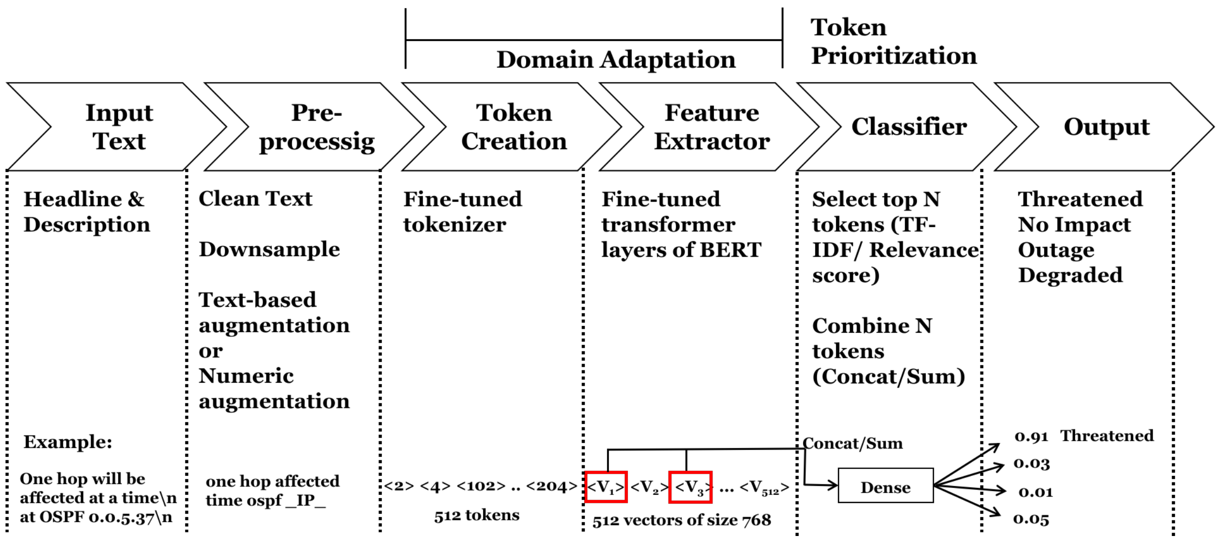


Fig. 3: Overview of the proposed approach

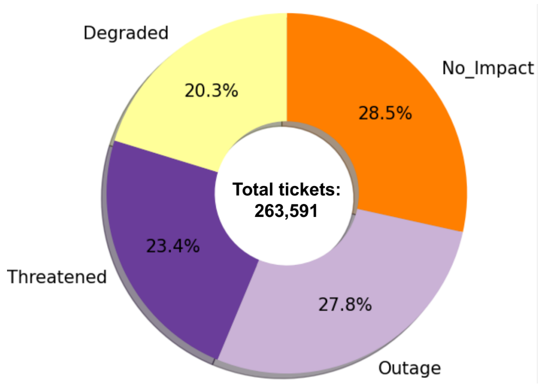


Fig. 4: Class distribution after textual augmentation

A. *Pre-processing*

The first step in any text classification problem is to clean the text. Human written text may contain many characters (e.g., punctuation marks) which are not particularly useful for classification. We perform the following operations as our pre-processing steps.

Text cleaning: We start by converting all text to lowercase to remove case sensitivity. Next, we address challenges in the data, including NULL and duplicate values, and exclude test tickets with “Test” in the “headline” column. We then utilize regular expressions to eliminate repeated punctuation, non-meaningful characters like new lines and HTML tags, and remove common stop words (e.g., ’a’, ’an’, ’the’). We notice numerous non-relevant words, such as IP addresses, employee names, employee email addresses, and URLs, in our dataset, which contribute to sentence context but do not aid in NCT ticket classification. Instead of deletion, we follow the approach outlined in [2] to replace them with a designated keyword referred to as a “volatile token”.

Random downsample of the majority class: After text cleaning, we aim to balance our dataset. As depicted in Figure 2, most NCT tickets are in the “No Impact” class. To rectify this, we randomly trim “No Impact” tickets until they match the count of the second largest class, “Outage”.

Upsample of the minority classes: We balance the “Outage” and “No Impact” classes through random downsampling. To address the scarcity of “Threatened” and “Degraded” class tickets, we consider two methods discussed in Section II: text-based augmentation and numeric augmentation. In our experiments, we explore both approaches as described below:

i) Textual augmentation: In the text-based augmentation process, we utilize the NLPAug [15] library to augment the “Threatened” class by doubling the number of tickets in this category. This aligns “Threatened” with the “Outage” class, which is nearly twice its size. Likewise, we augment each “Degraded” ticket four times to balance it with the “Outage” class. Regarding the augmentation process, we employ synonym and antonym replacement, random deletion, spelling correction, split augmentation with “insertion” action, and word embedding-based augmentation. We utilize both contextual and non-contextual word embedding for this purpose. Figure 4 illustrates the updated class distribution of our dataset after text-based augmentation.

ii) Numeric augmentation: In Section II, we discussed the use of numeric augmentation techniques for achieving dataset balance. To convert our text data into a numeric format, we employ a pre-trained BERT model. This model transforms the text input into numeric representations using transformer layers. We then save these numeric representations with their corresponding labels in a CSV file. Finally, we apply the SMOTE technique [16] for numeric augmentation. In our experiments, we increase the number of samples of minority classes to be exactly the same as the majority classes. We train a bidirectional LSTM model by following [8] on this balanced data. The result of this experiment is discussed in Section IV.

B. Domain adaptation

Our work's domain adaptation process, illustrated in Figure 3, involves two key steps: tokenizer fine-tuning and pre-trained BERT transformer layer fine-tuning. These steps are essential for adapting to domain-specific terms in our dataset.

Finetune the tokenizer: Tokenization is the process of converting text into numbers using a vocabulary list. BERT employs "WordPiece" tokenization, breaking words into subwords or characters if they're not in the vocabulary list. For example, "cutover" becomes "cut" and "over" if not in the list. We finetune BERT's tokenizer to include domain-specific words due to our dataset's many such terms.

Finetune the transformer layers: The common approach to text classification involves loading a pre-trained model like BERT, which is readily available online, and fine-tuning it on a specific dataset. This method is effective when the dataset resembles BERT's pre-training data, such as the "BookCorpus" and "English Wikipedia". However, our dataset contains domain-specific words like IPRAN and QAM, which are uncommon in standard English sentences. Consequently, we initially fine-tune the transformer layers on our dataset, following the same procedure as pre-training BERT. We evaluate the fine-tuned BERT model using a metric called "perplexity score", which measures its ability to predict the next word in a sequence. A lower perplexity score indicates better predictive performance. For this fine-tuning, we train for 20 epochs on our dataset. The perplexity score after each epoch is shown in Figure 5. At the beginning of the training, the perplexity score is high. With this fine-tuning, the score gradually converges. We see the benefit of fine-tuning our dataset in Figure 5.

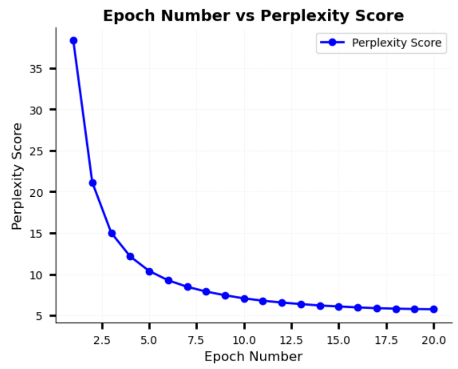


Fig. 5: Perplexity score of the fine-tuned BERT model on our dataset

C. Token prioritization

In this paper, we introduce two feature vector selection strategies for the classifier input. Figure 3 displays the chosen feature vectors (in red boxes) using our methods, which rely on either TF-IDF or probabilistic word relevance scores. Our approach offers three variations based on token prioritization and augmentation methods: TF-IDF with textual augmentation (TF-IDF+TextAug) for one experiment, and probabilistic relevance scores with (Relv. score + TextAug) and without (Relv. score) textual augmentation for the other two experiments.

TF-IDF score: The TF-IDF score quantifies a word's importance in a document or set of documents through term frequency (TF) and inverse document frequency (IDF). TF measures a word's importance within a data sample, while IDF assesses its uniqueness across all samples in the dataset, making rarer words more valuable for conveying meaningful information. We calculate the TF-IDF score for each word in the dataset before the training starts. During the training, we select the top n vectors based on this TF-IDF score. We vary the value of n in our experiments described in Section IV. In the example shown in Figure 3, the value of n is two. Therefore, two vectors (shown in the red boxes) are selected.

Relevance score: We introduce a novel probabilistic relevance score that accounts for a word's importance in classifying a specific dataset class. In contrast, TF-IDF only considers word frequency. Our score is derived using Bayesian confirmation theory as outlined in [7], leveraging the Bayes theorem [18] to combine prior beliefs with evidence likelihood for classifying tickets based on specific words. The posterior probability can be calculated as follows:

$$P(C_i|T_j) = \frac{P(C_i) \times P(T_j|C_i)}{P(T_j)}$$

Here, C_i is the i^{th} class in our dataset, and T_j is the j^{th} token in a sample data. Using the Bayes theorem, we can calculate the posterior and prior probabilities. Now, we can use the Bayesian confirmation theory [7] to either confirm or disconfirm whether a token is relevant for classifying in a particular class or not. The confirmation theory is given below:

$$\begin{cases} T_j \textit{ confirms } C_i, & \textit{iff } P(C_i|T_j) > P(C_i) \\ T_j \textit{ is irrelevant to } C_i, & \textit{iff } P(C_i|T_j) = P(C_i) \\ T_j \textit{ disconfirms } C_i, & \textit{iff } P(C_i|T_j) < P(C_i) \end{cases}$$

According to the confirmation theory, if the difference between posterior and prior is positive, then the token confirms the classification to a specific class. If the difference is 0, then the token is irrelevant for classification. Finally, if the difference is negative, then the token is not relevant for classification in that particular class. Since we have four classes in our dataset, we get four posterior and prior probability scores for each token in a single input. We use the following formula to combine all these four probability scores and get a single relevance score for each token in the input.

$$R_j = \frac{1}{K} \times \sum_{i=0}^{K-1} |P(C_i|T_j) - P(C_i)|$$

Here, R_j is the relevance score for j^{th} token, and K is the total number of classes. Using the above formula, we get a single value for each token in the input. We calculate the prior probability and likelihood before starting the training and save them in a file. During the training, we use this information to calculate the posterior for each class and then calculate the final relevance score. Afterward, based on the relevance score, we select the feature vectors of the top n tokens from the input.

Combining multiple vectors: We select multiple vectors using TF-IDF or relevance score (R_j) and combine them for classifier input. We experiment with two strategies: addition or concatenation. Concatenation increases input dimension, adding complexity. Addition maintains input dimension but may affect input uniqueness in feature space.

IV. Evaluation

We discuss the experiments and results of our proposed approach in this section. First, we discuss our compared approaches, and then we show the results of our proposed approach in comparison with the compared approaches.

Evaluation Setting: Table I shows the compared approaches and how they differ from our proposed approach. We compare our proposed approach with three existing approaches from the literature. The first three rows in Table I show the compared approaches. The last three rows show some variations of our proposed approach.

TABLE I: Compared approaches

Approaches	Pre-processing	Domain adapt.	Token prioritization	Classifier
BERT [7]	Downsample	No	First token	Dense
BERT + WL [9]	Downsample	No	First token	Dense
NumericAug [8]	Downsample + SMOTE	Yes	First token	BiLSTM
TF-IDF + TextAug	Downsample + TextAug	Yes	TF-IDF	Dense
Relv. score	Downsample	Yes	Relv. score	Dense
Relv. score + TextAug	Downsample + TextAug	Yes	Relv. score	Dense

Relv. = Relevance, TextAug = Textual Augmentation, Adapt. = Adaptation, WL = Weighted Loss

The first compared approach is the pre-trained BERT model proposed in [7]. Then, as a common approach to deal with data imbalance, in our second compared approach, we combine BERT with the weighted loss (WL) [9]. In the third compared approach [8], we use numeric augmentation (SMOTE) after extracting features using fine-tuned BERT model. A bidirectional LSTM (BiLSTM) model is used as a classifier in this compared approach.

Since our dataset is highly imbalanced, the F1 score is the most important evaluation metric in our experiments. When dealing with imbalanced datasets, accuracy alone can be misleading and may not provide a comprehensive understanding of the model's performance. In all the approaches, we use the same hyperparameters (epoch, learning rate, optimizer, regularize) to train the BERT model for a fair comparison. For the bidirectional LSTM model, we follow the same hyperparameters mentioned in [8]. Each of the experiments is run five times, and the average score for each evaluation metric is reported in the following subsection. The experiments are conducted on a machine equipped with two Tesla P40 GPUs, each having 24GB of memory, and a main memory of 196GB.

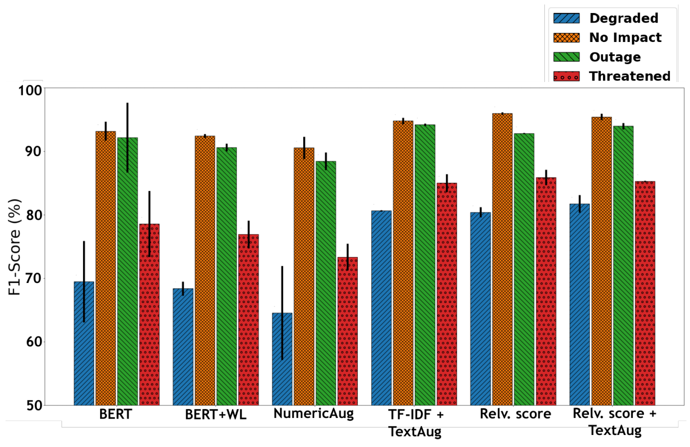


Fig. 6: Class-wise F1 score of all the approaches

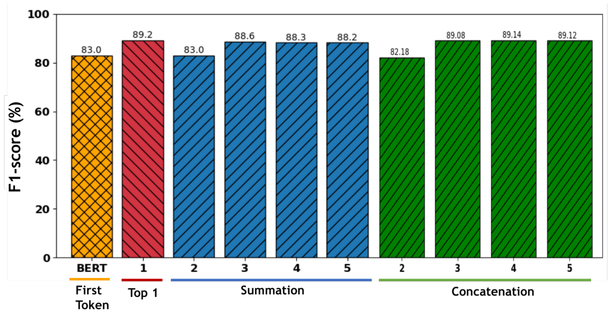
Evaluation Results: The average score of all the evaluation metrics is shown in Table II. Each variation of our proposed approach (the last three rows in Table II) outperforms all the compared approaches in all four evaluation metrics. Since our dataset is imbalanced, it is necessary to look at the class-wise performance of the model. Figure 6 shows the F1 score for each class in our dataset by all the approaches. The F1 score of the compared approaches on minority classes is significantly lower than all variations of our proposed approach, indicating the ability of our proposed approach to handle data imbalance. Furthermore, the larger error bars on the compared approaches indicate an unstable performance of the compared approaches. On the other hand, the F1 scores of each variation of the proposed approach are stable, which indicates a consistently improved performance. If we consider the average F1 score, our best approach is the textual augmentation with TF-IDF score outperforming the baseline approach (BERT) by 7.16%. Another interesting result is the F1 score achieved by Relv. score approach that uses only the relevance score without any form of augmentation (the second last in Figure 6).

TABLE II: Evaluation metrics of all the approaches

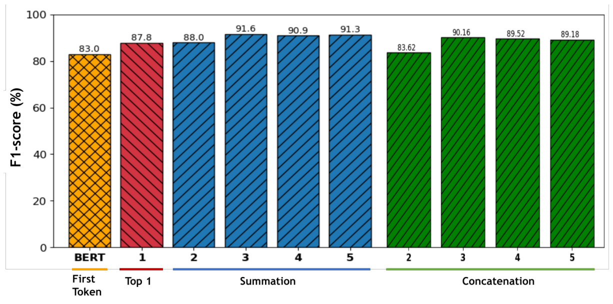
Approaches	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
BERT [7]	89.20	84.40	81.98	83.00
BERT + WL [9]	88.30	83.58	83.18	83.34
NumericAug [8]	85.68	78.02	81.11	79.20
TF-IDF + TextAug	**92.18**	**88.78**	**89.94**	**90.16**
Relv. score	**92.74**	**90.40**	**87.46**	**88.90**
Relv. score + TextAug	**92.68**	**88.84**	**89.52**	**89.14**

As mentioned in Section III-C, we use two strategies (summation and concatenation) to combine the selected feature vectors. So, in the following, we discuss how the performance varies with different values of n. Here, n is the number of selected tokens based on either TF-IDF or relevance score. Figure 7 shows the results of summation and concatenation operations on the selected tokens based on relevance score (Figure 7a) and TF-IDF score (Figure 7b). When $n = 1$, we select the top token based on either TF-IDF or relevance score.

Since there is only one token selected in this case, there is no need for summation or concatenation operations to combine features. The result for $n = 1$ is shown in the "red" bar in Figure 7. The results of the summation operation are shown in the "blue" bar, and the "green" bar shows the results of the concatenation operation. Figure 7 clearly indicates that both



(a) Comparing summation and concatenation of selected tokens with varying "n" values using the relevance score formula



(b) Comparing summation and concatenation of selected tokens with varying "n" values using the TF-IDF score

Fig. 7: Comparison between the two feature combination strategies in our proposed token-prioritization strategies

token prioritization strategies perform better than the compared "first token selection" strategy. We experiment with even higher values of n ($n = 10$, $n = 15$) and observed that the F1 score does not increase much after $n = 5$. In our proposed approach, n is a hyperparameter, and we need to select the best value of n with experiments like other hyperparameters in deep learning. The choice between summation and concatenation depends on the specific characteristics of the data.

V. Conclusion

In this paper, we tackle several challenges for ticket classification that may occur when dealing with datasets from real networks. Our dataset contains NCTs submitted over the past few years during the operation of a major telecommunication operator in Canada. The existing methods from the literature cannot effectively deal with the ticket classification problem, as the distribution of ticket types in real networks is imbalanced. In this paper, we propose two ways to prioritize tokens present in NCTs, along with other parts, that can improve the F1 score even in an imbalanced dataset. Our frequency based prioritzation helps decide which words in the text are more important than others but requires a balanced dataset that can be achieved by textual or numeric augmentation. On the other hand, the probabilistic relevance based prioritzation can do the same even in an imbalanced dataset. We discuss two ways to combine the selected feature vectors. One is adding the selected feature vectors, and the other is concatenating them. We found the concatenation operation works better than the summation on our dataset. We also address the problem of domain-specific words by fine-tuning the tokenizer and transformer layers of the feature extractor. In our experiments, the best model performs better than the standard ones by 7%.

Acknowledgement: This work was supported in part by Rogers Communications Canada Inc. and in part by a Mitacs Accelerate Grant.

References

[1] M. W. Asres et al., "Supporting Telecommunication Alarm Management System With Trouble Ticket Prediction," in *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1459-1469, Feb. 2021.
[2] C. Shorten, T. M. Khoshgoftaar, and B. Furht, "Text data augmentation for deep learning," *Journal of Big Data*, vol. 8, pp. 1–34, 2021.
[3] S. Maldonado, J. López, and C. Vairetti, "An alternative SMOTE oversampling strategy for high-dimensional datasets," *Applied Soft Computing*, vol. 76, pp. 380–389, 2019.
[4] J. P. Gujjar, H. P. Kumar, and M. G. Prasad, "Advanced NLP Framework for Text Processing," *IEEE International Conference on Information Systems and Computer Networks* (ISCON), pp. 1–3, 2023.
[5] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," arXiv preprint arXiv:1901.11196, 2019.
[6] S. Y. Feng et al., "A survey of data augmentation approaches for NLP," arXiv preprint arXiv:2105.03075, 2021.
[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1, pp. 4171–4186, Jun. 2019.
[8] David, Jeniffer, Jiarong Cui, and Fatemeh Rahimi. "Classification of Imbalanced Dataset using BERT Embeddings." (2020).
[9] Shrivastava, I. (2020) Handling class imbalance by introducing sample weighting in the loss function, Medium. Available at: https://bit.ly/3D14AqH (Accessed: 06 July 2023).
[10] M. Hu, "Three-way Bayesian confirmation in classifications," *Cognitive Computation*, vol. 14, no. 6, pp. 2020–2039, 2022.
[11] Y. Shehu and R. Harper, "Enhancements to Language Modeling Techniques for Adaptable Log Message Classification," *IEEE Transactions on Network and Service Management*, 2022.
[12] Y. Shehu and R. Harper, "Improved fault localization using transfer learning and language modeling," *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2020.
[13] J. Tong, Z. Wang, and X. Rui, "A Multimodel-Based Deep Learning Framework for Short Text Multiclass Classification with the Imbalanced and Extremely Small Data Set," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.
[14] Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.
[15] J. Yong, "NLPAug: Data Augmentation Library for NLP," 2021. [Online]. Available: https://github.com/makcedward/nlpaug.
[16] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique". *Journal of Artificial Intelligence Research*, Vol. 16, no. 1, pp. 321-357, Jan 2002.
[17] Mao, L. (2020) Entropy, perplexity and its applications, Lei Mao's Log Book. Available at: https://leimao.github.io/blog/Entropy-Perplexity (Accessed: 04 July 2023).
[18] J. Joyce, "Bayes' Theorem," *The Stanford Encyclopedia of Philosophy*, Stanford University, 2021. [Online]. Available: https://plato.stanford.edu/archives/fall2021/entries/bayes-theorem/.