Predicting Support Ticket Criticality: Using sparse imbalanced data and NLP to predict support ticket urgency

Matt Kane, Elena Petrov and **Nick Sylva**
{mattkane, ebpetrov, nick.sylva}@berkeley.edu
Division of Data Science and Information
University of California
Berkeley, CA 94720-1776, USA

Abstract

Client support ticket data captured in issue tracking systems (ITS) is inherently sparse, repetitive, and includes domain-specific language. However, ITS are both necessary and invaluable business tools to monitor and manage client satisfaction. This paper explores a variety of different attempts to use BERT transfer learning to more effectively use client support ticket data to predict the criticality of a newly created support ticket without requiring expensive pre-training of BERT to capture domain-specific knowledge.

Keywords: customer service, support tickets, issue tracking system, BERT, sparse data

1. Introduction

Bidirectional Encoder Representations from Transformers (BERT) and its direct descendants are currently considered state of the art techniques for natural language processing (NLP). Applying a BERT model via transfer learning to most natural language problems generally yields valuable results as the strength of BERT's generalizability can be empirically demonstrated. However, our attempt to apply the standard fine-tuned BERT model to multi-class client support ticket classification performed poorly. BERT fine-tuning was outperformed by multiple other approaches using frozen BERT weights with additional machine learning techniques.

We hypothesized that some domain-specific data pre-processing and enrichment before feeding it to BERT would yield improved model performance without requiring expensive, domain-specific pre-training as in Rongali et al. (2020). We developed a process that we call "semantic augmentation" to inject domain-specific information into the model. Semantic augmentation consisted of replacing acronyms, domain-specific language, and short-hand with phrases that better mimic natural language. However, classifying support ticket criticality using BERT fine-tuning alone, despite semantic augmentation, proved to be a significant challenge because support ticket data are inherently sparse and highly imbalanced. We achieved better results by freezing BERT's weights and transferring its general language knowledge as embeddings for use in simple dense and convolutional neural networks.

1.1 Support Ticket Data

Many businesses today use some form of a customer support ticket tracking system. These systems are commonly referred to as Issue Tracking Systems. Their primary purpose is to collect information on interactions with clients, but have become a critical part of enterprise infrastructure, yielding large, valuable datasets that businesses can use to improve operations and relationships with clients. These systems often include dashboards and statistics to measure the overall health of the business including responsiveness, product quality, and customer satisfaction.

We worked with data sourced from a middle market software company that serves approximately 6,000 clients in the United States and Canada. Their particular ticket system was built in-house, but follows a standard support ticketing data structure and workflow whereby customer support data is collected through two channels, email and phone calls. Client emails are automatically captured in the ticketing system, while phone calls are summarized as notes by customer service representatives who receive the calls. A support ticket tracks a single client issue (though some track multiple, seemingly-related issues), and is comprised of a number of ticket actions that include a criticality rating for the ticket of "Low", "Normal", "High", or "Critical." Ticket actions primarily reflect interactions with the customer, but include escalations and responses within the company needed to move towards issue resolution. The dataset is comprised of approximately 425,000 tickets with 2.7 million ticket actions over a 10 year period from 2010 to 2019.

1.2 Related Works

We used BERT by Devlin et al. (2018a) as our base language model for transfer learning for our task. We propose a much simpler model to improve classification of information in a ticketing systems than Customer Obsession Ticket Assistant (COTA) by Molino et al. (2018b) in which ranking and deep networks were used to better classify tickets.

2. Pre-Processing Data

One goal of our paper is to help BERT perform better at processing natural language in support ticket systems by applying pre-processing and data enrichment techniques to the original dataset. These techniques attempt to address two difficult attributes of support ticket systems:

1. Acronyms, short-hand, and domain-specific language that have little to no context in BERT's general language model, and

2. Significant class imbalances inherent to this multi-class dataset.

Before addressing these attributes, we will describe a necessary data cleansing task because of how the support ticket data is collected.

2.1 Cleaning Repetitive Data and Regular Normalization

Our first pre-processing task consisted of cleansing and normalizing the data. Support ticket systems often ingest client emails automatically into the ticket actions. In cases where this

process is not automated, redundant information is often captured because of copy and paste actions by support representatives. This creates a large amount of unnecessary text that may inhibit BERT's ability to understand the ticket. We eliminated email header information and removed repetitive emails chains. As most email software retains prior emails as appended text to a new email, ticketing software data also includes this repetitive text. We postulated that performing these steps would increase the signal-to-noise ratio of our data and eliminate features unnecessary for our task.

We implemented a simple, one-pass algorithm over an ordered set of ticket actions (first by ticket and then by ticket action) that cached for a single future read, the text of the prior ticket action and the ticket identifier. On the next read, if the new ticket action belonged to the same ticket identifier and the second half of the prior ticket action matched the end of the current ticket action, we removed the end of the new ticket action up until the point where they no longer matched while traversing the text backwards. We then stored the original ticket action text in the cache and read forward to the next ticket action. This eliminated repetitive emails chains, and still preserved any additional notes that may have been added above the email by a support representative.

2.2 Semantic Augmentation for Acronyms and Domain-Specific Words

Our second pre-preprocessing task entailed data enrichment. We replaced acronyms, short-hand, and domain-specific words with natural language phrases. Before running the BERT tokenizer which tokenizes words into the 30,000 word BERT WordPiece vocabulary, we used a simple, open source English tokenizer available from OpenNLP and Alex Point on GitHub, Point (2019). We counted the tokens in the dataset and ordered them in descending order by count. Of the over 1.75 million unique tokens in the dataset, 17,063 unique tokens accounted for 90% of the total non-unique tokens in the dataset (Table 1).

Table 1: Unique Words by Percentile of Total Words

Percentile of Total Words	Unique Words	Avg. Count per Word
10%	5	1,298,602
20%	16	800,591
30%	40	367,277
40%	91	156,440
50%	206	67,638
60%	467	30,107
70%	1,052	13,124
80%	2,789	3,711
90%	12,397	433
100%	1,736,034	1

Reviewing this set of 17,063 tokens revealed that a small, but critical fraction were acronyms, short-hand or other domain-specific words that were unlikely to be recognized by the BERT tokenizer. Next, we ran these 17,063 tokens through the BERT tokenizer to understand how these words would be separated into WordPieces. Using these data, we replaced tokens lacking a meaningful tokenization with more understandable natural language. We manually mapped 544 (3%) of these BERT unrecognizable tokens to natural

language phrases. Acronyms and short-hand were easy replacements, but other domain-specific words like product names were replaced with natural language describing their behavior. The process of reviewing all tokens, creating natural language phrases for 544 of them, and conducting needed replacements took fewer then 3 hours to complete by a single domain expert. Pre-training BERT for domain-specific knowledge on a dataset of this size would have required many more hours and financial resources. Examples of natural language replacements are included in Table 2.

Table 2: Examples of Word Replacements with Natural Language Phrases

Word to Replace	Phrase Replacement
EE	employee
TIN	Taxpayer Identification Number
SSN	Social Security Number
W2	Employee Annual Withholding Form
1099	Annual Vendor Tax Form
LVM	left voicemail
HR	human resources

2.3 Downsampling and SMOTE for Solving the Class Imbalance

Class imbalance is a common property of real-world datasets. This support ticket dataset is no different, with the majority class (criticality of "Normal") making up 78% of all labels. The real world implications for support ticket data are exacerbated by the fact that our smallest minority class ("Critical"), or as described by Chawala et al. (2002) as the "abnormal" class, only comprises of 0.25% of our dataset (Table 3). In order to address the data imbalance, we applied two techniques to help the models better detect minority classes during training: (1) downsampling, a simple random sampling technique by which we downsampled the two largest majority classes, and (2) Synthetic Minority Over-sampling Technique, a.k.a. SMOTE, Chawala et al. (2002), a sophisticated oversampling technique to boost the minority classes and generate a balanced dataset.

Table 3: Class imbalance in baseline datasets

Class	Baseline 1: Regular Normalization	Baseline 2: With Semantic Augmentation
Critical	0.254%	0.257%
High	17.051%	16.789%
Normal	78.160%	78.473%
Low	4.535%	4.480%

3. Modeling

3.1 Datasets

Our semantic augmentation and data cleaning techniques yielded two different baseline datasets:

1. **Baseline Dataset 1:** Normalized and cleaned with no semantic augmentation.

2. **Baseline Dataset 2:** Baseline 1 dataset with semantic augmentation applied.

We applied class imbalance resolution techniques to each of these baselines to create two additional versions of each for a total of six datasets. One version downsampled the two majority classes, "Normal" and "High," to the frequency of the third-most common class, "Low." The second version used SMOTE to oversample synthetic text labelled with the minority class. Our balancing techniques were applied only to the training set for each dataset in order to preserve real-world class frequencies for model validation.

3.2 Model Specifications

All three of our model specifications utilized the uncased version of the BERT base model, which consists of 12 layers, 768 hidden nodes, 12 attention heads, and 110 million parameters. In order to be fed to the model, the text of each ticket action was truncated to the first 126 WordPiece tokens with the BERT-specific [CLS] and [SEP] tokens defining the start and end of each sequence. We chose a length of 128 to capture as much information as possible while also considering computational efficiency.

3.2.1 Model 1: Simple Classifier with BERT Fine Tuning

Model 1 is the simplest model, consisting of BERT with a single four-node dense layer with softmax activation serving as the prediction output. We used the BERT pooled sequence representation associated with the [CLS] token as the output to the final dense layer. Fine-tuning of BERT weights was enabled for this model during training. The intent behind this model specification was to examine the efficacy of BERT fine-tuning as applied to our unique use-case.

3.2.2 Model 2: Frozen BERT with Dense Neural Network

Model 2 extends Model 1, but has a two key differences. First, BERT weights were frozen, disabling fine-tuning. With this approach, the 768-dimension pooled sequence representation simply acted as an embedding. Second, instead of immediately classifying each ticket action after passing through BERT, we added four fully-connected layers prior to the final softmax output layer. The goal for Model 2 is to examine if the dense network would be able to better detect the differentiating features in our text over BERT with fine-tuning.

3.2.3 Model 3: Frozen BERT, Convolutions, and Dense Neural Network

Model 3 enhances Model 2 by including a convolutional neural network. Because BERT's pooled sequence representation only includes a single value for each token, we instead used the full output, which consists of a 768-dimensional embedding for each token in the sequence. The BERT output embeddings were convolved with 196 filters, 64 each of three different sizes. Each convolution was max-pooled with the pooled output concatenated into a single output vector. Finally, the output vector was passed through a four fully-connected layers to a softmax output for prediction.

4. Results

We chose to measure success of each model using the area under the Receiver Operator Characteristic Curve (AUROC). We also represent the confusion matrix graphically to help understand the success of each model. Table 4 summarizes the top three results. Based on the micro-average AUROC, our top two results were generated using Model 3 using datasets without applying any balancing techniques. Surprisingly, Dataset 2 did not outperform Dataset 1.

We initially expected Model 1 to serve as the baseline model upon which we would improve upon; however, the model performed poorly (refer to figures in Appendix B) and was unable to predict more than a single class for all 6 dataset/semantic augmentation/balancing combinations.

Table 4: Top 3 Summary Results

Model	Dataset	Balancing	Accuracy	Micro Avg AUROC	Macro Avg AUROC
Model 3	Dataset 1	None	84.89%	86.54%	96.74%
Model 3	Dataset 2	None	78.53%	78.26%	95.48%
Model 2	Dataset 1	None	81.11%	78.75%	94.91%

5. Conclusion

Since its introduction in 2018, BERT has proved itself to be one of the most significant innovations in natural language processing. While many papers have documented BERT's success in fine tuning tasks across various domains, our findings revealed that fine tuning support ticket data on the BERT base uncased trained model yielded unsatisfactory results. We believe this is primarily due to the unique attributes of support ticket datasets, namely the fact that (i) the ticket text is riddled with abbreviations, short-hand, domain-specific language and fragmented language rather than proper full sentence prose, and (ii) the dataset is highly imbalanced by class.

These two attributes are inherently misaligned with the underlying BERT base models which are pre-trained on a large text corpus of proper natural language prose. Our results suggest that bespoke business datasets like client support tickets require an investment in pre-training to fully benefit from BERT capabilities. Nevertheless, we were able to generate good results by using models comprised of layered neural networks over frozen BERT weights without pre-training.

In a separate project, we would like to explore other potential enhancements including (i) hyperparameter tuning, and (ii) model architectures using unfrozen BERT weights followed by neural network layers.

References

Chawla et al. Smote: Synthetic minority over-sampling technique. *Journal of artificial intelligence research 16*, 2002. URL https://arxiv.org/pdf/1106.1813.pdf.

Devlin et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018a. URL https://arxiv.org/pdf/1810.04805.pdf.

Molino et al. Cota: Improving the speed and accuracy of customer support through ranking and deep networks. *arXiv:1807.01337*, 2018b. URL https://arxiv.org/pdf/1807.01337.pdf.

Rongali et al. Improved pretraining for domain-specific contextual embedding models. *arXiv:2004.02288*, 2020. URL https://arxiv.org/pdf/2004.02288.pdf.

Point. Open source nlp tools, 2019. URL https://github.com/AlexPoint/OpenNlp.

Appendix A. Results Table

Table 5: Full Summary Results

Model	Dataset	Balancing	Accuracy	Micro Avg AUROC	Macro Avg AUROC
Model 3	Dataset 1	None	84.89%	86.54%	96.74%
Model 3	Dataset 2	None	78.53%	78.26%	95.48%
Model 2	Dataset 1	None	81.11%	78.75%	94.91%
Model 2	Dataset 2	None	79.51%	69.28%	93.91%
Model 3	Dataset 1	SMOTE	65.76%	82.64%	93.05%
Model 1	Dataset 2	SMOTE	78.55%	50.00%	91.19%
Model 1	Dataset 2	None	78.53%	50.00%	91.17%
Model 1	Dataset 1	None	78.16%	50.00%	91.05%
Model 1	Dataset 1	Downsampled	78.09%	50.00%	91.01%
Model 2	Dataset 1	SMOTE	61.20%	75.60%	89.86%
Model 2	Dataset 2	SMOTE	58.05%	65.63%	88.14%
Model 3	Dataset 2	SMOTE	60.71%	73.76%	87.09%
Model 2	Dataset 1	Downsampled	52.69%	79.82%	86.22%
Model 3	Dataset 2	Downsampled	52.06%	76.25%	85.83%
Model 2	Dataset 2	Downsampled	47.53%	68.38%	82.58%
Model 1	Dataset 2	Downsampled	16.72%	50.80%	71.15%
Model 3	Dataset 1	Downsampled	55.90%	86.10%	59.07%

Appendix B. Model 1 Results

B.1 Model 1 with Regular Normalization

Figure 1: Model 1 Regular Normalization Confusion Matrices

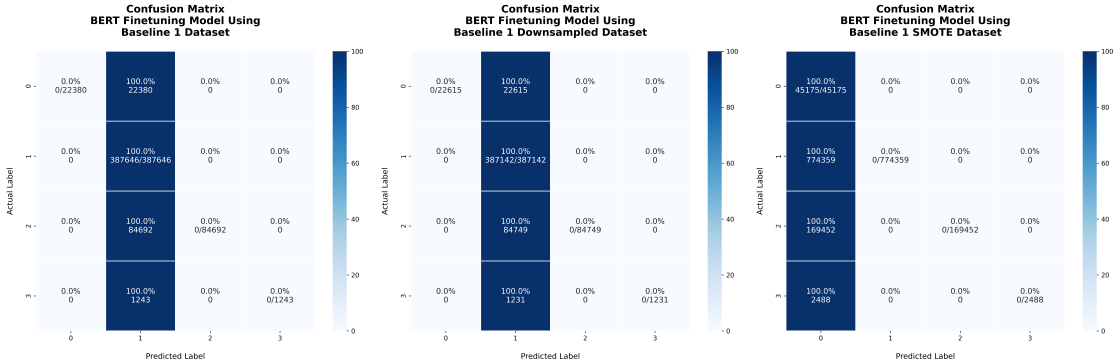
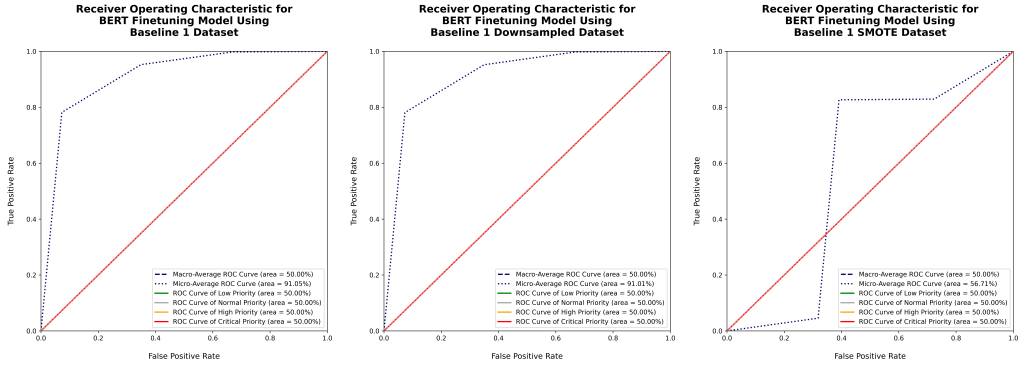


Figure 2: Model 1 Regular Normalization ROC Graphs



B.2 Model 1 with Semantic Augmentation

Figure 3: Model 1 with Semantic Augmentation Confusion Matrices

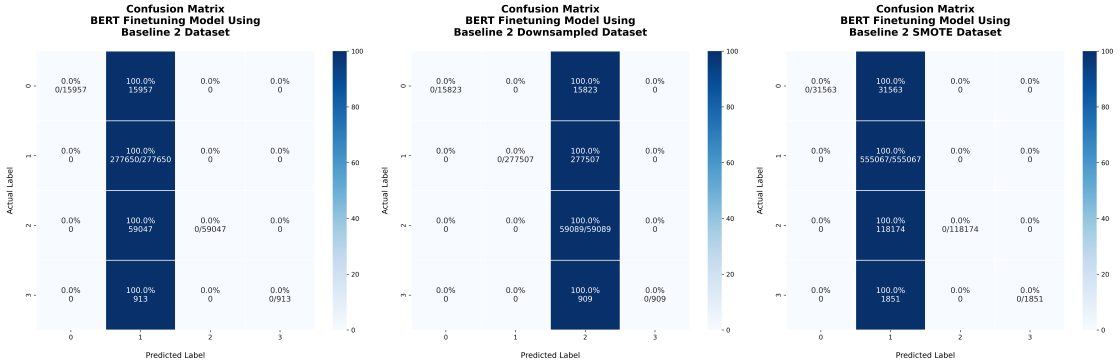
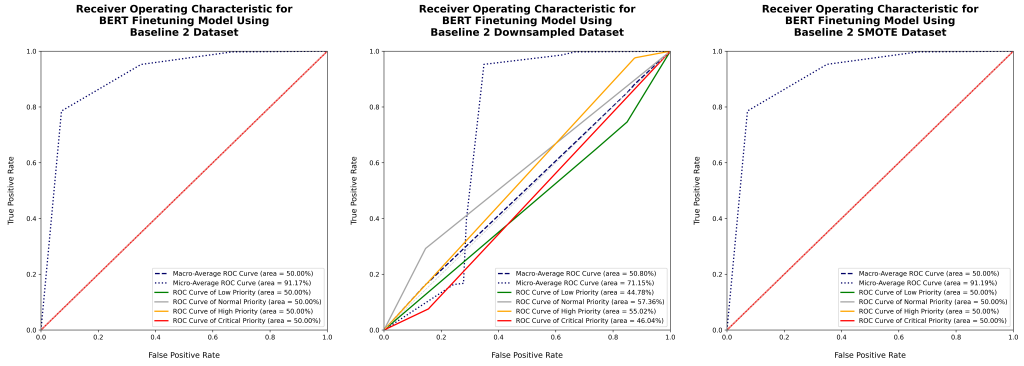


Figure 4: Model 1 with Semantic Augmentation ROC Graphs



Appendix C. Model 2 Results

C.1 Model 2 Regular Normalization

Figure 5: Model 2 Regular Normalization Confusion Matrices

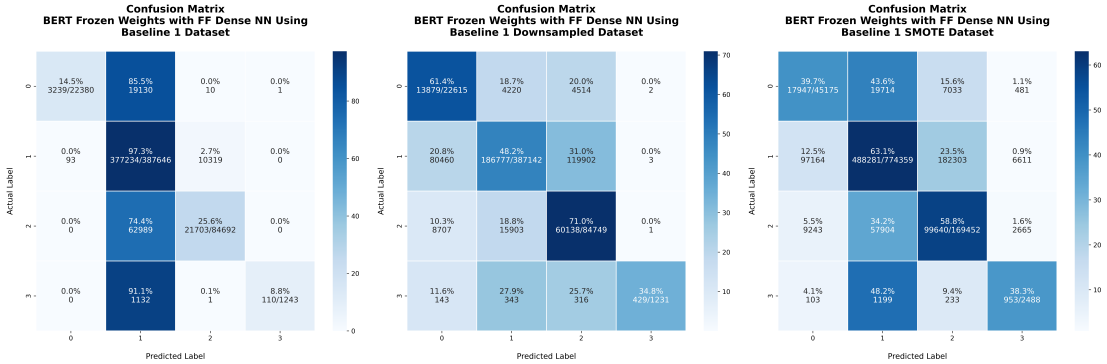
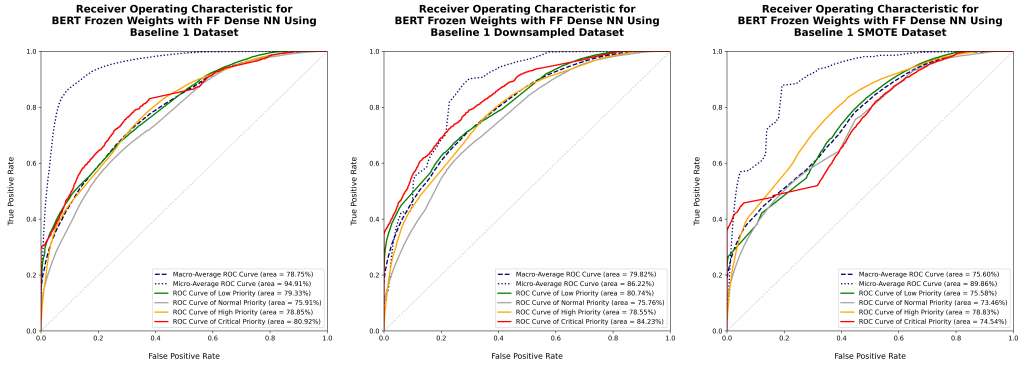


Figure 6: Model 2 Regular Normalization ROC Graphs



C.2 Model 2 with Semantic Augmentation

Figure 7: Model 2 with Semantic Augmentation Confusion Matrices

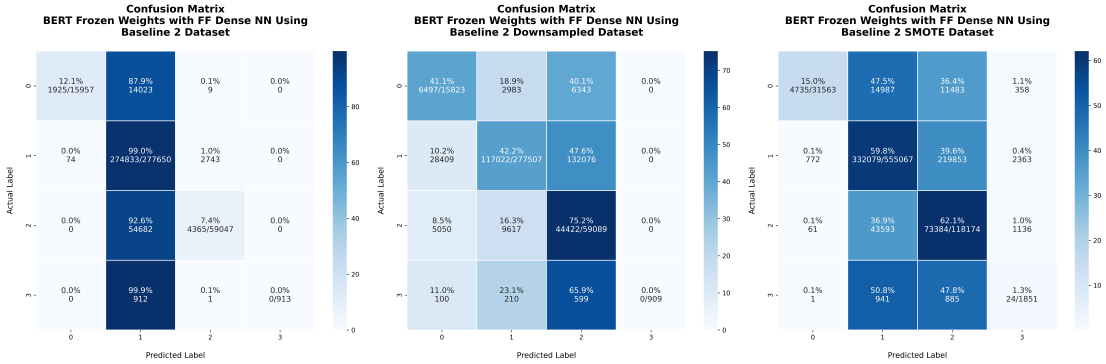
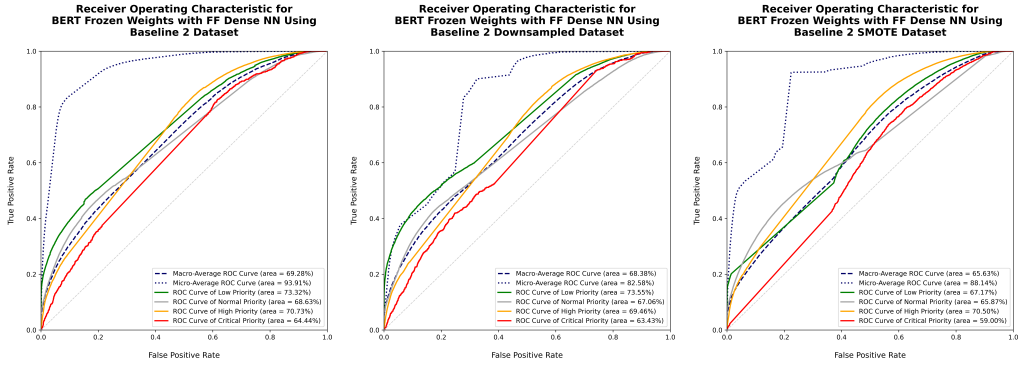


Figure 8: Model 2 with Semantic Augmentation ROC Graphs



Appendix D. Model 3 Results

D.1 Model 3 Regular Normalization

Figure 9: Model 3 Regular Normalization Confusion Matrices

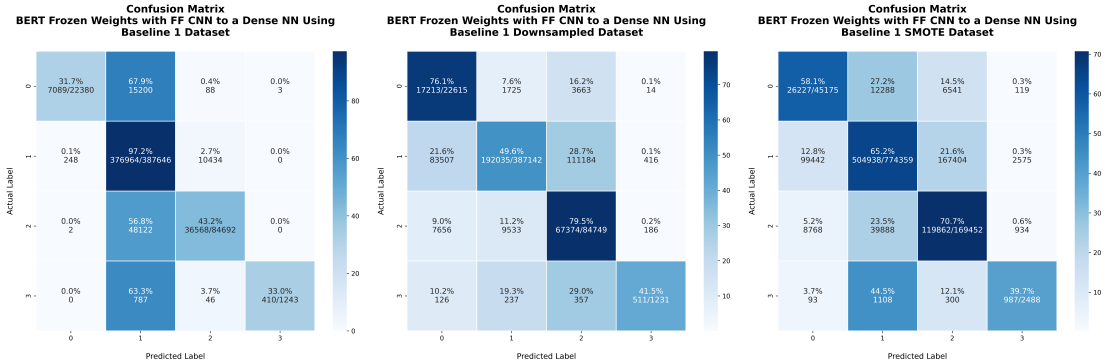
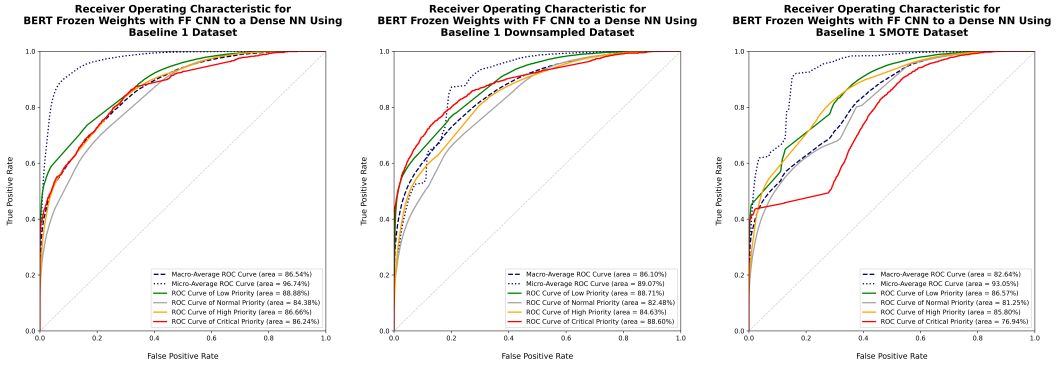


Figure 10: Model 3 Regular Normalization ROC Graphs



D.2 Model 3 with Semantic Augmentation

Figure 11: Model 3 with Semantic Augmentation Confusion Matrices

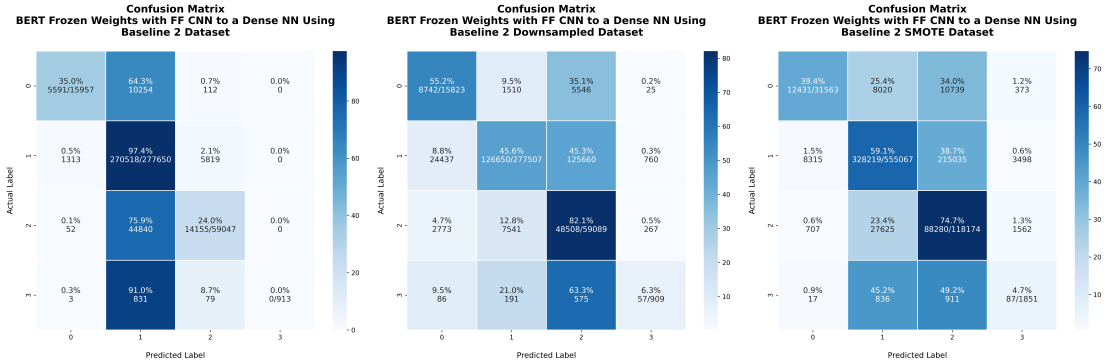


Figure 12: Model 3 with Semantic Augmentation ROC Graphs