Impact Genome Classification

DSC 672: Data Science Capstone

**ABSTRACT**

The Impact Genome Project maintains a registry with the verified outcomes of social programs. These programs are currently being categorized manually. To improve efficiency, they want to use a classification model to predict the outcomes of each program.

As this project involves Natural Language Processing (NLP), our group decided to develop a BERT model, which has become the industry standard for NLP tasks. Impact Genome provided us with a dataset containing values for Program Report ID, Program Description, Impact Area, Genome, Outcome, and Outcome ID.

Before developing the model, our team needed to perform preprocessing and address class imbalance in the dataset. To address the class imbalance, we supplemented the dataset with synthetic data that contained more observations for the minority classes.

We developed three hierarchical BERT classification models to predict outcomes based on program descriptions. The most successful model had a 68% accuracy when predicting outcome ID.

KEYWORDS: Natural Language Processing (NLP), BERT model, Text classification,

Hierarchical classification, Imbalanced dataset

**Author Information**

Julia Aptekar, DePaul University, School of Computing, [japtekar@depaul.edu](mailto:japtekar@depaul.edu)

John Leniart, DePaul University, School of Computing, [jleniart@depaul.edu](mailto:jleniart@depaul.edu)

Arham Mehdi, DePaul University, School of Computing, [kmehdi@depaul.edu](mailto:kmehdi@depaul.edu)

Natalie Olechno, DePaul University, School of Computing, [nolechno@depaul.edu](mailto:nolechno@depaul.edu)

**INTRODUCTION**

The Impact Genome Registry, part of the publicly funded Impact Genome Project, is a trusted, centralized database that allows non-profits to report their impact for potential funders to see. By standardizing and verifying the outcomes of social programs, the registry serves as a third-party source relied upon by nonprofits, governments, private foundations, and corporations worldwide. It opens possibilities for grantee discovery, benchmark programs, improved ROI (return on investment), and increased contributions to social programs. The standardized impact taxonomy and centralized database for reporting impact can enhance the evaluation of grants and increase the value of grants and donations.

Impact Genome’s taxonomy provides a consistent method for classifying social programs, helping potential funders identify areas to support. It also enables comparisons of related programs’ performance. Comparing non-profits that claim different outcomes can be challenging. As in the case of three non-profits focused on homelessness, one might describe their outcome as “reducing homelessness”, another as “supporting the homeless”, and a third as “housing the homeless”. Our goal is to develop a classifier that takes a program description as input and assigns it to the appropriate outcome category.

The data was provided to Professor Adam Hecktman. He is the founder and Chief Data Architect of For Good Advisory, where he collaborates with non-profits and funders to leverage data and data analytics. Impact Genome contracted For Good Advisory to develop a classification model that predicts the outcome given a program description.

Impact Genome currently uses a classifier that was built using Google’s Vertex AI, a fully managed platform that requires no programming. However, that model’s accuracy is only 53%. Our team is seeking to develop a new classification model that can outperform that accuracy and thus serve as a more effective tool for categorizing new social programs.

**LITERATURE REVIEW**

**Transformers**

The Transformer model architecture enhances parallelization, leading to faster training and improved performance across multiple tasks. Unlike sequential RNNs or LSTMs, the Transformer uses self-attention to process input sequences in parallel, efficiently capturing long-range dependencies important for multi-task learning scenarios (Vaswani et al., 2017). BERT leverages this architecture through pre-training on large corpora and fine-tuning for task-specific applications (Koroteev, 2021). The self-attention approach allows the model to dynamically weight different input sequence elements, adapting its contextual focus to the unique demands of various tasks and consequently enhancing both generalization and predictive performance.

Open-source libraries like HuggingFace Transformers have streamlined the implementation of Transformer-based models, providing tools to train and fine-tune pre-trained models like BERT, with enhanced interoperability between deep learning frameworks (Wolf et al., 2020). Transformer architectures, including BERT, have become dominant in natural language processing, consistently outperforming convolutional and recurrent neural networks across various language understanding tasks (Minaee et al., 2021; Gasparetto et al., 2022). These developments highlight the utility of Transformer models for multi-task classification, demonstrating their ability to capture and utilize nuanced contextual representations.

**BERT & Other Models**

Introduced by Google in 2018, BERT revolutionized NLP by reducing the need for task-specific architectures. It outperformed state-of-the-art models on the GLUE benchmark, becoming a near-default approach for NLP challenges (Garrido-Merchan et al., 2023). By conditioning on both left and right contexts across all layers, BERT learns deep bidirectional representations from unlabeled text, making it ideal for complex classification datasets (Devlin et al., 2019).

Pre-trained sentence encoders have advanced NLP, though concerns persist about their interpretability. Research suggests these models, especially in deeper layers, can represent complex syntactic and semantic structures beyond mere statistical co-occurrences (Tenney et al., 2019). The proposed BERT models leverage this capability, using deep bidirectional encoding to handle multiple NLP tasks effectively. By capturing both low- and high-level language features, these models show strong potential for multi-task classification, with targeted fine-tuning enhancing performance on limited training data (Chi et al., 2019).

**Addressing Class Imbalance**

Data imbalance often leads to poor classification performance, particularly with insufficient minority class samples (Ceccon, 2023). In real-world scenarios, obtaining balanced data can be challenging and generating text to supplement minority classes may result in contextual and semantic information loss (Padurariu & Breaban, 2019). While BERT adapts to limited training data, its performance degrades with significant training-test data divergence (Tayyar Madabushi et al., 2019; Ghosh et al., 2022).

To mitigate these challenges, the proposed models employ a comprehensive approach. Synthetic data generation using large language models like GPT-4 and T4 creates contextually rich examples for underrepresented classes, preserving semantic nuances in NLP tasks (Preeti, 2024; Ganganwar & Rajalakshmi, 2023). Balanced datasets consistently improve classification accuracy, making data augmentation critical (Shaikh et al., 2021).

The models integrate advanced training techniques to combat class imbalance. Focal loss prevents common classes from dominating training by emphasizing misclassified examples (Lin et al., 2020; Ceccon, 2023). Uncertainty weighting manages task balance (Kendall et al., 2018), while curriculum learning enables dynamic task prioritization, allowing the model to progressively tackle tasks from easier to more challenging (Guo et al., 2018).

**Future Work with BERT**

While not utilized during our project, it could be beneficial to experiment with other variations of BERT such as Sentence-BERT (SBERT), monoBERT, and duoBERT. SBERT uses siamese and triplet network structures to produce semantically meaningful sentence embeddings that can be compared to each other using cosine similarity (Reimers et al., 2019). Both monoBERT and duoBERT are used for multi-stage document ranking. The latter uses pairwise classification and can offer improved ranking accuracy (Nogueira et al., 2019).

**DATA**

Impact Genome has provided two datasets, both stored in Microsoft Excel files. The first dataset has six columns and 6,649 rows (Figure 1). The names of the columns are: Program Report ID, Program Description, Impact Area, Genome, Outcome, and Outcome ID. All six columns contain categorical data. Program ID and Outcome ID have numerical values, and the other four columns contain string values.

The second dataset contains only two columns: Program Description and Outcome ID. These two files contain the same exact observations (i.e. the same programs and outcomes). The second file just has four fewer columns.

Outcome ID is the target variable that our classification model will be predicting. In the dataset, there are 289 unique outcome IDs. However, the dataset is very imbalanced. Some outcomes only appear once or twice. Some appear hundreds of times. The most common outcome appears 361 times in the dataset. There are also a handful of outcomes that appear more than 200 times in the dataset.

There are 12 unique values for impact area and 50 unique values for genome in the dataset. Like outcome ID, both of these variables have clear majority and minority classes. The most common impact area appears over 1400 times in the dataset, and the least common impact area appears fewer than 20 times. The most common genomes appear over 600 times in the dataset, and the rarest genomes appear fewer than five times.

It is important for our group to have an understanding of the majority and minority classes present in the dataset. If we do not address the class imbalance, the model we develop will likely overfit to the majority classes and underfit to the minority classes.

The vast majority of outcome IDs only appear once or twice. Looking at a histogram of the outcome ID counts (Figure 2), you can see it is clearly skewed. This skew is also noticeable when looking at the descriptive statistics for the outcome ID counts (Figure 3). The average count for an outcome ID in this dataset is 22.29. That mean value falls outside of the interquartile range (IQR), which indicates the data is skewed. The IQR for this data ranges from 2 (Q1) to 19 (Q3).

This is the data that Impact Genome started with. Professor Hecktman noted that the dataset contains missing values and special characters that will need to be cleaned. He also noted that while the first file has a few extra columns, that additional data might not add value to our classification model.

In addition to the data provided by Impact Genome, Professor Hecktman’s team has also created some synthetic data using a Python script that leverages ChatGPT. This synthetic dataset was created to supplement the number of observations for the minority class outcomes. This allows us to train our model with more data for the outcomes that were underrepresented in the original dataset. The synthetic dataset has 4,409 rows and five columns. The only column from the original dataset that does not appear in the synthetic dataset is Program Report ID.

All 12 impact areas and all 50 genomes are represented in the synthetic dataset. The main difference between the synthetic dataset and the original dataset is the frequency at which the outcome IDs appear. The outcome IDs that appear more than 20 times in the synthetic dataset are all part of the minority classes in the original dataset. Most of those outcome IDs appear only once or twice in the original dataset.

As we alluded to earlier, we needed to perform some data cleaning on the program description values because many of them contained special characters that could hinder our Natural Language Processing (NLP) model. While some NLP models have the ability to interpret and contextualize certain punctuation marks, they would likely be confused by the special characters present in some of the program descriptions in our dataset (Figure 4).

To clean this text data, we utilized regular expressions. Using the sub method from the regular expression (re) library in Python, we were able to specifically target and remove all special characters that we did not want in the program descriptions.

**METHODOLOGY**

**Exploratory Data Analysis**

As was previously mentioned in this report, within the dataset there are 12 unique values for impact area, 50 unique values for genome, and 289 unique values for outcome ID. As we proceeded with our exploratory data analysis (EDA), we wanted to understand the prevalence of different combinations of these values and clearly define the majority and minority classes at each level: impact area, genome, and outcome.

For the next step in our EDA, we generated a heat map that shows the counts for all combinations of impact area and genome (Figure 5). This chart shows the impact areas Critical Human Needs, Economic Development, Education, and Public Health are part of some of the most frequent combinations.

Then, we examined the genomes. We created bar charts that show the total number of programs under each genome. We divided the 50 genomes into two groups: the 25 genomes with the most number of programs (Figure 6) and the 25 genomes with the fewest (Figure 7). These charts give us a clear picture of the majority and minority classes at the genome level.

Lastly, we performed some exploratory analysis on the text values in the program description column. We generated a word cloud (Figure 8) to attempt to identify any common words and themes when looking at all of the program descriptions contained in the dataset. We also looked at word clouds generated from the program descriptions for each impact area. This provided us with some additional context for how these programs are categorized.

**Research Questions**

When we began this project, we defined four research questions. Throughout the course of this project, we have been able to answer those questions.

BERT’s contextual embeddings provide a deep understanding of language semantics, significantly outperforming frequency-based methods like TF-IDF, which rely on isolated word occurrences. This allows BERT to better capture nuanced descriptions of social programs, leading to improved classification accuracy. By applying synthetic data augmentation and class-weighted loss functions, we improved the recall and macro F1-score of underrepresented classes. Without these techniques, the model heavily favored majority classes, making it ineffective for minority-class predictions.

The pre-trained BERT embeddings allowed our model to leverage sentence-level semantics, leading to improved performance across outcome categories. The macro F1-score improved significantly compared to baseline models, particularly for categories with limited training examples. Some key preprocessing steps included lowercasing and removing special characters to standardize text input, tokenization using BERT’s WordPiece tokenizer to maintain linguistic integrity, and synthetic data augmentation to address imbalanced class distributions.

**BERT Model 1: Method**

The model processing pipeline included text preprocessing to normalize program descriptions. This involved removing special characters, lowercasing all text for consistency, and tokenizing input sequences using the BERT tokenizer to ensure they fit within BERT’s maximum sequence length. Input sequences were then padded to maintain a uniform length. The dataset was split using stratified sampling to maintain class distributions across training and validation sets, ensuring rare outcomes appeared in both sets to prevent underrepresentation. Additionally, synthetic data was introduced to supplement underrepresented classes, improving minority class recall and overall model generalization.

The model architecture consists of a pre-trained BERT base model as the foundational encoder, followed by three independent classification heads for Impact Area, Genome, and Outcome ID prediction. The Impact Area classifier processes BERT’s pooled output through a fully connected layer with ReLU activation and dropout regularization, followed by a classification layer to predict one of 12 Impact Areas. The Genome classifier utilizes both BERT embeddings and predicted Impact Area probabilities as input, allowing hierarchical dependencies to influence genome classification. The final Outcome classifier follows a similar structure, using the program description, predicted Impact Area, and predicted Genome to classify into 203 Outcome IDs.

The model was trained using the AdamW optimizer with a learning rate of 2e-5, and training was structured across hierarchical levels, with four epochs for Impact Area, five epochs for Genome, and eight epochs for Outcome ID classification. A ReduceLROnPlateau scheduler was implemented to reduce the learning rate when validation performance plateaued. Learning rates were adjusted based on model components, with lower learning rates applied to the BERT encoder to preserve pre-trained representations, while higher rates were assigned to task-specific layers. Early stopping was introduced to halt training if the Outcome ID F1-score did not improve for three consecutive epochs.

To improve classification on imbalanced data, focal loss replaced standard cross-entropy loss, with gamma parameters adjusted to focus training on hard-to-classify examples. Additionally, uncertainty weighting was used to dynamically balance loss contributions from different classification tasks, preventing any single task from dominating the training process.

To further enhance performance, curriculum learning and hard negative mining were incorporated. Curriculum learning gradually increased the weight of the Outcome ID classification task over the first five epochs, ensuring the model built a strong foundation in Impact Area and Genome classification before focusing on the more complex Outcome ID classification. Hard negative mining was introduced after three epochs, prioritizing high-loss samples for training, allowing the model to refine its understanding of difficult cases. To manage computational demands, selective sampling and memory optimizations were applied, ensuring efficient training without excessive resource consumption.

**BERT Model 2: Method**

Data Pre-processing

The model processing pipeline includes text preprocessing to normalize program descriptions. Input sequences are tokenized using the BERT tokenizer and padded to a uniform length to remain within BERT's maximum context length (McCormick & Ryan, 2019). The dataset is split using stratified sampling which maintains the highly imbalanced class distribution across training and validation sets. The train-test split preserves the relative frequencies of the Outcome categories, ensuring rare outcomes appear in both training and validation sets in proportion to their overall frequency (Figure 12). This prevents cases where rare outcome classes would be entirely absent from either the training or validation sets, impacting model performance evaluation.

Model Architecture

The architecture (Figure 9) consists of a pre-trained BERT base model (Devlin et al., 2019) as the foundational encoder, followed by three specialized classification heads arranged in a hierarchical structure. The BERT encoder processes program descriptions to generate contextual embeddings, which feed into the hierarchical classification framework. The model’s structure leverages the natural categorization structure found in the program descriptions, where broader Impact Areas contain multiple Genomes, which in turn contain multiple specific Outcomes.

At the first level, the Impact Area classifier processes BERT’s pooled output through a fully-connected layer with ReLU activation and dropout regularization, followed by a classification layer. The model extracts feature representations from the Impact Area probability distribution through a dedicated feature extractor to inform subsequent classification tasks.

The second-level Genome classifier incorporates both the BERT embeddings and the Impact Area probabilities as input, implementing a form of hierarchical conditioning where higher-level predictions influence lower-level ones. The concatenated features pass through similar fully-connected layers before classification into possible Genome categories.

For the final Outcome prediction, the model uses a multihead cross-attention layer (with 4 attention heads) between the Impact Area and Genome feature representations (Vaswani et al., 2017) . The cross-attention layer allows the model to focus on relevant interactions between Impact Areas and Genomes that are particularly predictive of specific Outcomes. The architecture concatenates BERT embeddings, Impact Area probabilities, Genome probabilities, and the attended features to make the final prediction through a dedicated classification head into 289 possible Outcome categories.

Training Strategy

Several advanced training techniques are applied when training the model to help improve Outcome prediction performance.

Optimization and Scheduling

Optimization is performed using the AdamW optimizer with a learning rate of 3e-5 and a ReduceLROnPlateau scheduler that halves the learning rate when validation performance plateaus. The scheduler monitors the macro F1 score of the Outcome predictions and reduces the learning rate after two epochs without improvement. The optimization strategy uses parameter groups with different learning rates for the model components mirroring the model’s architecture. Lower learning rates (2e-5) are assigned to the BERT encoder parameters to preserve the pre-trained contextual language understanding, intermediate learning rates are assigned to Impact Area and Genome classifiers (3e-5) and feature extractors (7e-5), while the highest learning rate (2e-4) is applied to Outcome classification layers due to its complexity. This allows fine-tuning to occur more aggressively in task-specific layers while making conservative adjustments to the foundational BERT encoder. Early stopping with a patience of 3 epochs is also implemented based on Outcome F1 scores to prevent overfitting.

Managing Loss

Three loss-focused techniques work together to address different aspects of the multi-task learning problem. Focal loss addresses class imbalance within each individual task, uncertainty weighting balances the importance between the three different tasks (Impact Area, Genome, and Outcome classification), and hierarchical consistency loss ensures predictions respect the relationships between these hierarchical categories.

Focal Loss for Class Imbalance

To address significant class imbalance, Focal Loss replaces standard Cross Entropy Loss. It adds a modulating factor that reduces the contribution from well-classified examples while emphasizing misclassified ones (Lin et al., 2020). The gamma parameter is set higher for Outcome classification (γ=4.0) than for Impact Area and Genome classification (γ=2.0), reflecting the greater imbalance in the Outcome categories. This focuses training on hard, misclassified examples rather than allowing common classes to dominate gradient updates.

Uncertainty Weighting

While Focal Loss addresses imbalance within tasks, uncertainty weighting manages the balance between the three classification tasks (Kendall et al., 2018). The model learns task-specific log variances that inversely weight each task’s loss, dynamically adjusting the relative importance of Impact Area, Genome, and Outcome classification. These learnable parameters prevent any single task from dominating the training process, which is important given the different complexities and scales of the three classification tasks.

Consistency Loss

The consistency loss term enforces structural relationships between tasks by penalizing predictions that violate the hierarchical relationships between Impact Area, Genome, and Outcome. This is implemented by constructing probability matrices that capture the co-occurrence statistics between these categories based on training data. The loss penalizes differences between direct prediction logits and predictions derived from these hierarchical relationships, encouraging the model to make predictions that respect classification hierarchy.

These techniques collectively establish a well-structured loss function that addresses within-task imbalance, between-task balance, and structural consistency, allowing the model to produce reliable predictions across all classification levels.

Managing Training Process

The model uses two dynamic training strategies that evolve throughout the training process: curriculum learning and hard negative mining. These techniques complement each other by adapting training priorities over time, with curriculum learning adjusting task importance and hard negative mining focusing on challenging examples.

Curriculum Learning

Curriculum learning gradually increases the weight of the outcome classification task over the first few epochs (Guo et al., 2018). It scales the Outcome loss weight from 1.0 to 3.0 over the first 5 epochs of training. This gradual increase provides time for the model to establish a strong foundation in the higher-level Impact Area and Genome classification tasks before shifting focus to the Outcome task. By learning the broader categories first, it can better inform Outcome classification. Initially, Outcome receives equal importance compared to other tasks, allowing the model to make balanced progress across all levels. The weight gradually increases to 3.0 to emphasize the Outcome task given its greater complexity without destabilizing the learning of the other tasks.

Hard negative mining

Hard negative mining is implemented to address the challenge of efficient learning in a dataset with significant class imbalance and varying example difficulty (Chang et al., 2017). By periodically sampling a subset (30%) of the training data, evaluating their loss values, and creating a weighted sampler that prioritizes high-loss instances, the model focuses computation on the most informative training examples. Beginning after the third epoch and recalculating every two epochs, this technique allows the model to focus on challenging cases that contribute most to classification errors while still maintaining exposure to the full dataset distribution. To manage the computational burden of this process, several memory optimization techniques are applied, including selective sampling, periodic CUDA cache clearing, and using a small batch size (4) for evaluation. These optimizations keep memory requirements manageable during each forward pass through the model while preserving the benefits of prioritized learning from difficult cases.

**BERT Model 3: Method**

Data Pre-Processing

The processing pipeline for the model requires text preprocessing to normalize text data. Special characters were cleaned, duplicate classes were combined, and all labels were encoded. The encoded dataset was then split using a modified train-test split function which duplicated single-instance classes, manually split any class that appeared twice into each dataset, and then utilized stratified sampling to divide the remaining observations so that outcomes appear in both sets in proportion to their overall frequency. This ensures that rare classes will each appear at least once in each dataset, preventing errors down the line.

A custom dataset class is then used to tokenize and convert both datasets to tensors. Those tensors were then made into data loaders to save memory during training.

Model Architecture

The architecture uses a pre-trained BERT base model (Delvin et al., 2019) as its foundational text encoder, followed by three specialized classification heads. The encoder processes program description text into general contextual embeddings. The model architecture is designed to align with the natural hierarchical structure of the provided datasets. Impact Area is the broadest category, with each containing multiple Genomes, which in turn, contain various specific Outcomes.

At the first level, the pooled output from BERT undergoes dropout regularization (0.3) and is passed into the Impact Area classifier, which is a fully connected linear layer that predicts the Impact Area.

At the second level, the Genome classifier takes that same pooled output from BERT and separately uses it to predict Genomes. Unlike a strict conditional approach, this model does not explicitly use Impact Area predictions as direct inputs to subsequent classifiers, but instead maintains shared feature representations across levels, ensuring consistency in learned representations.

The final classification layer, the Outcome classifier, similarly takes BERT’s pooled output to predict Outcome. By maintaining a unified representation from BERT across all classification heads, the model ensures efficient learning across levels while simplifying hierarchical dependencies.

This architecture uses a single BERT encoder and contains multiple independent classification heads, without enforcing any strict hierarchy or dependence within the data. This simple design allows for faster learning, and for BERT to learn dependencies among and between the observations on its own, leading to increased accuracy. This version of the model has performed better on this dataset than previously-designed models, which enforced strict conditional dependencies.

Training Strategy

Several techniques were used in the training phase to help improve OutcomeID prediction metrics. First, training on the combined synthetic and human-validated dataset increased accuracy, which is likely because the synthetic dataset was designed to help boost representation of minority classes in an attempt to mitigate harms from the severe class imbalance present in the dataset. Then, classes that only appeared once in the dataset were duplicated to ensure a stratified split would be possible. Once the single-instance classes were duplicated, any classes with only two observations were manually split, one into the train and the other into the test dataset. Finally, stratified sampling was conducted on the rest of the dataset, ensuring similar class distributions within both datasets, which is especially important when facing class imbalance.

The model uses the AdamW optimizer with a learning rate of 2e-5 and an epsilon value of 1e-8.

Focal Loss

Class imbalance is first addressed through the use of a Focal Loss function, which allows specification of severity of imbalance within a class. Since Impact Area and Genome were, relatively, less imbalanced their loss function was Focal Loss (γ = 2.0). Outcome, which suffered a more severe imbalance, had its own loss function Focal Loss (γ = 4.0).

Dynamic Loss Weighting

While Focal Loss helps remediate imbalance found within a class, it does not address the imbalance present between the classes themselves. Impact Area is easier to classify than Genome solely because there are fewer options for the model to choose from. With that logic, Outcome is much harder to classify than either Impact Area or Genome. To factor this particular imbalance into the model, class-specific loss weights are used. Starting values are assigned at 2 for both Impact Area and Genome, and 4 for Outcome. As the model trains each epoch, its validation accuracy is examined and the loss weights are adjusted accordingly. Well-trained classification levels have their weight reduced, while struggling levels become more important to the model.

**RESULTS AND CONCLUSION**

**BERT Model 1: Results**

The model achieved strong performance across all classification levels, with 87.3% accuracy for Impact Area classification, 86.3% accuracy for Genome classification, and 66.3% accuracy for Outcome classification on the validation dataset. The model attained a macro F1-score of 0.70 for the Outcome classification task, reflecting its performance across all outcome categories regardless of their frequency in the dataset.

The training metrics showed even stronger performance, with 95.2% accuracy for Impact Area, 91.8% for Genome, and 79.5% for Outcome classification on the training dataset. The gap between training and validation performance suggests some overfitting, which is expected given the complexity of the model and the imbalance across Outcome ID classes.

The best model was obtained at epoch 8, demonstrating that curriculum learning and hard negative mining continued to yield improvements in later training stages. The training loss consistently decreased from 5.27 in epoch 1 to 1.12 in epoch 8, confirming stable learning progression. The validation loss of 2.07 compared to a training loss of 1.03 indicates some overfitting, suggesting that additional regularization techniques or data augmentation could improve generalization.

Despite these challenges, the hierarchical BERT-based classification approach effectively captured the dependencies between Impact Area, Genome, and Outcome ID, which allowed us for performance improvements in predicting program outcomes.

**BERT Model 2: Results**

The model achieved strong performance across all classification levels (Figure 10), 88.76% accuracy for Impact Area classification, 83.74% accuracy for Genome classification, and 68.03% for Outcome classification on the validation dataset. The model attained a macro F1 score of 0.72 for the Outcome classification task, representing the model’s performance across all outcome categories regardless of their frequency in the dataset. The training metrics showed even stronger performance with 96.57% accuracy for Impact Area, 83.74% for Genome, and 85.56% for Outcome classification. The gap between training and validation suggests some overfitting; however, this is expected given the model’s complexity (Figure 11).

The best model was obtained at epoch 27, demonstrating that curriculum learning and hard negative mining continued to yield improvements in later training stages. The model improved gradually despite the complexity of the classification task, and challenge of balancing performance across all levels.

The validation loss of 2.0540 compared to a training loss of 0.4416 confirms the presence of some overfitting, suggesting additional regularization techniques or data augmentation could benefit future iterations of the model. The results demonstrate that the hierarchical BERT-based approach effectively captures the dependencies between the different classification tasks, achieving good performance on outcome prediction.

**BERT Model 3: Results**

The model exhibited strong classification abilities, achieving high accuracies at all levels of the hierarchy (Figure 13). It attained 88.66% accuracy for Impact Area classification, 84.45% for Genome classification, and 66.8% for Outcome ID classification. It also achieved an F1 score of 0.67 for Outcome ID classification, which indicates strong performance across Outcome IDs despite their severe imbalance in relative representations in the dataset. The training metrics reflected even stronger performance, with 96.11 % prediction accuracy for Impact Area, 93.77% accuracy for Genome, and 82.16% accuracy when predicting Outcome ID. The discrepancy in accuracy may indicate overfitting, but could also be due to the complexity of the model and the severe imbalances in the datasets.

The dynamic loss weights proved to be especially helpful. As seen in the outputs (Figure 14), (Figure 15), the adjustment of loss weights allowed the model to focus more on training the Outcome classifier once the other two produced satisfactory results. This helped the model achieve strong results after only twelve epochs.

The difference in training loss (0.999) and validation loss (3.447) indicates that the model struggles with overfitting, so future work should take even more steps to combat it. Still, the results indicate that despite not having any strictly-enforced hierarchy or dependence, this model was able to effectively learn dependencies among the observations on its own and use them to achieve good performance on Outcome ID prediction.

| **Performance Metrics** | **Logistic Regression** | **BERT Base** | **BERT Model 1** | **BERT Model 2** | **BERT Model 3** |
| --- | --- | --- | --- | --- | --- |
| Impact Area Accuracy | N/A | N/A | 87.3% | 88.8% | 88.7% |
| Genome Accuracy | N/A | N/A | 86.3% | 83.8% | 84.45% |
| Outcome ID Accuracy | 38.55% | 0.2358 | 66.3% | 68.0% | 66.8% |
| F1 Score | 0.413 | 0.0900 | 0.70 | 0.72 | 0.67 |
| Precision | 0.527 | 0.0909 | 0.72 | 0.76 | 0.72 |
| Recall | 0.386 | 0.1406 | 0.73 | 0.73 | 0.67 |

Our best BERT-based classifier demonstrated 88.8% accuracy when predicting Impact Area, 83.8% accuracy when predicting Genome, and 68.0% accuracy when predicting Outcome ID. The model’s performance varied across different outcome categories.

Using synthetic data to address the class imbalance issue helped improve the overall accuracy of our models. More specifically, the synthetic data improved the model’s ability to predict the minority classes that were underrepresented in the original dataset. Training the model using a combination of the original data and synthetic data resulted in an F1 score of 0.6722, while training on only the original data resulted in an F1 of 0.3933.

Despite the efforts to correct class imbalance, including a focal loss function and training on synthetic data, there were still some minority classes which were never predicted. Out of nearly 300 outcome IDs, 24 of them were never predicted by the model. Out of those 24, 20 of the outcome IDs only appeared a single time in the dataset, which had over 10,000 observations. Still, identifying those outcome IDs provides valuable information for future model design and testing.

BERT, with its contextual embeddings, offered significantly enhanced classification accuracy when compared to other Natural Language Processing methods, such as Google Vertex AI and Logistic Regression.

To improve upon our work in the future, we could incorporate domain-specific embeddings or experiment with ensemble modeling. We could also try implementing a more sophisticated class-weighting strategy, such as dynamically adjusting loss weights based on real-time misclassification rates. This could help further address the class imbalance in the dataset.

**AUTHOR CONTRIBUTIONS**

Julia’s contributions:

Conducted exploratory data analysis and fit a flat BERT text classifier. Constructed several hierarchical BERT classifiers, focusing on methods to improve accuracy including hyperparameter tuning and several different model structures. Evaluated the models and saved the best-performing one. For the paper, contributed to methods, results and future work sections.

John’s contributions:

In addition to working on code for EDA, Preprocessing, baseline BERT model, and Logistic Regression model, I was also responsible for most of the administrative work related to this project. I coordinated meetings with our team and with Professor Hecktman, I created the templates for our project deliverables, and was responsible for all submissions in D2L.

Arham’s contributions:

Built and fine-tuned the BERT model to improve classification accuracy. Conducted exploratory data analysis to understand dataset characteristics and optimized hyperparameters and evaluated model performance using various metrics. Cleaned data throughout the course of the project as issues arose. Evaluated different models and analyzed their performance across Impact Area, Genome, and Outcome ID classification.

Natalie’s contributions:

Conducted exploratory data analysis, pre-processing, and experimented with BERT classification model architecture, weighting, and hyperparameter tuning to improve outcome prediction. Evaluated model performance and saved model. Built a pipeline to make predictions using saved models. Contributed to project organization. For paper, primarily contributed to abstract and literature review, references. Also contributed to methods and results.

**FUTURE WORK**

While we tried a multitude of strategies to remedy the class imbalance in the dataset, it still proved to be a problem, affecting our accuracy. If we were to continue working on this data, we would get even more specific with our approach, assigning different class imbalance remedies depending on the degree of imbalance overall, and within each impact area and respective genome. We could also merge similar less-popular outcomes for general prediction, and then fit a separate model to split the category of programs into their proper outcomes. While this process would be very labor-intensive, such specific fine-tuning may be what the model needs in order to assign labels more accurately.

We could also generate more synthetic data from the provided code and train our model on an even larger dataset. Training on the provided synthetic dataset, once combined with the human-validated data we were originally given, greatly improved model accuracy when compared to solely training on human data. Generating and training on an even larger synthetic dataset, especially one with observations meant to alleviate some of the class imbalance in the original dataset, could further improve model performance. There are some combinations of traditional NLP models and BERT that would be interesting to evaluate in comparison with our model.

While there are techniques that may help, the class imbalance in the dataset proved to be a major shortcoming of the data. There is no consistent number of subclasses in each class of the hierarchy designed by Impact Genome, and the way in which their labels are assigned may have contributed to some of the imbalance found in the dataset. For example, if two genomes were observed nearly as frequently as one another, but one contained many possible outcomes and the other only a few, the prevalence of specific outcomes can be artificially inflated or diffused. There is no consistent distribution of programs at any level of the hierarchy,

and no consistent distribution of labels either, but the label inconsistency does not combat the inconsistency found between the outcomes.

Our models can be applied to a wide variety of text classification tasks, given that the input text is not excessively long. They could be useful for hierarchical sentiment analysis of online discussions, such as those within comment sections on platforms like YouTube or Instagram. Our models could also help classify reviews left on platforms such as Yelp or Tripadvisor, where large categories of businesses can be broken down into more and more specific establishments. It would be very interesting to see how our models perform on data that may be more balanced or may have fewer potential labels.

**FIGURES AND GRAPHS**

Figure 1: Screenshot of the first 15 rows of the dataset

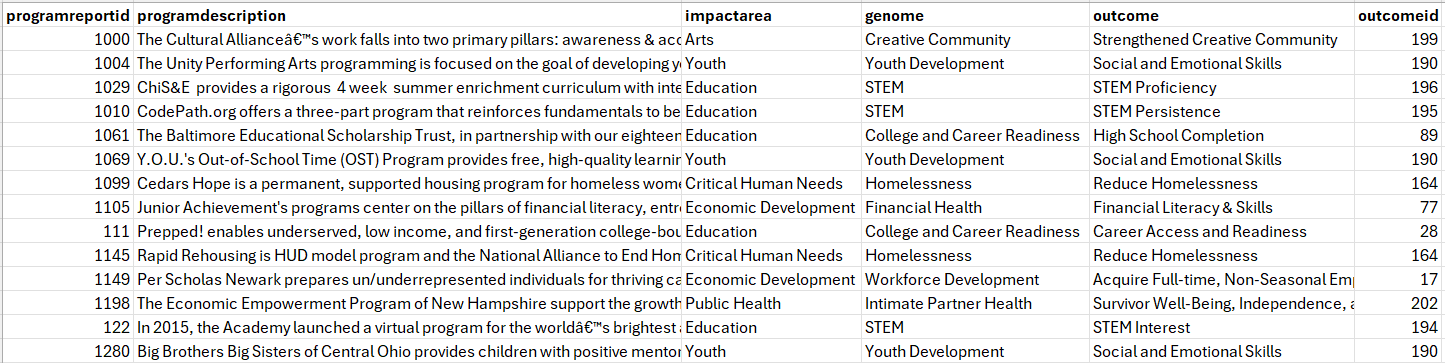


Figure 2: Histogram of Outcome ID counts

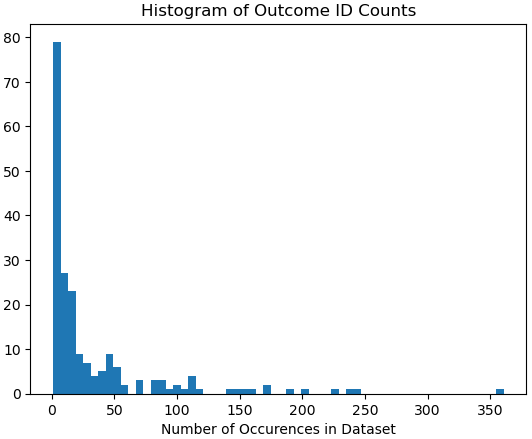


Figure 3: Descriptive statistics for outcome ID and the frequency (count) of each outcome ID

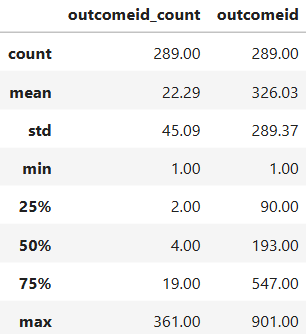


Figure 4: Screenshot of program descriptions with special characters (highlighted) that were removed during preprocessing

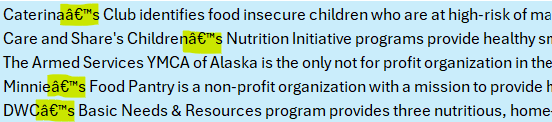


Figure 5: Heat map of counts for different combinations of impact area and genome

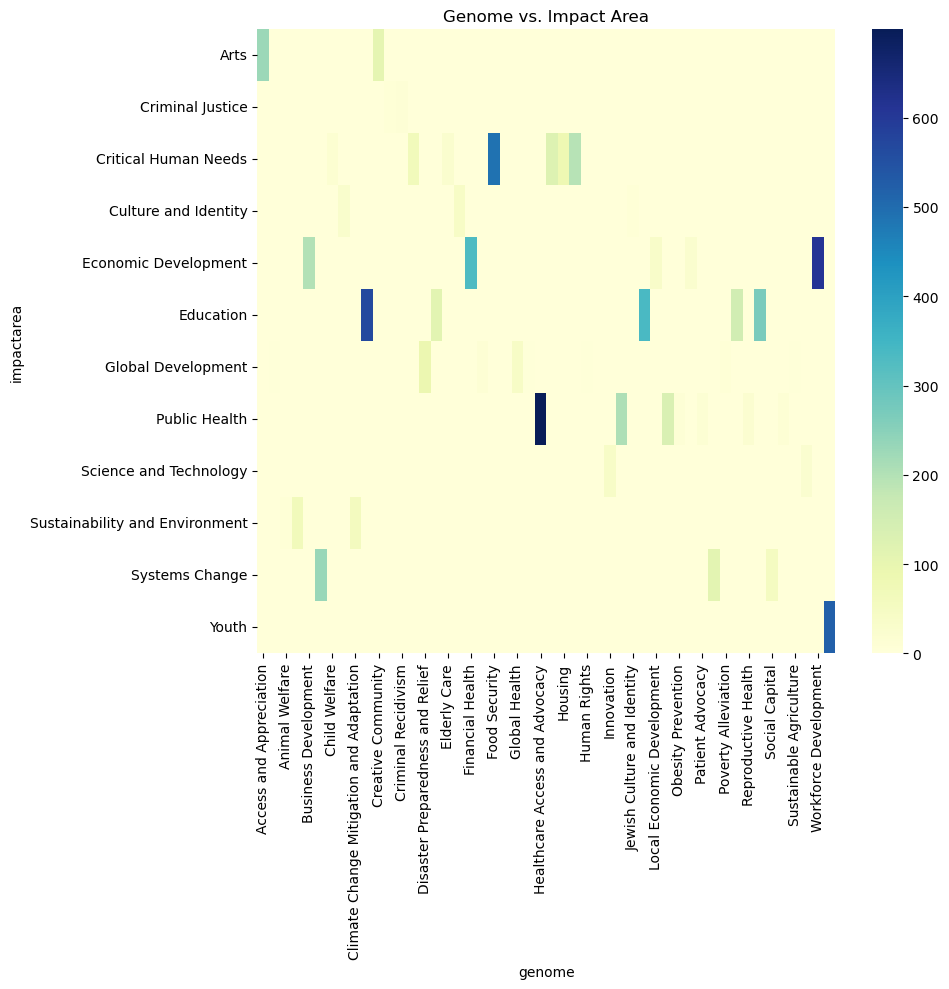


Figure 6: Bar chart showing the 25 genomes with the most programs

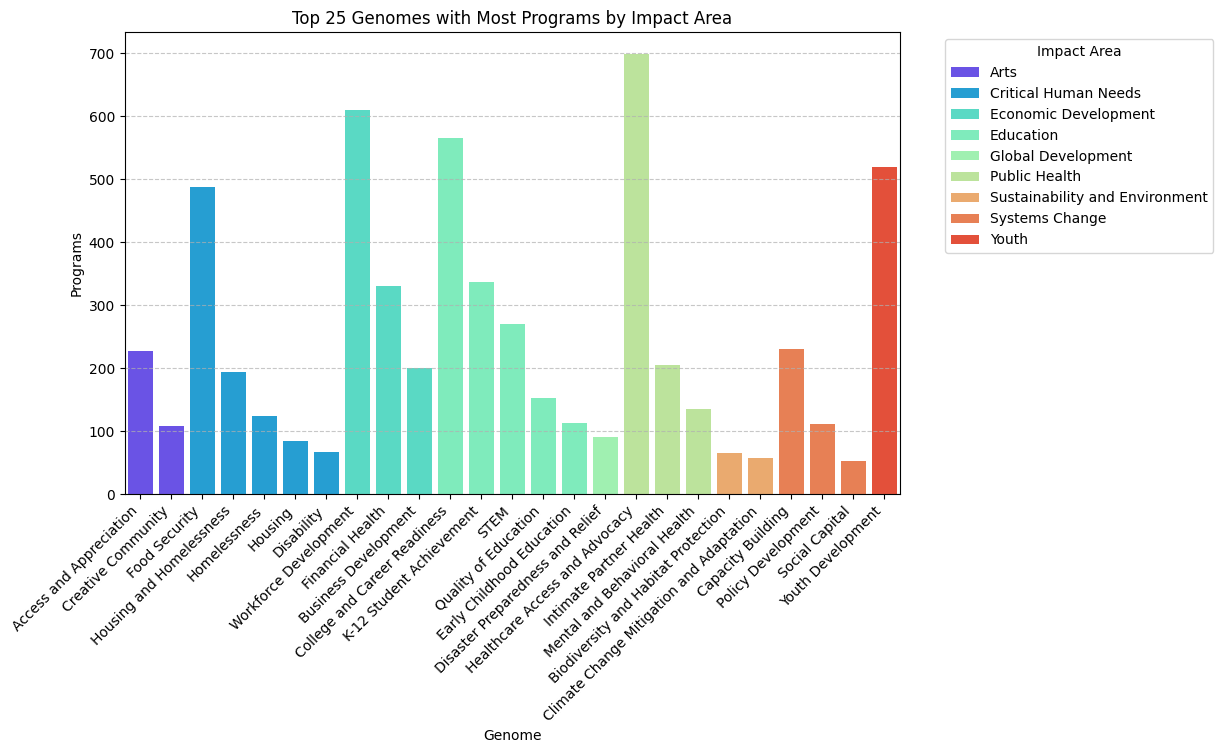


Figure 7: Bar chart showing the 25 genomes with the fewest programs

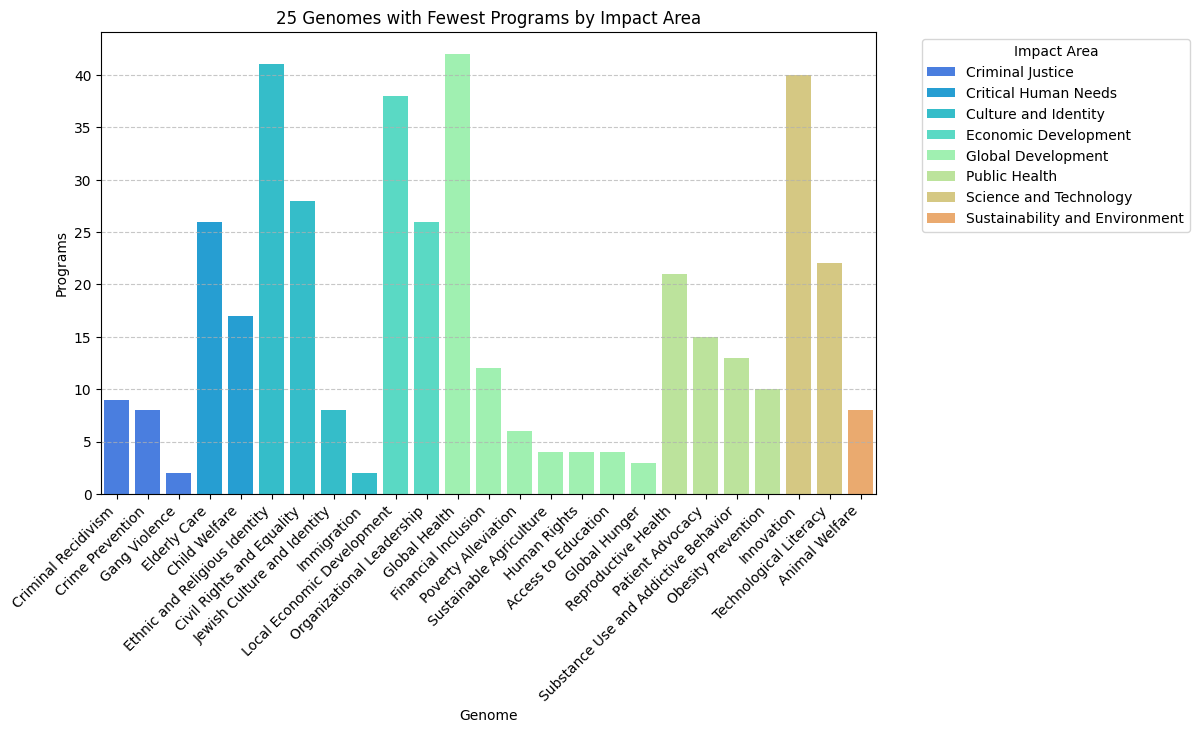


Figure 8: Word cloud of text from all program descriptions

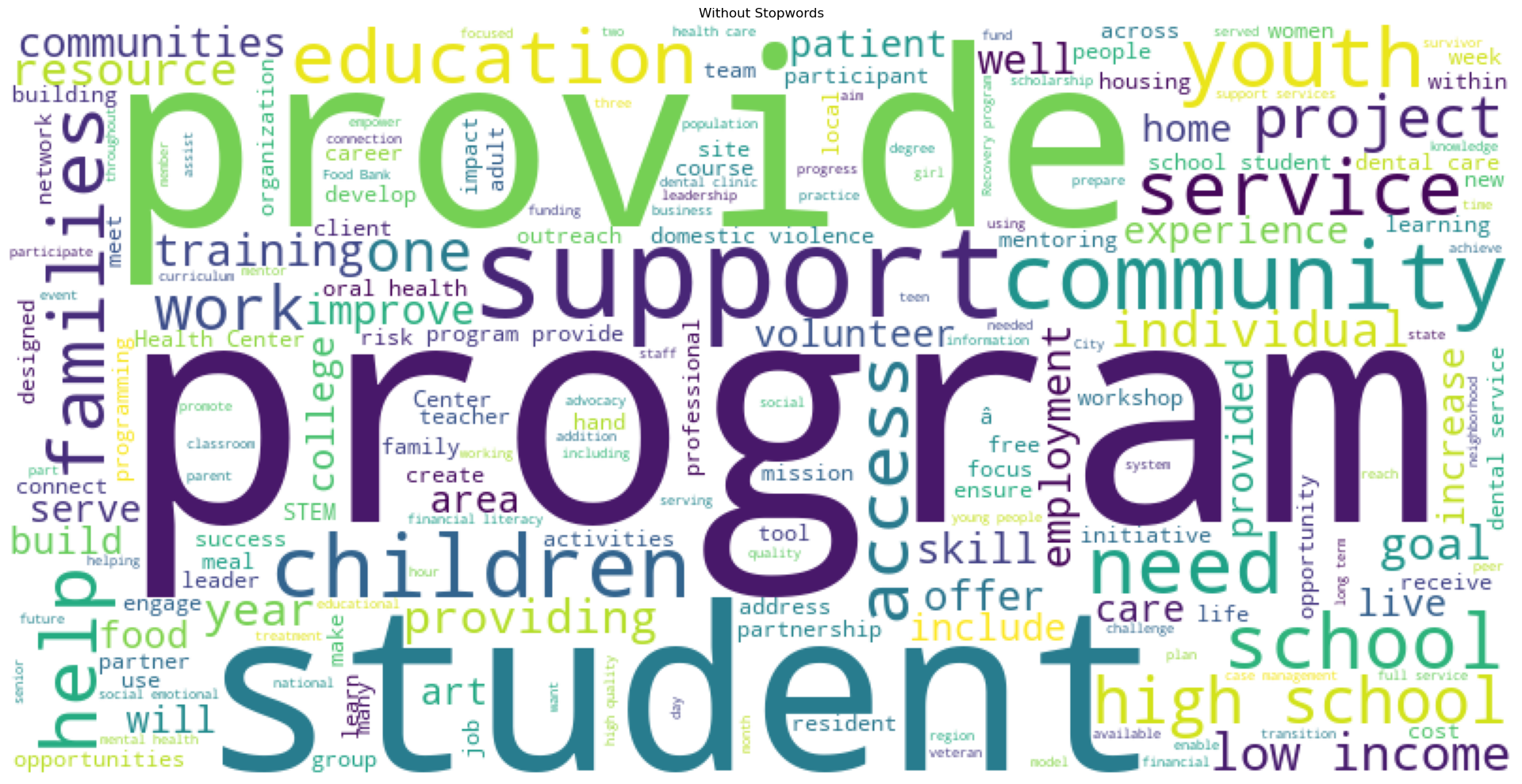


Figure 9: Model 2 Architecture.

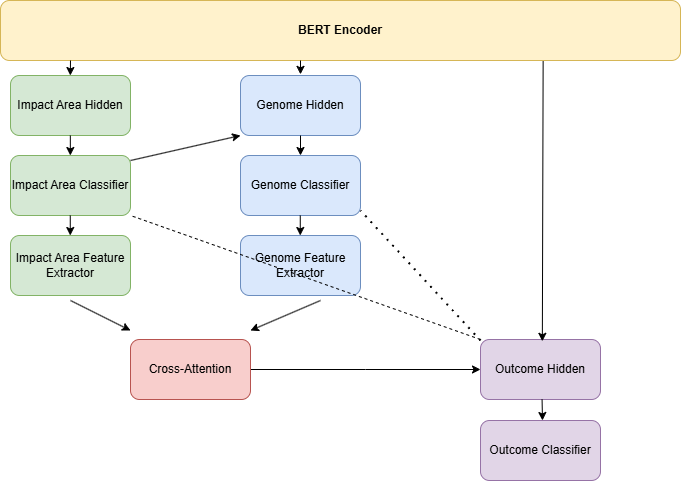


Figure 10: Model 2 Results.

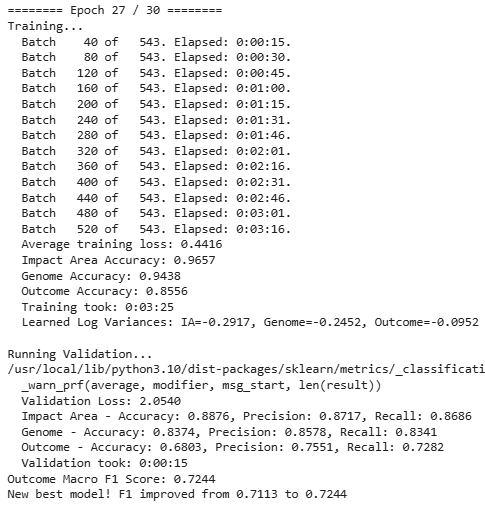


Figure 11: Model 2 Training and Validation Loss, Outcome F1 Score.

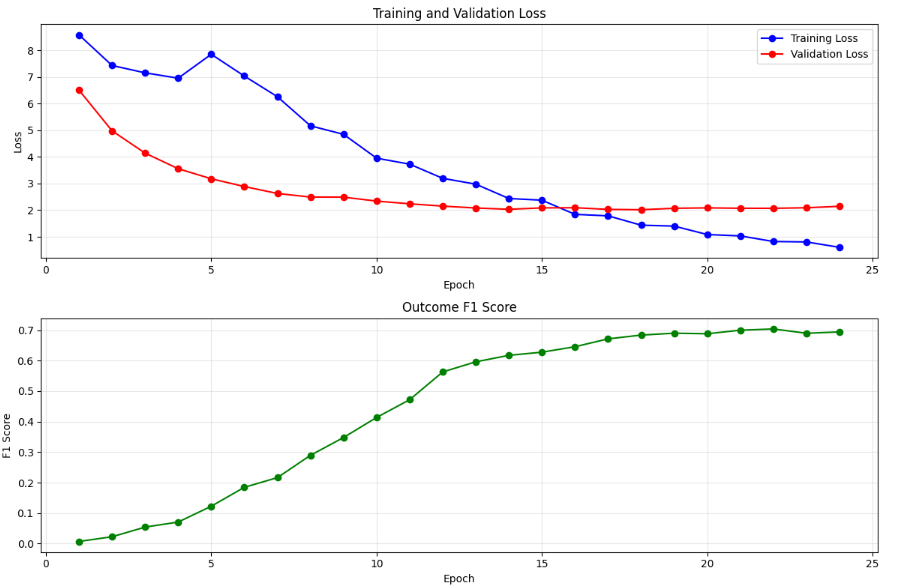


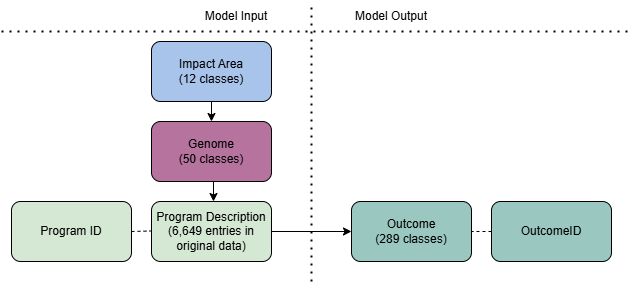
Figure 12: Model input/output.  


Figure 13: Model 3 Results.

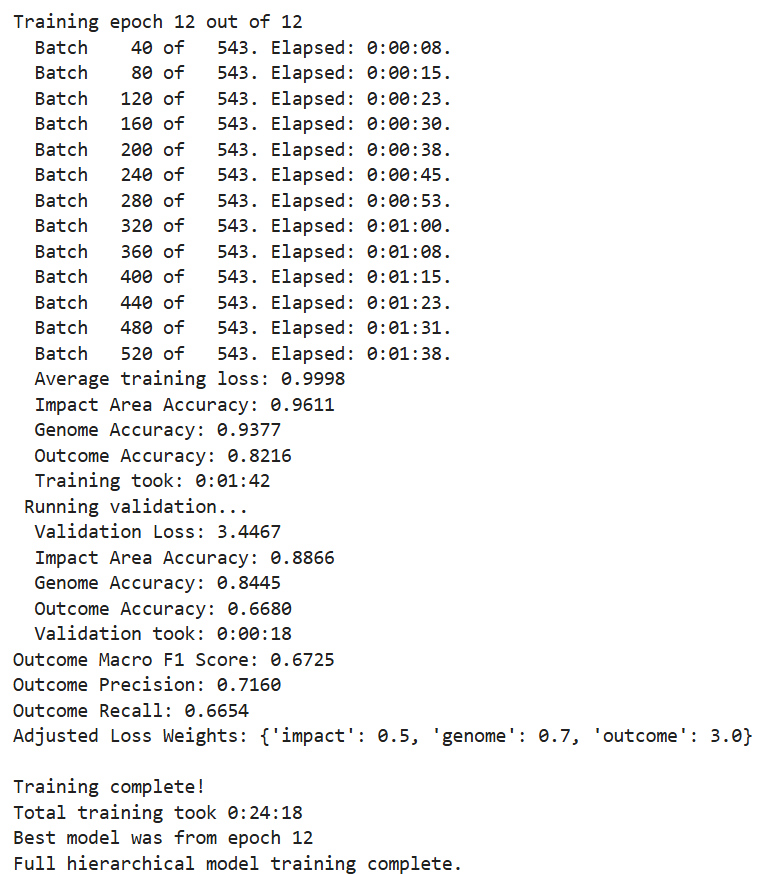


Figure 14: Model 3 Validation Accuracies.

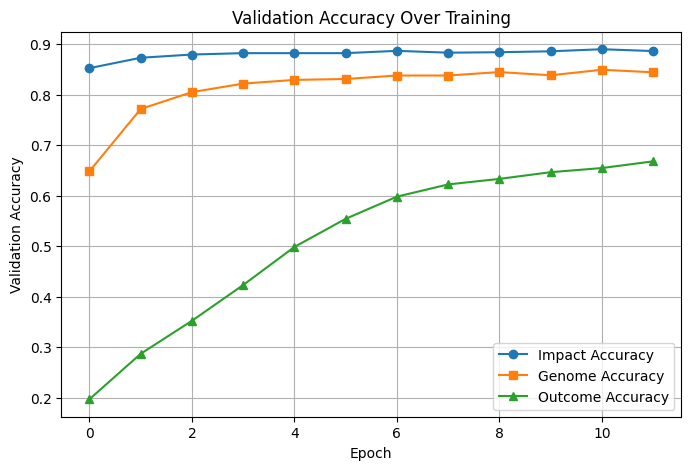
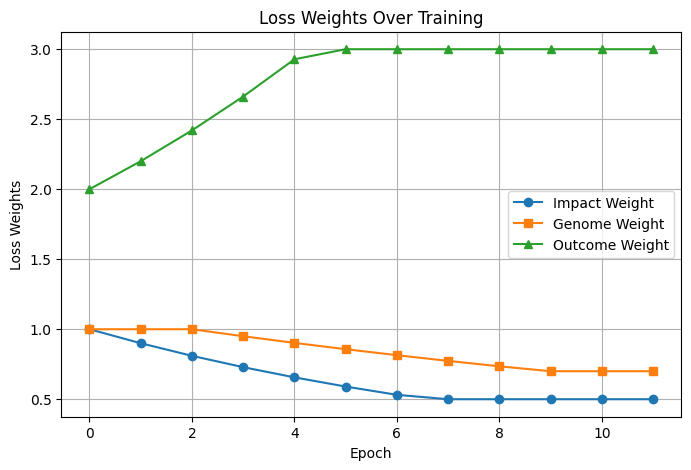


Figure 15: Model 3 Loss Weights.



**REFERENCES**

A. Vaswani, N. Shazeer, N. Parmar, L. Uszkoreit, J. Kaiser, M. Polosukhin, A. Devlin, M. Lee, S. Tsai, H. Zhou, and J. Kaiser, "Attention is All You Need," Proc. of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, 2017, pp. 6000-6010. <https://doi.org/10.48550/arXiv.1706.03762>.

Ceccon, T. S. (2023). Applying BERT language model to poem classification: a study on data imbalance issues. *Lume*. https://doi.org/https://lume.ufrgs.br/handle/10183/259959

Chang, H.-S., Learned-Miller, E., & McCallum, A. (2017). Active Bias: Training a More Accurate Neural Network by Emphasizing High Variance Samples. *arXiv*. <https://doi.org/http://dx.doi.org/10.48550/arXiv.1704.07433>

Chi, S., Qiu, X., Xu, Y., & Huang, X. (2019). How to Fine-Tune BERT for Text Classification? *ResearchGate*. <https://doi.org/10.48550/arXiv.1905.05583>

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 4171–4186. <https://doi.org/10.18653/v1/n19-1423>

Dhina, M. M., & Sumathi, S. (2022). An innovative approach to classify hierarchical remarks with multi-class using Bert and customized naïve Bayes classifier. *International Journal of Engineering, Science and Technology*, *13*(4), 32–45. <https://doi.org/10.4314/ijest.v13i4.4>

Ganganwar, V., & Rajalakshmi, R. (2023). Employing synthetic data for addressing the class imbalance in aspect-based sentiment classification. *Journal of Information and Telecommunication*, *8*(2), 167–188. <https://doi.org/10.1080/24751839.2023.2270824>

Garrido-Merchan, E. C., Gozalo-Brizuela, R., & Gonzalez-Carvajal, S. (2023). Comparing Bert against traditional machine learning models in text classification. *Journal of Computational and Cognitive Engineering*, *2*(4), 352–356. <https://doi.org/10.47852/bonviewjcce3202838>

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A survey on text classification algorithms: From text to predictions. *Information*, *13*(2), 83. <https://doi.org/10.3390/info13020083>

Ghosh, K., Bellinger, C., Corizzo, R., Branco, P., Krawczyk, B., & Japkowicz, N. (2022). The class imbalance problem in deep learning. *Machine Learning*, *113*(7), 4845–4901. <https://doi.org/10.1007/s10994-022-06268-8>

Guo, M., Haque, A., Huang, D.-A., Yeung, S., & Fei-Fei, L. (2018a). Dynamic Task Prioritization for Multitask Learning. *Lecture Notes in Computer Science*, 282–299. <https://doi.org/10.1007/978-3-030-01270-0_17>

Guo, M., Haque, A., Huang, D.-A., Yeung, S., & Fei-Fei, L. (2018b). Dynamic Task Prioritization for Multitask Learning. *Lecture Notes in Computer Science*, 282–299. <https://doi.org/10.1007/978-3-030-01270-0_17>

Kendall, A., Gal, Y., & Cipolla, R. (2018). Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantic. *arXiv*. <https://doi.org/https://doi.org/10.48550/arXiv.1705.07115>

Koroteev, M. (2021). BERT: A Review of Applications in Natural Language Processing and Understanding. *arXiv*. https://doi.org/ <https://doi.org/10.48550/arXiv.2103.11943>

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2020). Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(2), 318–327. <https://doi.org/10.1109/tpami.2018.2858826>

McCormick, C., & Ryan, N. (2019, July 22). *Bert fine-tuning tutorial with pytorch*. BERT Fine-Tuning Tutorial with PyTorch · Chris McCormick. <https://mccormickml.com/2019/07/22/BERT-fine-tuning/#41-bertforsequenceclassification>

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep Learning Based Text Classification: A Comprehensive Review. *arXiv*. <https://doi.org/https://doi.org/10.48550/arXiv.2004.03705>

Nogueira, R., Yang, W., Cho, K., & Lin, J. (2019). Multi-Stage Document Ranking with BERT. *arXiv*. <https://doi.org/https://doi.org/10.48550/arXiv.1910.14424>

Padurariu, C., & Breaban, M. E. (2019). Dealing with data imbalance in text classification. *Procedia Computer Science*, *159*, 736–745. <https://doi.org/10.1016/j.procs.2019.09.229>

Prahbu, S., Mohamed, M., & Misra, H. (2021). Multi-class Text Classification using BERT-based Active Learning. *arXiv*. https://doi.org/https://doi.org/10.48550/arXiv.2104.14289

Preeti. (2024, November 13). *Data augmentation techniques for imbalanced datasets*. Medium. <https://medium.com/%40preeti.rana.ai/data-augmentation-techniques-for-imbalanced-datasets-9214374ffe44>

*Reducelronplateau*. ReduceLROnPlateau - PyTorch 2.6 documentation. (n.d.). <https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html>

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv*. <https://doi.org/https://doi.org/10.48550/arXiv.1908.10084>

Shaikh, S., Daudpota, S. M., Imran, A. S., & Kastrati, Z. (2021). Towards improved classification accuracy on highly imbalanced text dataset using Deep Neural Language models. *Applied Sciences*, *11*(2), 869. <https://doi.org/10.3390/app11020869>

Tayyar Madabushi, H., Kochkina, E., & Castelle, M. (2019). Cost-sensitive Bert for generalisable sentence classification on Imbalanced Data. *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*. <https://doi.org/10.18653/v1/d19-5018>

Tenney, I., Das, D., & Pavlick, E. (2019). Bert rediscovers the classical NLP pipeline. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. <https://doi.org/10.18653/v1/p19-1452>

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., & Moi, A. (2020). *Transformers: State-of-the-Art Natural Language Processing*. <https://arxiv.org/pdf/1910.03771>