Small-Scale Patient-Physician Communication Text Classification: Improved Performance through Advanced Techniques

Heyuan Huang, Gaoqianxue Liu, Ji Qi, Yingxue Pan

May 3, 2023

1 Abstract

In this study, we present an approach to classifying Patient-Physician Communication (PPC) text messages using both machine-learning-based models and the BERT model. We enhance our machine-learning-based models by extracting more interpretable features through chi-square tests and 4-fold binary classifications, as well as addressing data insufficiency and imbalance issues using data augmentation techniques such as SMOTE, GPT-4 augmentation, and Easy Data Augmentation (EDA). Furthermore, we improve our BERT model by employing the Bio-ClinicalBERT pretrained model. Our findings demonstrate that these improvements significantly increase the overall accuracy, precision rate, and recall rate of both machine-learning-based models and BERT models from the baseline models in classifying PPC text messages.

2 Introduction and Background

Patient-Provider Communication (PPC) refers to the analysis and interpretation of the language used by patients and healthcare providers in the context of a clinical encounter. Effective and rapid PPC facilitates patient adherence monitoring and timely reaction to changes in health, and as a result, it improves patient-centered outcomes. In this project, we aim to automate the process of categorizing electronic PPC messages by building a variety of machine-learning models and natural language processing models and improving their performance.

The original dataset was provided by Professor Samah Fodeh. It contains 275 secure messages between patients and providers/clinicians. The data was stored in an .xlsx with 11 sheets: "Info Giving - Patient", "Info Giving - Clinician", "Info Giving - Auto", "Info Seek - Clinician", "Info Seek - Patient", "Emotion - Patient", "Emotion - Clinician", "Partnership - Clinician", "Partnership - Patient", "SDM - Patient", and "SDM - Clinician". These sheet names correspond to the manually-coded labels given to the messages. These labels are coded both by the function of each message and by the author of each message. There are five functions: Information Giving, Information Seeking, Emotion, Partnership, and Shared Decision-making (SDM). All five functional categories consist of messages written by patients and clinicians (denoted by "Patient" and "Clinician" respectively), while the Information Giving class also contains messages automatically generated by the secure message system (denoted by "Auto").

3 Data Visualization

We wanted to get some basic insights into the dataset so we first visualized the number of messages in each category. The result is depicted in 1. By looking at this bar plot, we found that the dataset is greatly imbalanced, with more than 150 samples in the largest class "Info Giving" and only a few samples in the smallest class "SDM". We anticipated that this significant imbalance could potentially compromise the model performance and tried to address this issue after building our baseline models.

We also wanted to better understand the differences between categories so we visualized the distribution of message lengths for each category. The result is depicted in 2. We found that the message length distribution is quite different across categories.

Finally, we explored the top 10 semantic keywords for each category to better understand the differences between the functions. We found that when we didn't restrict the document frequency of keywords, the word "dr" appears in the list of top 10 semantic keywords for 7 out of 11 categories. It is natural that the abbreviated word for "doctors" is significant in data produced in the clinical context and it doesn't tell us anything about the categories. Then we tried to restrict the document frequency to 0.7 and the result is more informative. E.g., for the category "Info Seek - Clinician", the top 10 semantic keywords are 'let', 'port', 'start', 'lacy', 'thanks', 'morning', 'procedure', 'able', 'reached', and 'results'. The word 'let' is probably from the phrase 'let me know', which corresponds with the function of information seeking.

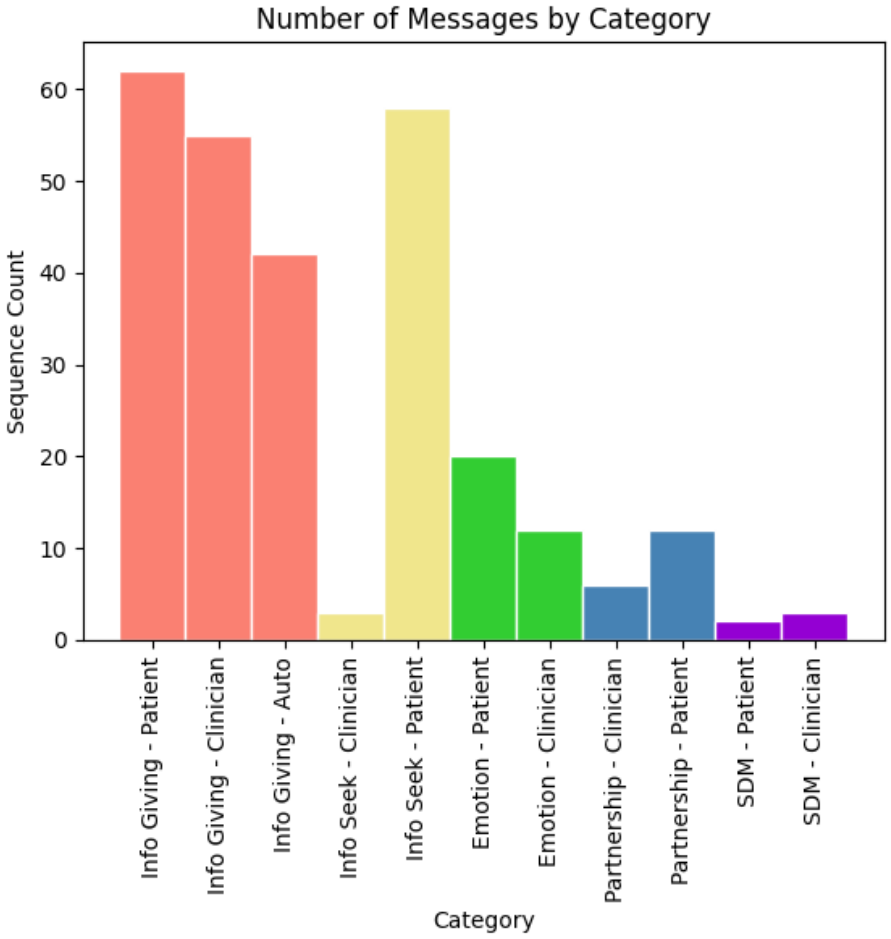


Figure 1: The number of messages in each category in the original data. Messages that have the same function are in the same color.

4 Machine learning methods

4.1 Text Data Preprocessing

We build our baseline multi-class machine learning models on the original dataset.

There are five classes in the original dataset: Information Giving, Information Seeking, Emotional Support, Partnership, and Shared Decision-Making. However, the Shared Decision-Making class only has 5 messages in total, from the Patient and Clinician sides. In this stage, we removed this class from our dataset to build our initially rudimentary 4-class baseline machine learning models.

Each class has messages from the Patient, Clinician and/or system Automatic sides. Message numbers counted by Label and Code are shown in 1 and 4. Code A stands for system Automatic message, C stands for Clinician and P stands for Patient.

The Emotional Support class has 32 messages, the Information Giving class has 159 messages, the Information Seeking class has 61 messages, and the Partnership class has 18 messages. In total, we have 270 messages for the four classes and the vocabulary size is 1537.

We use the NLTK package to lemmatize our text data's words in order to restore different inflected forms of one word. For example, after applying lemmatization, the word 'corpora' in our input text is restored to 'corpus'. By lemmatization, we can reduce the vocabulary size by grouping inflected words together without losing much semantic information.

4.2 Feature Extraction

We first select Term Frequency-Inverse Document Frequency (TF-IDF) as our classification feature. Term frequency, $tf(t, d)$, is defined as the frequency of term t in one document d:

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

where t' stands for all words in the document d, $f_{t,d}$ means the counts of word t in the document d. It is large when the word t occurs many times in document d.

Inverse document frequency, $idf(t, D)$, is defined to measure the uniqueness of a term t to some specific documents:

$$idf(t, D) = \log \frac{N}{|d \in D : t \in d|}$$

where N is the number of documents in corpus D, and the denominator is the number of documents that contain the term t. If the term doesn't appear in any documents, the denominator will be added 1 to avoid division by zero. It is large when the term t appears at a low frequency in documents. That is, rare terms lead to high idf values.

The TF-IDF feature is defined as the product of these two measures:

$$tf - idf(t, d, D) = tf(t, d) * idf(t, D)$$

It is high only when the term t is frequent in document d and not frequent in all other documents in D.

After lemmatization, we use TfidfVectorizer() function from the sklearn package to extract document-specific terms as our features. For the function parameters, we set ngram_range=(1,3) to include unigram, bigram and trigram features into consideration; max_df=0.75 to filter out common words that appear in many messages with a frequency over 75%; max_features=1000 to select n-gram features with the top 1000 high tf-idf values.

We use the t-SNE package to project our 1000-dimension data in a 2-dimension figure 4.

All data are distributed together, making boundaries hard to capture.

4.3 Model Construction

Our current feature matrix has the shape (270,1000), where 270 is the number of data, 1000 is the dimension of features. We split the data into a train set and a test set with a proportion of 0.75:0.25 in a stratified fashion. Hence, we have 202 samples for the train set and 68 samples for the test set.

We tried Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, XGBoost models as classifiers. Here we use the Random Forest classifier as an example and the complete performance metrics are listed in section 6: Baseline Results. Figure 5 is the confusion matrix of the Random Forest Classifier, from which we can calculate that its accuracy is 0.5, precision is 0.333, recall is 0.5, and F1-score is 0.4. All the 5 machine learning models on the original data feature matrix didn't achieve good performance, whose accuracy is between 50% to 57%.

In order to find the reason for poor performance, we printed the top 200 features with the highest tf-idf values, shown in Figure 6. Surprisingly, the top 200 features are all numerical terms that only appear in one single document frequently, but don't appear in all other documents, leading to high tf-idf values. Although they are highly document-specific, they don't contribute any semantic meaning to the text classification feature. Without enough informative features, machine learning models can't recognize patterns to distinguish classes.

To solve this problem, we propose two solutions:

1. Use regular expressions to recognize meaningless numeric terms such as phone numbers, zip codes, etc, and replace these terms with special tokens such as <NUMBER>.

2. Try another feature selection method, the chi-square test, rather than simply choosing features with the highest tf-idf values, to select more informative features.

5 BERT methods

Besides machine learning models, we also constructed two Bert models, namely the Baseline Bert model and the Improved Bert model. The Baseline Bert model was created using the 'Bert-base-uncased' pretrained model, whereas the Improved Bert model was developed based on the 'Bio-ClinicalBERT' pretrained model that specializes in clinical [1].

For Baseline Bert model, we used the original dataset as input and developed a 4-class classifier, which included "Emotion", "Info Giving", "Info Seeking", and "Partnership" categories. To prepare the dataset for analysis, we utilized stratified splitting, which divided the original dataset into training, validation, and testing sets, with a ratio of 0.7/0.15/0.15. Subsequently, we used BertTokenizer to tokenize the datasets, leveraging the pretrained model 'Bert-base-uncased'. The tokens were then converted into tensors and used to generate a tensor dataset, which was formatted into three dataloaders. Afterwards, we imported the 'Bert-base-uncased' pretrained model and constructed a classifier by implementing forward pass and backpropagation algorithms, using "AdamW" as the optimizer. The diagram illustrating this process is presented in 7.

The Improved Bert model is built with the augmented dataset. It follows the same pipeline as the Baseline Bert model 8. However, instead of the 'Bert-base-uncased' pretrained model, we utilized "Bio-ClinicalBERT". Moreover, instead of implementing forward pass and backpropagation, we employed the Trainer class. With these modifications, we were able to classify all five categories of the texts.

6 Baseline Results

6.1 Machine Learning Baseline Results

Based on the top 1000 highest tf-idf score features, the five machine learning models we experimented with, i.e., Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, XGBoost, had similar performance with mean accuracy from 50% to 57%. They all have the tendency to predict most test samples to Information-Giving class. Their classification metrics, except aforementioned Random Forest, are shown in the figure 9, 10, 11, 12.

6.2 BERT Baseline Results

The performance of the Baseline Bert model is satisfactory in terms of identifying "Information Giving" texts, but it exhibits extremely poor results for the remaining three categories. The accuracy of the validation set converges to 0.55, and for the test set, it is 0.54. 13 displays the loss curves for both the training and validation sets across 20 epochs, as well as the accuracy curve for the validation set. The confusion matrix and the classification metrics (precision, recall, f1-score, and support) for each class are presented in 14.

7 Improvement and Results

7.1 Feature Extraction by Chi-square Test and Binary Classifications

Section 4.3 revealed that the top features with the highest tf-idf values are semantically meaningless. To extract more relevant and interpretable features that are closely associated with the labels, we conducted a chi-square test and 4-fold binary classifications, and the results are illustrated in Figure 15. The following steps were taken for feature extraction using the chi-square test and binary classifications:

For each MODEL in the set logistic regression, decision tree, random forest, gradient boosting, XGBoost:

- Employ the MODEL for binary classification of TASKs (such as Info-giving vs. Non-Info-giving, Info-seeking vs. Non-info-seeking, Emotion vs. non-emotion, and Partnership vs. non-partnership).
- Extract features from the text messages using the TF-IDF vectorizer.
- Use the chi-square test for feature selection and select the top K features.
- Divide the dataset into training and testing sets.
- Train a MODEL classifier using cross-validation. Evaluate the trained classifier on the testing set.

After performing the binary classification tasks, we conducted a multi-class cross-validated classification using the top $4K$ selected features from all the binary classification TASKs combined.

Compared to the baseline features, the features in Figure 15 are more suitable for the classification tasks and more interpretable. For instance, "challenging" is a critical feature in distinguishing between the classes "Emotional Support" and "Non-Emotional." Hence, we decide to use the chi-square test and 4-fold binary classifications to extract features and train classifiers in the following machine learning based models.

7.2 GPT-4 Generated Data Augmentation

The utilization of Large Language Models (LLMs) such as GPT-3 can enhance short text classification performance by generating supplementary training samples and training classifiers on an expanded dataset that includes these samples, resulting in greater consistency when classifying unseen samples [4]. To further strengthen our classifiers and resolve the data insufficiency and imbalance issue, we incorporate the latest version of GPT, GPT-4, to learn the data features and message characteristics for each category. To achieve this, we direct GPT-4 to generate 150 messages for each category, based on its knowledge of PPC and the existing messages from our dataset. The augmented dataset for each category includes the original messages as well as an additional $|L|$ GPT-4 generated messages, where L is a four-dimensional vector with L_i corresponding to the number of randomly generated samples introduced for the i_th category $(0 \leq L_i \leq 150)$[3]. We employ cross-validation to ascertain the best hyperparameter value for L. This involves assessing the accuracy of the model on a test set that exclusively contains messages from the original data.

16 presents a comparison of the performance and confusion matrix between the XGBoost model without GPT-generated messages and the one that includes GPT-generated messages (GPT-augmented data). Our model, trained with the GPT-augmented data, exhibits superior performance across almost all the performance metrics in comparison to the XGBoost model without GPT-generated messages. Moreover, the predictions of our augmented model are less skewed towards a single class.

On the other hand, the GPT-augmented method also has a significant drawback: it introduces bias from the GPT model itself. As depicted in Figure 17, the t-SNE plots of the optimal GPT-augmented data reveal that while more samples for each class are closer to the original samples generated by GPT, thereby increasing the statistical power of the classifier, the embedding of GPT-generated messages is highly concentrated in the lower-left quadrant, making the classifier more biased. Therefore, it is imperative to control the number of GPT-generated samples introduced to the augmented data to trade off between statistical power and bias.

7.3 Easy Data Augmentation

Another method that we used to address class imbalance is Easy data augmentation (EDA). EDA is a technique that generates new samples from the existing data using four operations: Synonym replacement, Random Deletion, Random Swap, and Random Insertion [2]. Synonym replacement replaces n words in the sentence with synonyms from Wordnet. It is intuitive why this operation could work: it produces new sentences that are semantically equivalent to the existing sentence. However, it is not intuitive why the other three operations could also work: the generated sentences are not grammatically or semantically correct to humans. According to the authors, these operations introduce a reasonable amount of noise that prevents overfitting while also retaining the enough number of the original words to not change the overall meaning of the original sentence.

Using EDA, we augmented the three minor classes "Partnership", "Emotion", and "SDM" by generating 36, 32, and 55 new samples respectively. As a result, combining the original data and augmented data, the minor classes have about the same number of samples as the class "Info Seeking", which contains 62 samples.

We trained the machine-learning models using the new data and drew the T-SNE plots again (Figure 18). Unlike the data generated by GPT-4, each EDA-augmented message tends to cluster around its corresponding original message, and thus the augmented data don't converge on the embedding space like GPT-4 generated data do as we mentioned above.

7.4 Improved Results

7.4.1 Improved Machine Learning Results

After combining Easy Data Augmentation and GPT-4 generated data, we obtained a more balanced dataset with a larger amount of samples. The final dataset contains 1086 sentences with the vocabulary size of 2236. The Emotional Support class has 139 messages, Information Giving class has 361 messages, Information Seeking class has 150 messages, Partnership class has 131 messages. The total sample number is 781. We also split the train set and test set with a proportion of 0.75:0.25. That is, the train set has 585 samples and the test set has 196 samples.

We conduct 4 binary classification tasks and one 4-class classification task on the five machine learning models. All models' performance metrics have improved a lot, which means they can actually capture different patterns among classes and won't predict all test samples only in the 'Information Giving' class.

Here we still choose Random Forest's multi-class confusion matrix 19 and the Information-Seeking task's binary classification metrics 20 as our example. All other models' metrics are similar and printed in the jupyter notebook.

From the multi-class confusion matrix, we can find that the Random Forest model can predict each class more accurately. As for the binary Information-Seeking v.s. Non-Information-Seeking classification task, its accuracy is improved to 0.815, precision is 0.78, recall is 0.815, and F1-score is 0.758.

7.4.2 Improved BERT Results

The performance of the Improved Bert model in classification is significantly better compared to the Baseline Bert model, as evidenced by the accuracy of the validation set converging to 0.74 and the accuracy of the testing set reaching 0.76. However, the model still struggles to identify "Information Seeking" texts and often classify them incorrectly into the "Information Giving" class. The loss curves for both the training and validation sets across 20 epochs, as well as the accuracy curve for the validation set are displayed in 13. The confusion matrix and the classification metrics (precision, recall, f1-score, and support) for each class are presented in 21 and 22.

8 Conclusion

Among all models we have built, Clinical Bert trained with augmented data gives the best result with an overall accuracy of 0.76 when classifying texts into all 5 categories. All other models have an accuracy of less than 0.6 when classifying texts into 4 categories (excluding SDM). Before generating more data, the baseline machine learning models have an accuracy ranging from 0.50 to 0.57. After introducing the data generated by GPT-4 and EDA, the performance has improved. E.g., the accuracy of XGBoost rose from 0.50 to 0.586. Similarly, the baseline Bert model has an overall accuracy of only 0.54, but the clinical Bert model trained with generated data has an accuracy of 0.76. Given the observations mentioned above, we concluded that class imbalance does compromise model performance, and addressing this issue can improve the overall accuracy of both machine learning models and BERT. The improved performance of our best model can also be attributed to the fact that Clinical Bert is better than traditional Bert in processing texts in the clinical context.

8.1 Discussion and Future Work

8.1.1 Sentence Level Clustering

By taking a closer look at the dataset itself, we found that some messages are coded into more than one category. E.g. "Hi That sounds reasonable I would do the colonoscopy more electively when no pressing issues are noted Best Vik" is coded as both "Emotion" and "Share Decision-Making". This may "confuse" the models and thus induce a great challenge for classifying messages. Besides, we have also found that some messages contain multiple sentences which should be coded into different categories E.g. in the following sentence "Dr Is there any chance I could be given a summary of the procedure and what was done without having to wait until March 24th as the wait is driving me nuts with the thought of what could be It would just greatly put my mind at ease and thank you for a great procedure as there was absolutely no pain or discomfort once I awoke in recovery": "any chance I could be given a summary of the procedure" should be coded as "Info Seeking"; "driving me nuts" and "put my mind at ease" should be coded as "Emotion"; "there was absolutely no pain or discomfort once I awoke in recovery" should be coded as "Info Giving". However, in the original dataset, this message is coded as "Info Seeking", "Emotion", and "Partnership". Simply putting this message in multiple categories doesn't help the classification. We think it may more be appropriate to classify the messages sentence-by-sentence and then assign weights to denote the importance of the different functions of each message. We realized this issue but had to address it in the future instead of this semester since the original data is unpunctuated, making it difficult to tokenize each message into sentences.

8.1.2 Number Replacement

Unlike Electronic Health Records, Patient-Provider Communication data tend to be messy, as there is no data validation when the data is produced. "Words" that contain non-alphabetic symbols, such as phone numbers, email addresses, zip codes, dates and times, and medication dosage can be written in various formats in the PPC data. As mentioned before in the feature extraction part, these "unique words" can break feature extraction. To address this issue, in the future, we can try to preprocess the data by matching these non-alphabetic terms using regular expressions and replace them with special tokens such as ¡phone number¿, ¡date¿, ¡time¿, etc. This could help improve the result of feature extraction because, after replacement, these words are not unique anymore and will not be identified as important features. We couldn't complete this task before the semester ends due to the difficulty of matching various formats. An alternative we are considering is to try different values of the parameter "mindf" for TfIdfVectorizer. This parameter sets a cutoff of document frequency (df) and excludes terms that appear in too few documents when calculating Td-Idf. We could try multiple values of mindf and get a better feature extraction result.

8.1.3 Further Exploration on Balancing the Data

There are other methods that help address data imbalance. For example, in the future we plan on trying resampling the data using SMOTE(Synthetic Minority Over-sampling Technique): oversample the minority class (partnership and emotional support), undersample the majority class (info-seeking and info-giving), or combine both methods to balance the class distribution. We can also try different EDA parameters: introduce different amount of noise by specifying the number of operations performed on each sentence; introduce different amount of balance by specifying the number of new sentences generated from each sentence.

8.1.4 BERT hyperparameter search

With more GPU access, we can conduct BERT hyperparameter search using trainer API. This can potentially help further improve model performance.

9 Code Availability

The source code have been deposited at `https://github.com/TheLittleJimmy/CBB750`. All scripts used to reproduce all the analyses are also available at the same repository.

10 Bibliography

References

[1] Emily Alsentzer et al. *Publicly Available Clinical BERT Embeddings*. 2019. arXiv: `1904.03323 [cs.CL]`.

[2] Jason Wei and Kai Zou. "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6383–6389. URL: `https://www.aclweb.org/anthology/D19-1670`.

[3] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165 (2020). arXiv: `2005.14165`. URL: `https://arxiv.org/abs/2005.14165`.

[4] Salvador Balkus and Donghui Yan. *Improving Short Text Classification With Augmented Data Using GPT-3*. 2022. arXiv: `2205.10981 [cs.CL]`.

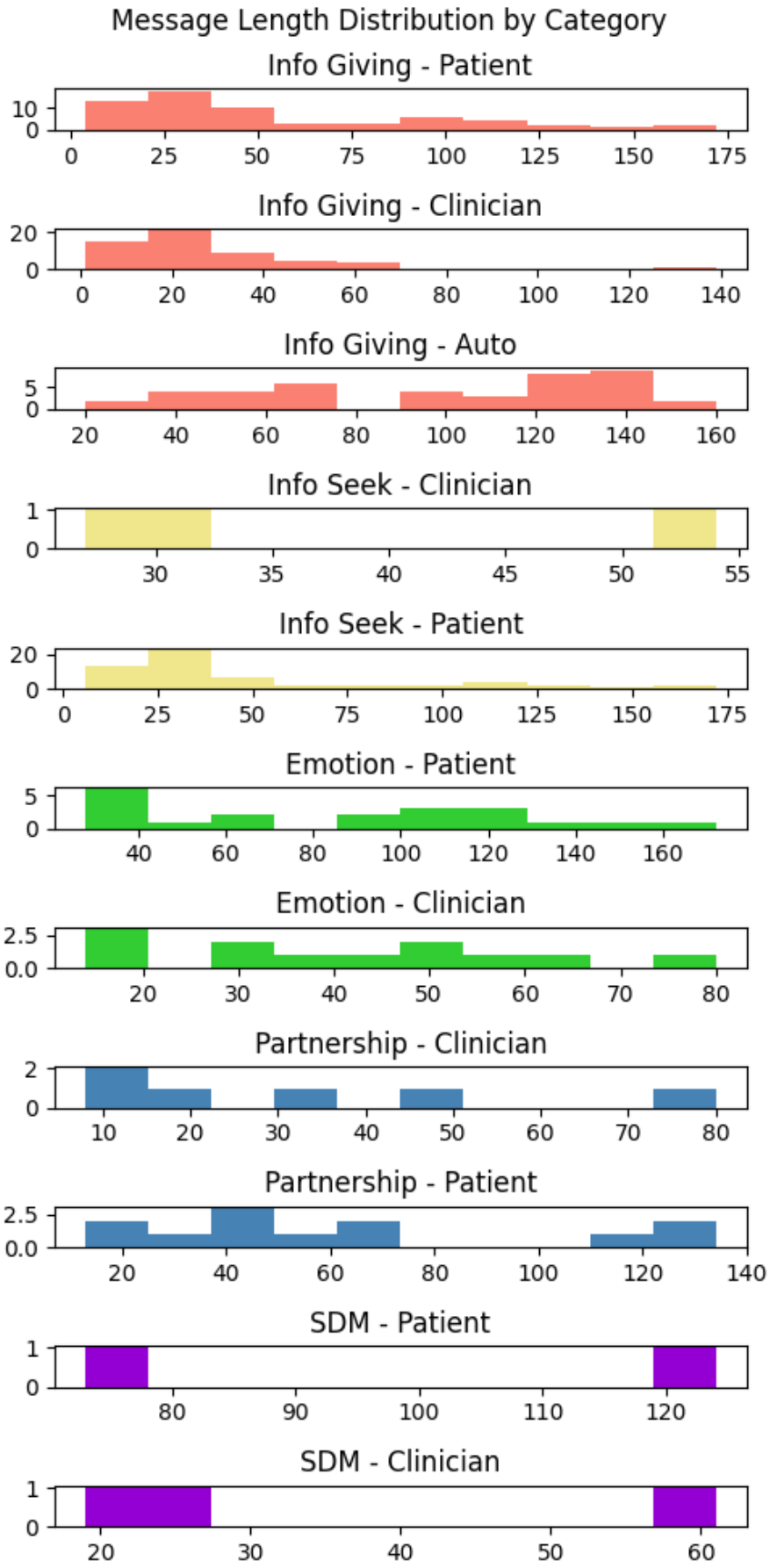


Figure 2: The histogram of message length for each category in the original data.

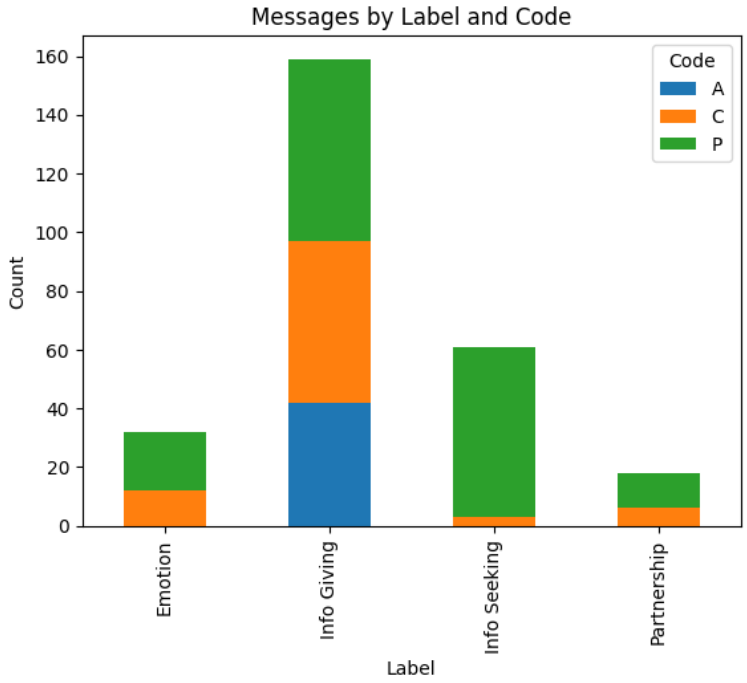


Figure 3: Message Count by Label and Sender Code

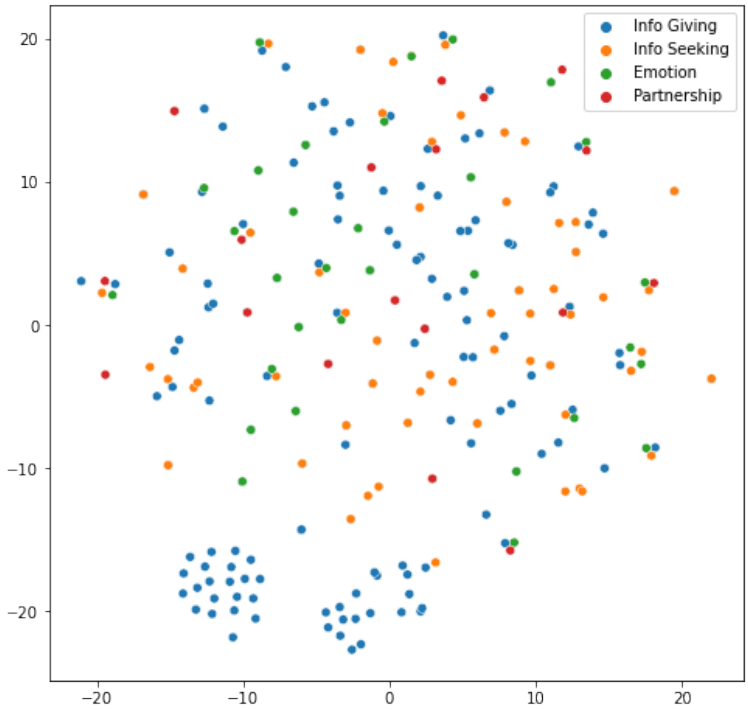


Figure 4: t-SNE plot of PPC dataset. The code annotations represent Patient (P), Clinician (C), Auto (A).

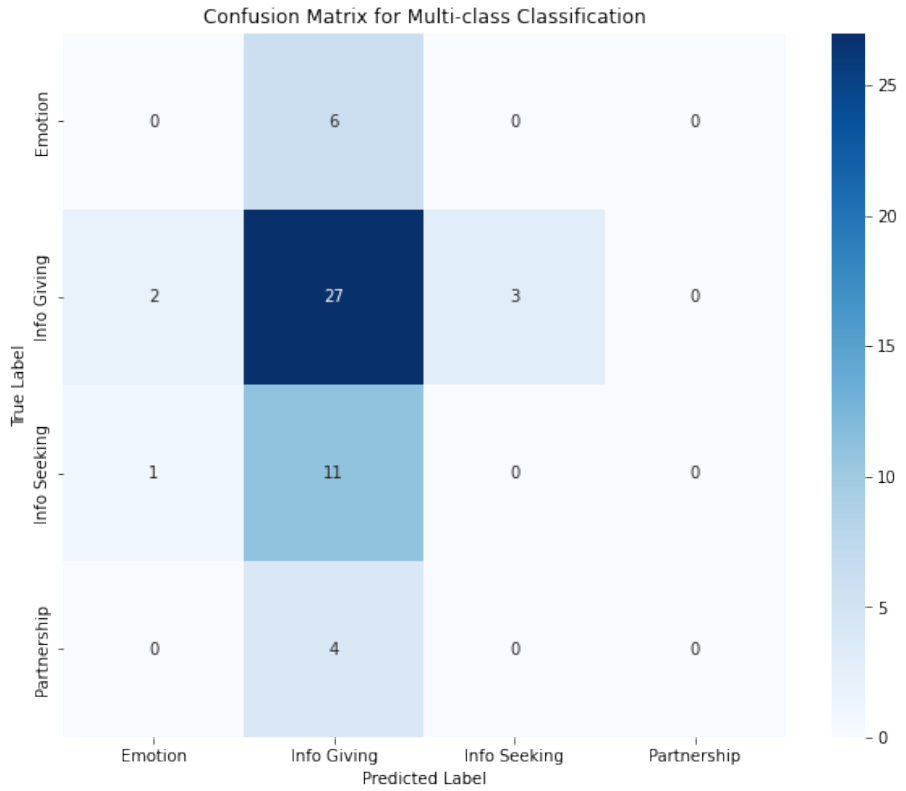


Figure 5: Confusion Matrix of Random Forest Classifier with Original Data

```
Top 200 features:
1 00
2 01032011
3 04
4 04042018
5 04262018
6 05172017
7 06272015
8 0630
9 06473
10 06477
11 06510
12 0670
13 06824
14 06830
15 10
16 100
17 1011
18 1095
19 10th
20 11
21 111
22 111417
23 11142017
24 1115
25 11th
```

Figure 6: Top features with the highest tf-idf values

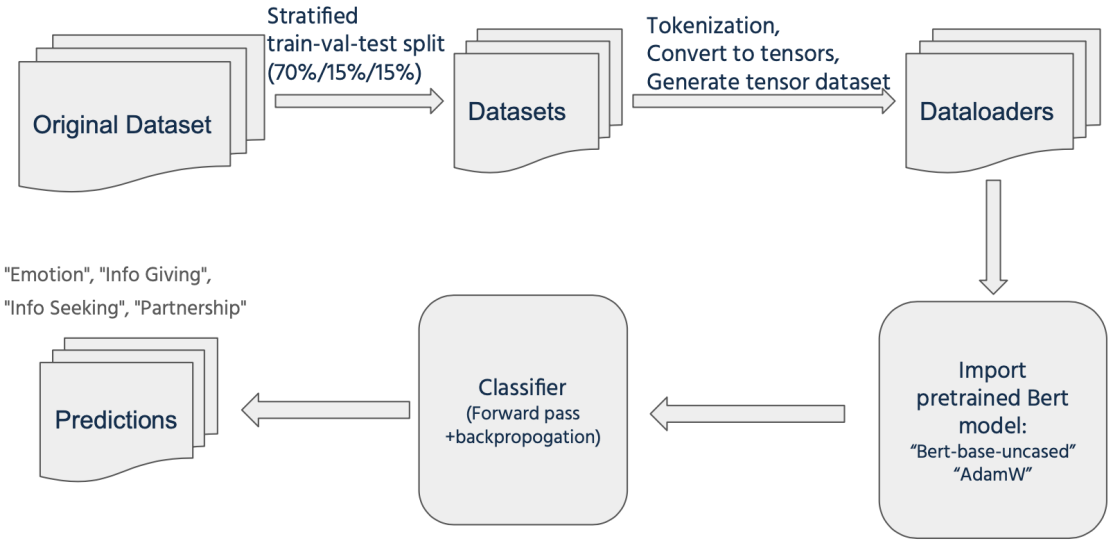


Figure 7: Pipeline for Baseline Bert Model

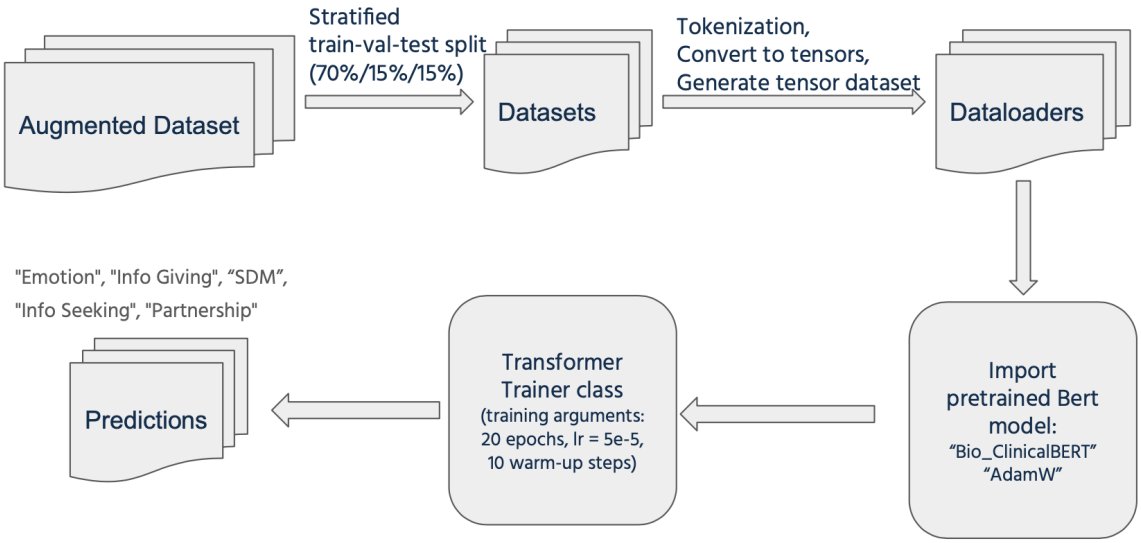


Figure 8: Pipeline for Improved Bert Model

```
Classification Report:
Accuracy: 0.593
Precision: 0.351
Recall: 0.593
F1-score: 0.441
Classification Report:
              precision    recall  f1-score   support

     Emotion       0.00      0.00      0.00         6
 Info Giving       0.59      1.00      0.74        32
Info Seeking       0.00      0.00      0.00        12
 Partnership       0.00      0.00      0.00         4

    accuracy                           0.59        54
   macro avg       0.15      0.25      0.19        54
weighted avg       0.35      0.59      0.44        54
```

Figure 9: Classification metrics of Logistic Regression

```
Classification Report:
Accuracy: 0.574
Precision: 0.456
Recall: 0.574
F1-score: 0.496
Classification Report:
               precision    recall  f1-score   support

     Emotion       0.00      0.00      0.00         6
 Info Giving       0.61      0.88      0.72        32
Info Seeking       0.43      0.25      0.32        12
 Partnership       0.00      0.00      0.00         4

    accuracy                           0.57        54
   macro avg       0.26      0.28      0.26        54
weighted avg       0.46      0.57      0.50        54
```

Figure 10: Classification metrics of Decision Tree

```
Classification Report:
Accuracy: 0.556
Precision: 0.432
Recall: 0.556
F1-score: 0.459
Classification Report:
               precision    recall  f1-score   support

     Emotion       0.00      0.00      0.00         6
 Info Giving       0.60      0.91      0.72        32
Info Seeking       0.33      0.08      0.13        12
 Partnership       0.00      0.00      0.00         4

    accuracy                           0.56        54
   macro avg       0.23      0.25      0.21        54
weighted avg       0.43      0.56      0.46        54
```

Figure 11: Classification metrics of Gradient Boost

```
Classification Report:
Accuracy: 0.500
Precision: 0.387
Recall: 0.500
F1-score: 0.426
Classification Report:
               precision    recall  f1-score   support

     Emotion       0.00      0.00      0.00         6
 Info Giving       0.58      0.81      0.68        32
Info Seeking       0.20      0.08      0.12        12
 Partnership       0.00      0.00      0.00         4

    accuracy                           0.50        54
   macro avg       0.19      0.22      0.20        54
weighted avg       0.39      0.50      0.43        54
```

Figure 12: Classification metrics of XGBoost

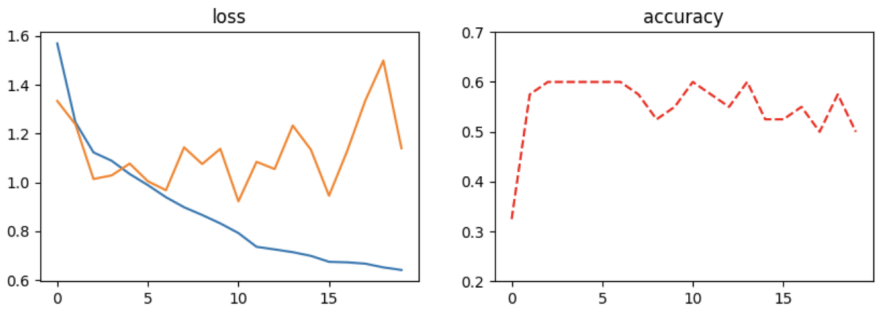
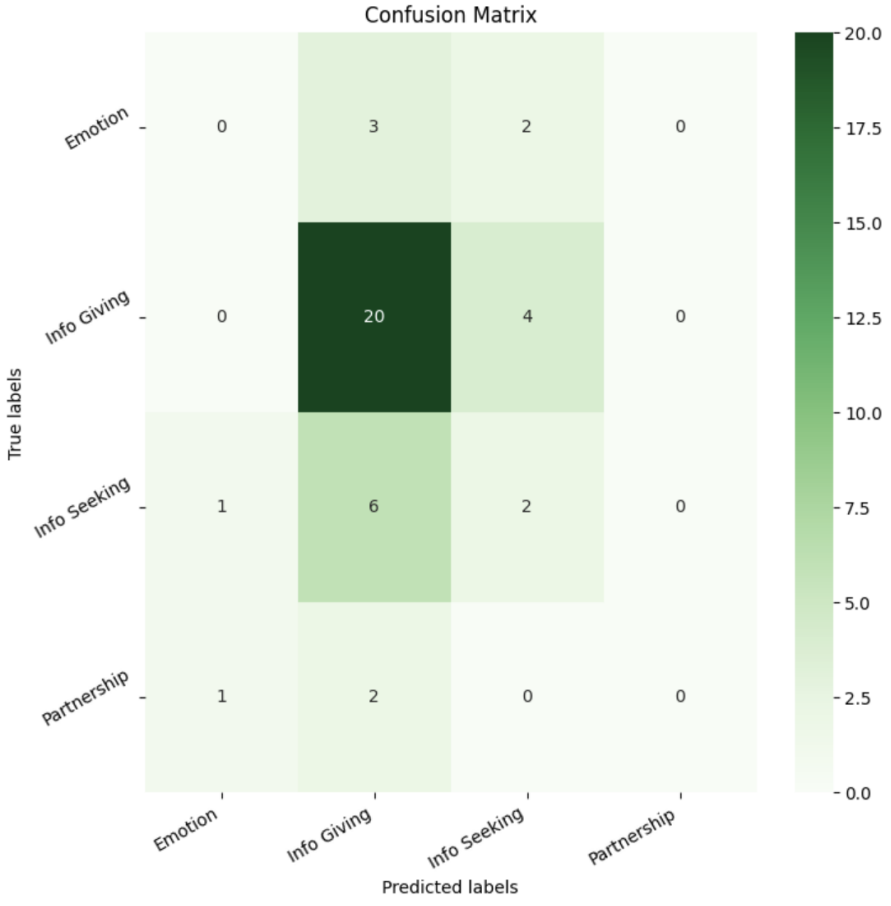


Figure 13: Loss and accuracy curves for Baseline Bert model
Left figure: loss curves for training set (Blue) and validation set (orange). Right figure: accuracy curve for the validation set.



	precision	recall	f1-score	support
Emotion	0.00	0.00	0.00	5
Info Giving	0.65	0.83	0.73	24
Info Seeking	0.25	0.22	0.24	9
Partnership	0.00	0.00	0.00	3

Figure 14: Confusion matrix and classification metrics for Baseline Bert model

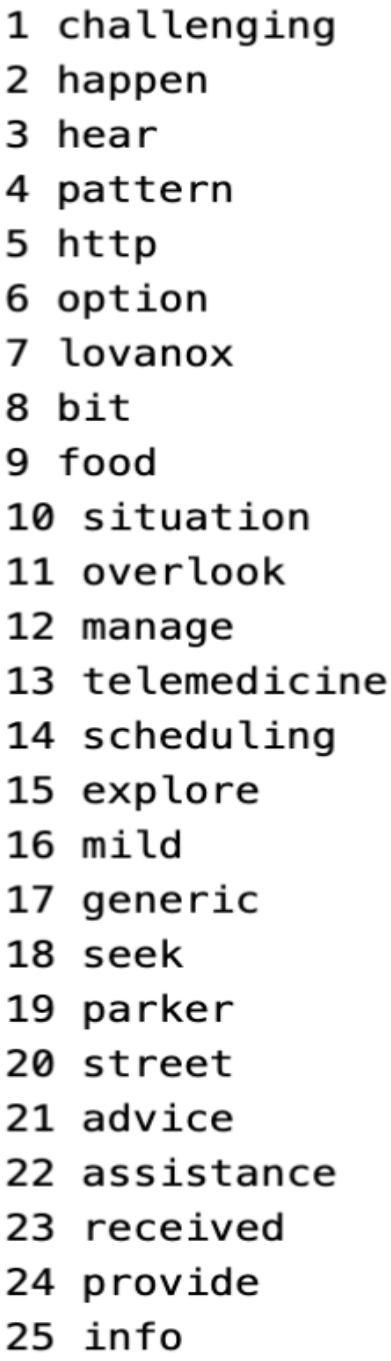
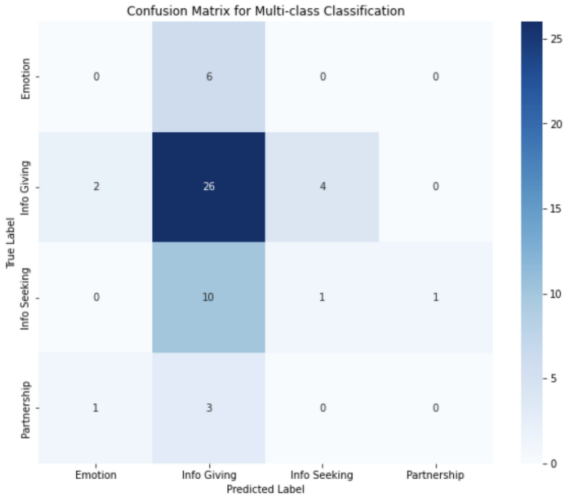


Figure 15: Top features with the highest tf-idf values after applying 4-fold binary classifications

Classification Report:
Accuracy: 0.500
Precision: 0.387
Recall: 0.500
F1-score: 0.426
Classification Report:

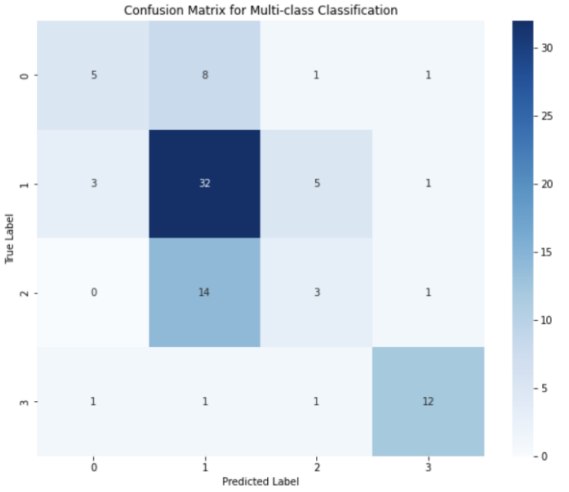
	precision	recall	f1-score	support
Emotion	0.00	0.00	0.00	6
Info Giving	0.58	0.81	0.68	32
Info Seeking	0.20	0.08	0.12	12
Partnership	0.00	0.00	0.00	4
accuracy			0.50	54
macro avg	0.19	0.22	0.20	54
weighted avg	0.39	0.50	0.43	54



XGBoost performance without GPT-generated texts

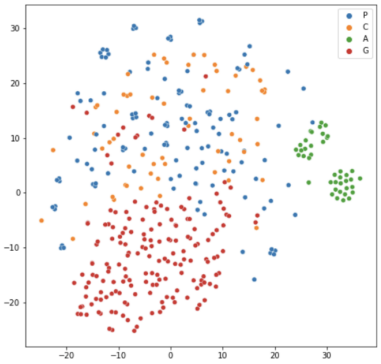
Classification Report:
Accuracy: 0.584
Precision: 0.557
Recall: 0.584
F1-score: 0.556
Classification Report:

	precision	recall	f1-score	support
Emotion	0.56	0.33	0.42	15
Info Giving	0.58	0.78	0.67	41
Info Seeking	0.30	0.17	0.21	18
Partnership	0.80	0.80	0.80	15
accuracy			0.58	89
macro avg	0.56	0.52	0.52	89
weighted avg	0.56	0.58	0.56	89

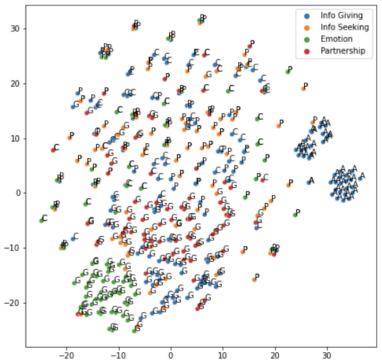


XGBoost performance with GPT-generated texts

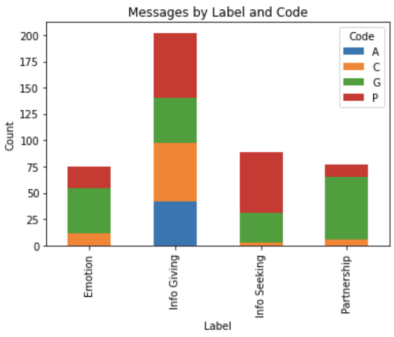
Figure 16: Performance and confusion matrix of:
Left: XGBoost model trained without GPT4-generated texts;
Right: XGBoost model trained with GPT4-generated texts



TSNE plot of message samples from Patient (P), Clinician (C), Auto (A), and GPT4-model (G)



TSNE plot of message samples in four categories. The letter annotation represents for the message source (Patient, Clinician, Auto, or GPT4-model)



Number of message samples in each category after adding the GPT-4 generated messages

Figure 17: GPT-augmented training set
Left: t-SNE plot of message samples from Patient (P), Clinician (C), Auto (A), and GPT4-model (G)
Middle: T-SNE plot of message samples in four categories. The letter annotation represents for the message source (Patient, Clinician, Auto, or GPT4-model)
Right: Number of message samples in the optimal augmented data by each category

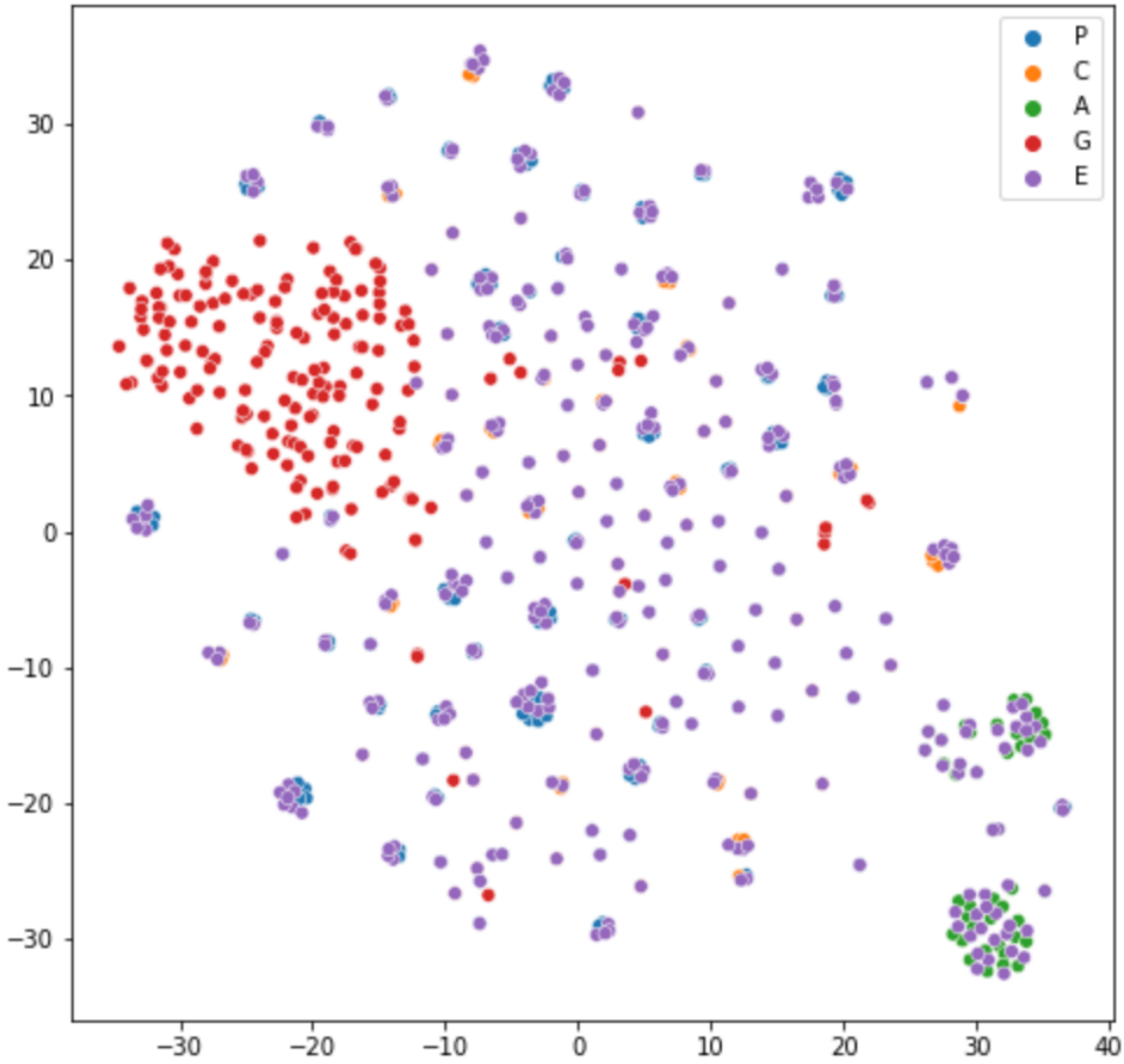


Figure 18: T-SNE plot of message samples in four categories. The letter annotation represents for the message source (Patient, Clinician, Auto, GPT4-model, or EDA

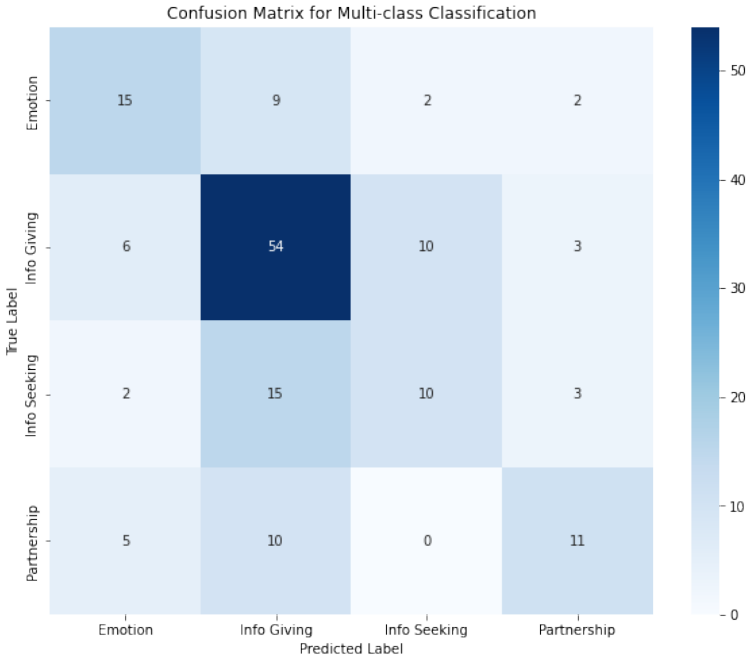


Figure 19: Confusion Matrix of Random Forest on the Augmented Data

```
Performing binary classification for task: Info-seeking vs. Non-info-seeking
Accuracy: 0.815
Precision: 0.780
Recall: 0.815
F1-score: 0.758
Classification Report:
                  precision    recall  f1-score   support

    Info Seeking       0.60      0.10      0.17        30
Non-info-seeking       0.82      0.98      0.90       127

        accuracy                           0.82       157
       macro avg       0.71      0.54      0.53       157
    weighted avg       0.78      0.82      0.76       157

Selected features: 12th, 1st, 494, 630, brand, cif, citrate, ctsim, delaying, dos, enema, fleet, food, forgot, happen, hormone, info, inqui
ring, iphone, jul, june, locally, lomotil, lovanox, luck, morning, overlook, perscription, port, procedure, provider, recover, refill, requ
est, reschedule, rescheduled, right, scan, scheduled, scripts, sept, september, spinal, surgery, uploaded, video, visit, vr, yale, zofram
```

Figure 20: Binary Classification metrics of Information-Seeking task on Random Forest

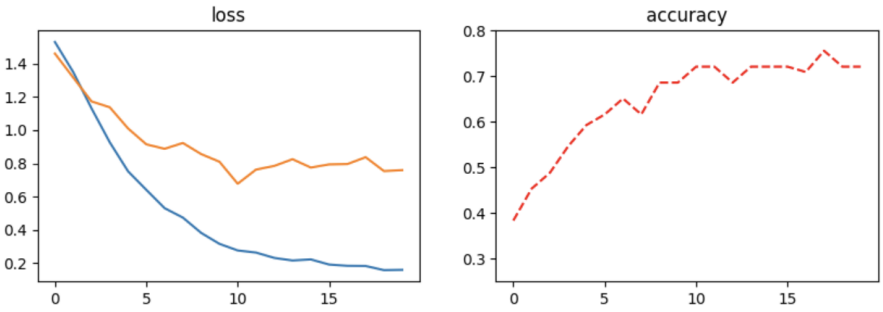
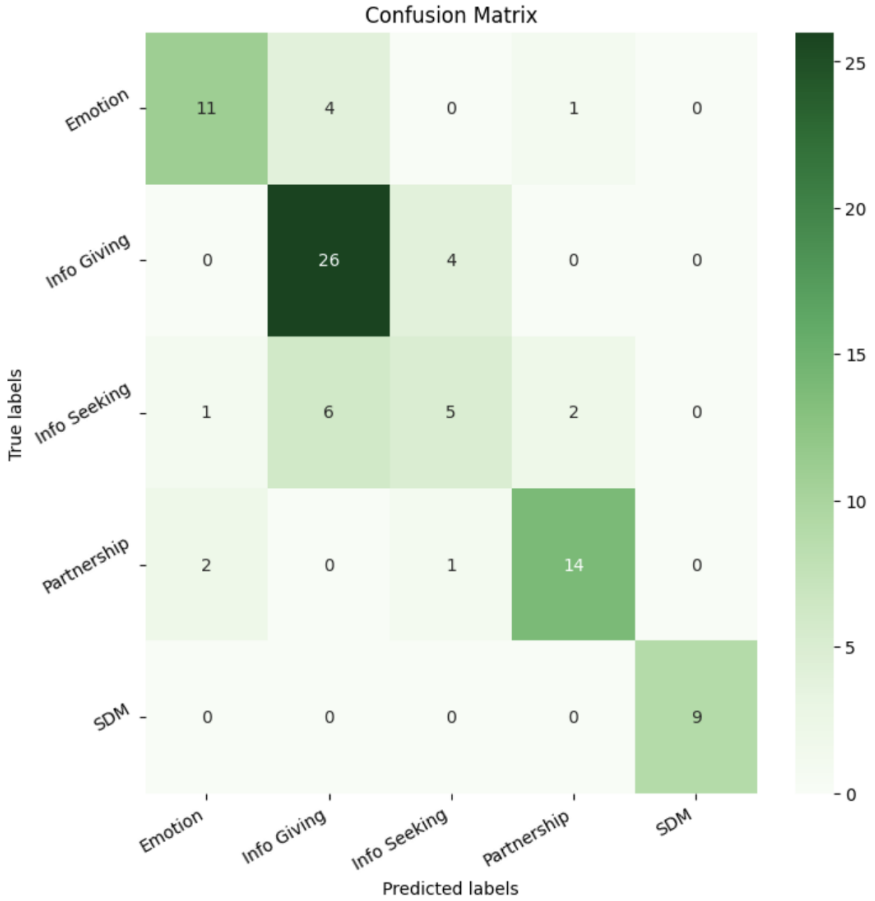


Figure 21: Loss and accuracy curves for Improved Bert model
Left figure: loss curves for training set (Blue) and validation set (orange). Right figure: accuracy curve for the validation set.
Left figure: loss curves for training set (Blue) and validation set (orange). Right figure: accuracy curve for the validation set.



	precision	recall	f1-score	support
Emotion	0.79	0.69	0.73	16
Info Giving	0.72	0.87	0.79	30
Info Seeking	0.50	0.36	0.42	14
Partnership	0.82	0.82	0.82	17
SDM	1.00	1.00	1.00	9

Figure 22: Confusion matrix and classification metrics for Improved Bert model