applied sciences

MDPI

Article

Research and Development of a Modern Deep Learning Model for Emotional Analysis Management of Text Data

Iryna Bashynska [1,*], Mykhailo Sarafanov [2] and Olga Manikaeva [2]

[1] Department of Enterprise Management, AGH University of Krakow, 30-059 Krakow, Poland
[2] Department of Information Systems, Odessa Polytechnic National University, 65044 Odesa, Ukraine; mi.study17@gmail.com (M.S.); manikaeva@gmail.com (O.M.)
* Correspondence: bashynska@agh.edu.pl

Abstract: There are many ways people express their reactions in the media. Text data is one of them, for example, comments, reviews, blog posts, messages, etc. Analysis of emotions expressed there is in high demand nowadays for various purposes. This research provides a method of performing sentiment analysis of text information using machine learning. The authors trained a classifier based on the BERT encoder, which recognizes emotions in text messages in English written in chat style. To handle raw chat-style messages, authors developed an enhanced text standardization layer. The list of emotions identified includes admiration, amusement, anger, annoyance, approval, caring, confusion, curiosity, desire, disappointment, disapproval, disgust, embarrassment, excitement, fear, gratitude, grief, joy, love, nervousness, optimism, pride, realization, relief, remorse, sadness, and surprise. The model solves the problem of multiclass multilabel text classification, which means that more than one class can be predicted from one piece of text. The authors trained the model on the GoEmotions dataset, which consists of 54,263 text comments from Reddit. The model reached a macro-averaged F1-Score of 0.50704 in emotions prediction and 0.7349 in sentiments prediction on the testing dataset. The presented model increased the quality of emotions prediction by 10.2% and sentiments prediction by 6.5% in comparison to the baseline approach.

Keywords: BERT; emotions prediction; General Language Understanding Evaluation (GLUE); GoEmotions; sentiment analysis

Citation: Bashynska, I.; Sarafanov, M.; Manikaeva, O. Research and Development of a Modern Deep Learning Model for Emotional Analysis Management of Text Data. *Appl. Sci.* **2024**, *14*, 1952. https://doi.org/10.3390/app14051952

Academic Editor: Douglas O'Shaughnessy

Received: 2 January 2024
Revised: 25 February 2024
Accepted: 26 February 2024
Published: 27 February 2024

1. Introduction

Supervised learning is widely used for solving various text classification problems. Any machine learning model consists of various mathematical operations, which means it cannot work with text data. Consequently, data have to be encoded into numeric sequences before any modeling starts. Various ML suites offer different toolkits and algorithms for this. For example, the TensorFlow library used in this study provides a TextVectorization layer that can be integrated into the model [1].

Classification problems are often solved by applying basic machine learning algorithms such as decision trees, K-nearest vectors, support vector machines, etc. These are easy to train, require relatively small resources, and perform well in cases with clear differentiation in data. However, with the development of data science, neural networks are actively gaining popularity. The reason for this is that deep learning neural networks have shown that they are capable of recognizing complicated patterns within the data. Models can track words with close semantics and predict text based on previously provided context.

However, how is it possible to check if the model can understand the text? One of the approaches to evaluating a machine learning model of human language perception is the General Language Understanding Evaluation (GLUE) benchmark: a collection of NLU tasks including question answering, sentiment analysis, and textual entailment, and an associated online platform for model evaluation, comparison, and analysis [2]. A few

models were evaluated and performed well on the GLUE benchmark [3–5]. One of them is BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [6]. The model itself does not solve any machine-learning problem, but per the authors of related work, "pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications".

This study aims to apply the BERT encoder for training a classifier for the emotional analysis of text information. Modeling was performed on the GoEmotions dataset [7]. The dataset consists of chat-style written comments in English extracted from the Reddit platform [8]. The main contributions of this study are the following:

- Approach to modeling on the GoEmotions dataset;
- Pre-process layer for chat-style text standardization;
- BERT-based classifier model for emotions prediction.

All code presented in this study is published in the GitHub repository: https://github.com/Shoomaher/sentiment-analysis (accessed on 25 February 2024). It contains modules developed and a Jupyter notebook that shows the process of training and testing the model. The authors used the Python (v.2.2.1) programming language and the following libraries:

- Pandas [9] for dataset investigation and manipulation;
- Numpy [10] for various array manipulations and calculations;
- Matplotlib [11] for data visualization;
- Unidecode [12] for fixing non-ASCII characters in data;
- TensorFlow [13,14] for building, training, and evaluating neural networks;
- Scikit-learn [15] to calculate the receiver operating characteristic (ROC) curve, calculate the area under the curve, and create a multilabel confusion matrix.

The next sections of this paper are organized as follows. Section 2 provides a discussion of related works. Section 3 describes the goal of the research paper and, based on it, defines the objectives of the research. In Section 4, a dataset investigation and preparation of text data for modeling is presented. Section 5 contains the modeling experiments description. Modeling results are provided in Section 6, as well as a comparison with the baseline model approach. Section 7 describes the limitations of this study. Finally, Section 8 consists of a conclusion and directions for further research.

2. Related Works

This section reviews three articles on the topic of emotion recognition from the text data.

The authors of the paper "GoEmotions: A Dataset of Fine-Grained Emotions" [7] proposed not only the dataset itself but also modeling results. The main purpose was to show the dataset's suitability for emotion recognition tasks. This was performed using the BERT fine-tuning approach and adding only a dense output layer. This was stated to be a strong baseline that leaves much room for improvement. Additionally, the authors trained a bidirectional LSTM model, which reached worse results than BERT on the GoEmotions dataset.

This study aims to achieve high annotation quality. However, the quality of text pieces, which is crucial for ensuring the model performs well with different inputs, is missing. This especially applies to the GoEmotions dataset, which consists of chat-style lexis. One of the most common and effective approaches for achieving better results in machine learning tasks is having clean data (or making them clean). The authors of this study will aim to achieve better results by focusing on data quality.

The research, "Exploring Transformers in Emotion Recognition: a comparison of BERT, DistillBERT, RoBERTa, XLNet, and ELECTRA" [16], also has to be mentioned. The authors presented a base comparison of the DistilBERT, ELECTRA, XLNet, and RoBERTa transformer models on the GoEmotions dataset. Additionally, the BERT model's results are

provided as a baseline. This study highlights that transformer models show state-of-the-art performance in different natural language understanding tasks.

The authors evaluated not only model performance but also computation costs. They stated that "we still have much room to improve models and create datasets for fine-grained emotions" [16]. Pre-processing was not applied in this work, while the base models were evaluated on a source dataset. Thus, similarly, the data quality question applies to this study.

In the paper "DialogXL: All-in-One XLNet for Multi-Party Conversation Emotion Recognition" [17], the authors present a pioneering approach to applying a pre-training transformer model XLNet for emotion recognition in conversation.

This article focuses on tracking the context and detecting different participating parties. One of its virtues is tracking intra- and inter-dependencies between participating parties. This work has a similar topic but different goals since our research is limited to classifying single pieces of text without considering context.

It needs to be addressed that the DialogXL model was evaluated on conversational datasets IEMOCAP, MELD, DailyDialog, and EmoryNLP, which are limited to a variety of emotions, including 5–6 emotion categories + neutral class.

In light of the above, this study aims to provide better emotion recognition results by creating a BERT-based model with enhanced data pre-processing.

3. Research Objectives

As mentioned before, this study aims to create an emotions classifier trained on the GoEmotions dataset with the use of a BERT encoder. Achieving good results in any machine learning task requires proper analysis of the data, as well as good quality data.

Thus, the authors outline the following key objectives for this research. First of all, to perform dataset label analysis and prepare it for training and testing neural networks. That includes splitting the dataset into the training, validation, and testing parts.

Secondly, examining the dataset messages and preparing clean data for modeling is necessary. This can be accomplished by preliminary cleansing of the dataset or by setting up some standardization inside of the model. In this study, authors follow the transfer learning approach and use the BERT pre-trained model. Consequently, a better quality of language understanding can be reached by giving the model more similar data to the pretrained one. This means that authors need to set up a data processing pipeline for language standardization.

Finally, the testing results need to be analyzed to understand if the model does not underfit or overfit and can classify and distinguish emotions from the text data. In addition to this, authors need to verify if the model can predict all emotions it was trained on. That objective is outlined due to the fact that the GoEmotions data is imbalanced.

4. Method

This section is divided into two parts. In the first part, the authors provide an overview of the data, key insights, and what was prepared for further modeling, mainly focusing on the dataset labels, e.g., the y part of the data. In the second part, they describe the approach for dividing the dataset into train, test, and split subsets. As for the third part, the authors review the investigation of texts, the X part of the data, and provide a text standardization layer to integrate into the model. The result of this section is a comprehensive set of tools for building and training ML models on the GoEmotions dataset.

4.1. Dataset Investigation

First of all, before any modeling, it is necessary to research what we are modeling. The authors chose the GoEmotions dataset, which contains a large collection of diverse text data covering a wide range of topics and emotions. This makes it ideal for training and testing emotional analysis models, as it provides a rich and varied source of data for researchers to work with [7]. This dataset contains text comments extracted from the Reddit platform [8].

Reddit is a social media platform for discussion, where people share their interests and run discussions on different topics, for example, mass culture, computer games, movies, gadgets, relationships, etc.

The GoEmotions dataset contains comments from 2005 (the start of Reddit) to January 2019, which were selected from subreddits with at least 10,000 comments [7]. The dataset includes only English comments with the use of chat-style language. It is important to mention that the dataset is impersonalized and contains [NAME] and [RELIGION] placeholders to reduce bias.

There are 54,263 comments included in the dataset, divided by the authors into three parts for modeling: training (43,410 elements), validation (5426), and testing (5427). As of now, all three parts are combined in a single dataset for review. The average length of text sequences is 68 characters, and the average number of words is 13.

The authors operate with emotions provided by the GoEmotions dataset. According to its paper, all comments were manually labeled with 27 emotion categories: admiration, amusement, anger, annoyance, approval, caring, confusion, curiosity, desire, disappointment, disapproval, disgust, embarrassment, excitement, fear, gratitude, grief, joy, love, nervousness, optimism, pride, realization, relief, remorse, sadness, surprise, and a neutral class.

The authors of the GoEmotions paper aim to "Provide the greatest coverage in terms of kinds of emotional expression". As we can see from the list of emotions present in the dataset, it mostly intersects with Robert Plutchik's emotion wheel, although the emotion categories "anticipation" and "trust" are not present. The correlation of emotions is out of the scope of this study. However, analysis of metrics reached on the testing dataset showed that the model sometimes confuses semantically close classes (Section 6.2).

The 27 emotion categories are divided into three sentiments: positive, negative, and ambiguous. While emotion is a psychological response of a writer of a text comment, sentiment shows a resultant feeling caused by this emotion. From the data standpoint, sentiment is a top-level group of emotions.

To summarize, this dataset provides a comprehensive list of emotions, which makes it appropriate for emotion classification.

Some of the texts are labeled with more than one category. The distribution of categories' counts per text is presented in Figure 1.

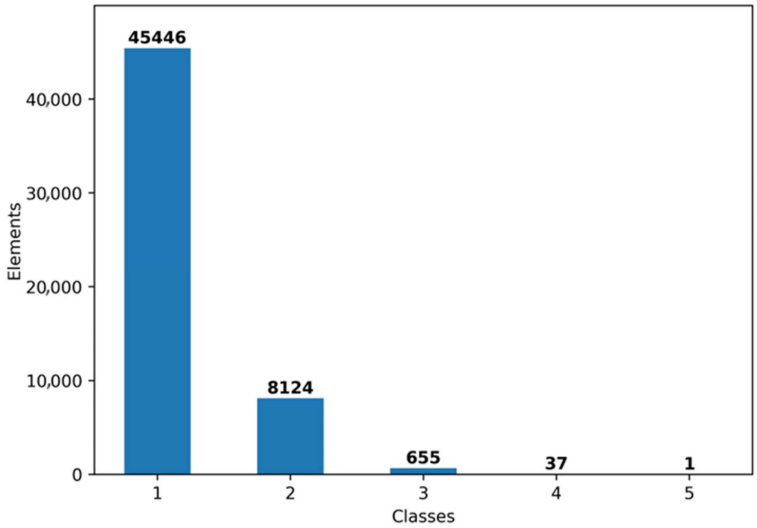

Figure 1. Distribution of quantities of categories per text message in the GoEmotions dataset (source: author's development).

As we can see, text comments are usually labeled with only one emotion. However, a significant value is added by texts labeled with two categories. Therefore, we need to solve not just the multiclass classification problem but also the multilabel classification,

meaning that the model should be able to predict more than one emotion category. The other approach is to exclude texts with two or more labels or select only one category for each of them. Consequently, the model will solve a multiclass classification problem by predicting only one correct class.

For the sake of clarity, we need to explain the difference between multiclass and multilabel classification problems. If a classification model determines if an element belongs to one of many groups, then the model solves the multiclass classification problem. The prediction will be a single class that fits best to this element. However, if an element can belong to several classes, then the model solves the multilabel classification problem. In this case, the result prediction will be a list of classes with zero, one, two, or more classes that fit an item.

It is common to express different emotions in a single message, for example, surprise and confusion: «Wow! I didn't expect this to happen. What should we do next?». This means that the second approach, which requires excluding texts with two or more emotion categories, would reduce model accuracy and usability. Consequently, the authors decided to follow the first approach, making a model that provides independent predictions of each emotion category and, thus, can predict more than one emotion for a text piece.

A common phenomenon for emotions analysis is emotions mixing. Mixed emotions are described as an expression of two or more emotions, usually the opposite ones. As mentioned before, a significant part of texts in the dataset is labeled with two or more emotions. Texts can be labeled with close emotions. Consequently, to verify if emotion mixing is presented in the dataset, we can select items labeled with emotions related to different sentiments. This results in 4200 texts selected. For example: "That was my first live Lions game (British and flights to DTW are expensive) I've never gone through so emotions in such a short amount of time" (excitement, realization, surprise), "How did they respond? You can't leave out the best part" (curiosity, disapproval).

It is also worth checking if the GoEmotions dataset contains texts labeled with opposite emotions according to their sentiments. That includes 981 elements. For instance: "I can watch test cricket, this would be no problems. Honestly though I'm annoyed I missed it, hopefully there's a re run" (annoyance, desire), "If he isn't respecting your clear boundaries then you should block him and move on with your life. He's not worth your time" (caring, disapproval).

Having such data supports the authors' decision to train the model to give independent predictions of each emotion category. However, the topic of emotions mixing is not further investigated in this study.

Taking into account that predictions are independent, we will remove the neutral class from the dataset, meaning that the text lacks any emotional characteristics and belongs to the neutral class if no emotions were triggered.

In Figure 2, we can see the class distribution in the dataset. As we can see, the dataset is imbalanced. For example, the "admiration" class consists of 5122 elements, the "approval" class has 3687, while the "grief" category contains only 96. As mentioned, the neutral class is to be removed and thus can be ignored in the current analysis.

Considering that we are solving a multiclass, multilabel classification problem, the authors need to encode labels accordingly. Consequently, the authors applied the strategy of "Problem transformation, whereby a multilabel problem is transformed into one or more single-label (i.e., binary or multiclass) problems" [18] and encoded labels using a multi-hot approach. Hence, 27 target variables were created, and each of them represents a binary value, whether the text is labeled with its emotion category or not.

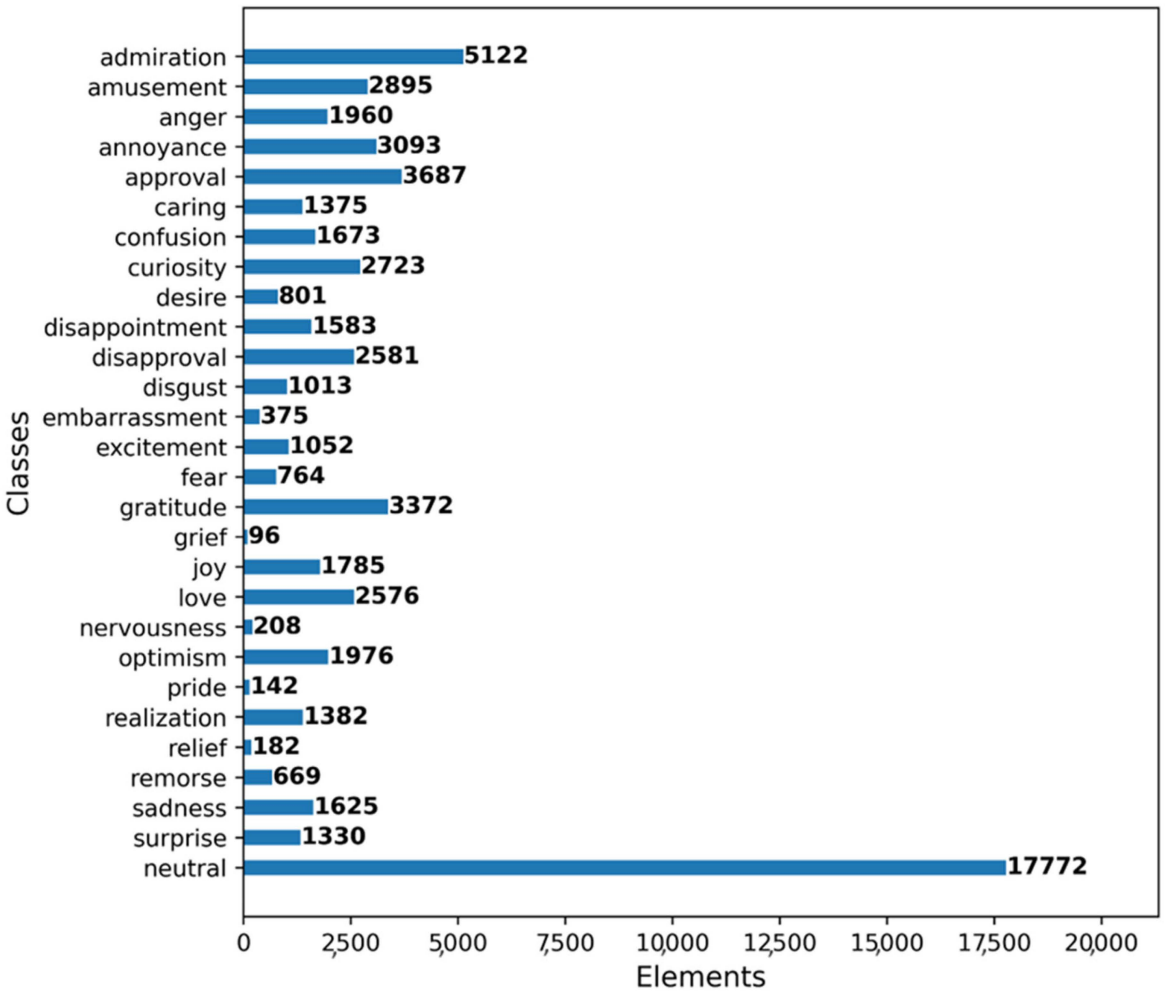

Figure 2. Distribution of classes in the GoEmotions dataset (source: author's development).

4.2. Train, Validation, Test Split

The process of training a neural network requires having three parts of data. The first and the biggest one is the train. It is the data the model sees and updates parameters by computing a loss function using backpropagation of error. This process is repeated several times, and each run is called an epoch. After each epoch, the model is evaluated by making predictions on the validation dataset and running metrics. This is necessary for tracking the model train process. Finally, when the training is completed, the model makes predictions on the test dataset, and its predictions are evaluated using metrics.

The dataset is already split into these three parts. However, the authors wanted to change the fraction of these parts during the experiments. Therefore, they decided to combine all three parts and implement a split. Since the dataset is imbalanced, the authors faced the issue that small classes could be unequally divided. Thus, they decided to implement the division of each category separately into train, validation, and test subsets.

In addition to this, the authors wanted to create a confusion matrix plot on the test dataset for all classes using the all-vs-all strategy. This can be achieved only by comparing one predicted label with only one true label. Consequently, the authors implemented a parameter for selecting only single-labeled texts in the test part.

During modeling experiments, it was noticed that models could not predict low classes, such as grief, relief, nervousness, etc. As a result, the authors added support for applying oversampling to the train data over the threshold supplied. For example, if the threshold value is 500, elements in classes of less than 500 are randomly repeated multiple times to reach 500 items. Finally, they added printing the random seed used for train, validation,

test split, and support for re-using it. As a result, datasets can be re-created. The train part is also randomly reshuffled after creation.

All code implementing this functionality can be found in the utils.make_dataframes function.

4.3. Text Standardization

Training a high-quality model requires operating with clean data. The better the data are pre-processed and standardized, the more accurate model predictions will be, especially considering that this study follows the transfer learning approach using the pre-trained BERT model. Authors of BERT used BooksCorpus and English Wikipedia datasets [6] for pre-training; thus, it is important to make text messages have a standard language form.

Text pre-processing is widely addressed in natural language processing and text classification studies. Each problem and data context requires an individual approach. A relevant example of data pre-processing in the healthcare domain is presented in the study "An Analysis on Large Language Models in Healthcare: A Case Study of BioBERT" [19]. Authors describe general data cleansing, as well as standardizing medical terms and applying custom tokenization "to accommodate the unique vocabulary and structure of biomedical and clinical texts. . . specialized tokenizers may be needed to handle medical terminology, abbreviations, and symbols" [19].

A similar problem is faced in the current research and presented later: the wide use of chat-style language with slang abbreviations, different word spelling, etc. This lexis appears as different tokens for the machine learning model. Hence, the model should be either initially trained on this data or adapted to processing such data. The first approach is not applicable since this study employs the transfer learning method, while the second one can be achieved by implementing custom tokenizers or additional pre-processing of the data.

Another relevant study is "CARER: Contextualized Affect Representations for Emotion Recognition" [20], in which authors perform emotion recognition on the dataset extracted from Twitter. Despite differences in audiences and texting styles, both Twitter and Reddit are discussion platforms on the Internet and present similar language.

Authors describe an identical problem of slang and coded words having the same meaning, such as "tnx" and "thanks" or "waaaaking me" instead of "waking me". That research provides its solution using "graph-based pattern representations" to extract emotion-relevant information.

This sub-section continues with the authors' research of the language used in the GoEmotions dataset and the implementation of the cleansing process step by step. This text pre-processing can be simmilarly applied to other datasets sourced from Internet discussion platforms, such as Reddit, Twitter, etc. However, additional processing and analysis might also be required, especially if texts are inclined towards some specific domain, as in the BioBERT study example mentioned before.

To begin with, the authors apply the unidecode module [12] to the text to remove non-ASCII characters. The main advantage of this tool is that it will not only remove any wrong characters, such as unreadable spaces and emojis but also replace them with correct interpretation if possible. For example, the "Latin small letter a with diaeresis" character (ä) becomes the "Latin small letter a", while the "single comma quotation mark" character becomes a usual "single quotation mark". This resolves any encoding issues, as well as removes different spellings of the same word.

The dataset is depersonalized and contains placeholders instead of name or religion references. For example, "I'm not talking about [RELIGION] anymore though. . .", "[NAME]. . . I'm sorry. This is just wrong. I, can't." These placeholders were removed from the text so that they do not affect the model and the model will be less biased.

The dataset contains text commentaries written in English using an informal chat-style language. To give it a standard look, the authors replace English contractions [21] with their full spelling where it is unambiguous. For instance, "can't" is simply replaced with

"can not". At the same time, "I'll" is left as is because it can either be "I will" or "I shall", and we can not exactly determine the correct form. Moreover, the form can affect the way the intention is expressed.

It is common for the chat style to stretch vowels and some consonant letters. For example, the following messages: "how'd you know? they're soooo good", "Loooool I didn't know that it's ridiculous". Regardless of the text data encoding algorithm, the words "soooooo" and "so" will have different encodings. To resolve this issue, in all cases where the letters 'a', 'e', 'i', 'o', 'u', 'y', 's', 'h', 'f', 'r', or 'm' are repeated more than three times, it is replaced with only one occurrence. The examples shown before become "how'd you know? they're so good" and "Lol I didn't know that it's ridiculous".

The next step taken was to replace abbreviations and chat words with their full phrases and meaning [22]. For instance, "ASAP" is replaced with "as soon as possible", while "L8R" becomes "later", and so on. In addition to this, it is common in chat style to use "r" instead of "are", "u" instead of "you", and "@" instead of "at". All of these are replaced with their full spelling.

What is more, all words with numbers and any punctuation characters except for a comma, dot, hyphen, apostrophe quotation, and exclamation marks are removed from the text. In addition to this punctuation fixing, multiple punctuation characters are replaced with only one of them. For example, "I love this!!! You got it!" becomes "I love this! You got it!". Moreover, punctuations are changed to a proper form, without spaces before and with only one space after.

Finally, leading and trailing spaces are trimmed, multiple spaces are replaced with only one, and text is converted to lowercase.

All of the processing above was packed into the standardized TextStandardizeLayer class that implements a TensorFlow [23] layer and can be integrated directly into the model before the encoder. This layer can also be used in other models working with chat language.

Although it comes as a ready-to-use solution, it is still necessary to investigate the data and set up text standardization according to the task. Additional functionality can be either added to the proposed layer or implemented as a separate layer before or after it.

5. Modeling

In the previous sections, the dataset was reviewed and split into the training, validation, and test parts. The text messages were standardized, and the target variables were encoded. Thus, we can proceed to build and train the model. This section is divided into two parts. In the first one, the authors propose a model structure, while in the second one, they review the training process.

5.1. BERT-Based Classifier Model

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a language model [6]. This is a masked model trained for language understanding and solving GLUE tasks [2]. Generally, the idea of masking was applied in the following way. A text corpus was initialized with random masks replacing word tokens. After that, the model was trained to predict the missing token based on its context. In addition, BERT was trained on the next sentence prediction task. One of the main advantages of BERT is its bidirectional design, meaning that the model is trained to analyze both left and right context, unlike only left-to-right analysis [6].

BERT is defined as a framework that consists of two parts: pre-training and fine-tuning. The process described before is the pre-training process for language understanding. The fine-tuning stage is presented as adding additional layers to the model and training on the specific task.

The authors followed the fine-tuning approach. The authors of BERT supply a family of pre-trained models of different sizes and on cased/uncased data. The model size is defined as a number of layers (L), hidden units (H), and attention heads (A). Considering that during the text standardization, all text was converted to lowercase, the authors used

an uncased model, which means that the model was pre-trained to ignore the case of words. Models are available to use via the TensorFlow HUB portal [24]; each model contains its own pre-process layer.

Although the BERT is recommended to be fine-tuned with only one additional layer, it is mentioned that additional layers can be applied according to specific tasks.

The authors propose the following structure of the classifier:

- Standardizer: text standardization layer described in Section 4.3;
- Pre-process: text pre-process layer added according to the selected BERT model;
- BERT encoder: BERT model selected;
- Dropout: random dropout of units;
- Dense layer: internal dense layer with ReLU activation function;
- Dropout: random dropout of units;
- Dense layer: output classifier layer with output units for each class. Considering that each output solves the binary classification problem, the Sigmoid activation function is applied.

The BERT model contains several outputs available for solving different tasks. In this study, the authors use pooled output [6]. The output shape is (batch size $\times$ BERT hidden units). As a result, each text sequence submitted is passed to output and encoded.

This architecture is defined in bert_model.BertSentimentModel class. The BERT model name is submitted as a parameter; the appropriate model will be used from the TensorFlow HUB.

In this study, the authors used a model from the small BERT family with 2 layers, 128 hidden units, and 2 attention heads, uncased. The reason for this is that the authors have limited computation resources since the training process was established on a laptop having the following configuration: Intel Core i5-7200U, 8 Gbs of RAM, and SSD drive (Intel, Santa Clara, CA, USA).

As for the other parameters, both dropout rates were set to 0.4 (randomly drop 40% of units), internal dense units were set to 256, and the output dense units were set to 27.

5.2. Train Process

In this sub-section, the authors review the training process, the parameters applied, and the metrics used.

First, the dataset was split into the train, validation, and test parts with a fraction of 85%. In addition, the authors applied oversampling of low classes with a threshold value set to 500 and set up a test dataset to include only single-labeled elements. Classes distribution for each part can be found in Appendix A. The dataset was batched by 64 elements in a batch. Considering that the target variables are encoded as binary values, the loss function was set to binary cross entropy [23].

The authors decided to use the AdamW optimizer because in the context of emotional analysis of text data, the use of the AdamW optimizer can be particularly useful as it helps to prevent overfitting, which is a common problem in natural language processing tasks. By using this optimizer, the model is better able to generalize to new, unseen data, improving its overall performance and making it more useful for real-world applications [25]. This is the optimizer BERT was initially trained on, and it performs regularization by weight decay. The authors applied a similar training schedule as BERT pre-training: linear decay of a notional initial learning rate, prefixed with a linear warmup phase over the first 10% of training steps [6]. The training was performed up to 10 epochs. As a result, each epoch consisted of 463 steps, adding to 4630 steps in total with 463 warmup steps. The initial learning rate was set to 3×10^{-4}. Decreasing the learning rate should perform better [26]; however, it would also require additional computation time or additional resources. Furthermore, the learning rate value has a higher impact on the model performance than the batch size [26].

During the model training and testing, the authors tracked the precision and recall for each predicted class individually and for all classes.

The indicator precision describes how close the measurement values are to each other. Precision is the number of true positives (TP) over the number of true positives (TP) plus the number of false positives (FP).

$$Precision = \frac{TP}{(TP + FP)}$$

TP—True Positive, FP—False Positive

Recall defines how accurately our model is able to identify the relevant data. Recall is the number of true positives (TP) over the number of true positives (TP) plus the number of false negatives (FN)

$$Recall = \frac{TP}{(TP + FN)}$$

FN—False Negative

Additionally, the authors tracked the F1-Score of all classes with weighted, micro, and macro averaging. A threshold value for all metrics was set to 0.5. The F1-Score is the Harmonic mean of the Precision and Recall.

$$F1 = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)}$$

Finally, the authors initialized the training early stopping to stop the training process in case validation loss does not decrease during the last two epochs. Thus, the model was trained as long as it increased its performance. During experiments, training was stopped after 7 epochs.

Figure 3 shows how precision, recall, and F1-Score metrics, as well as the value of the loss function, changed over epochs on the train data during the model fit.

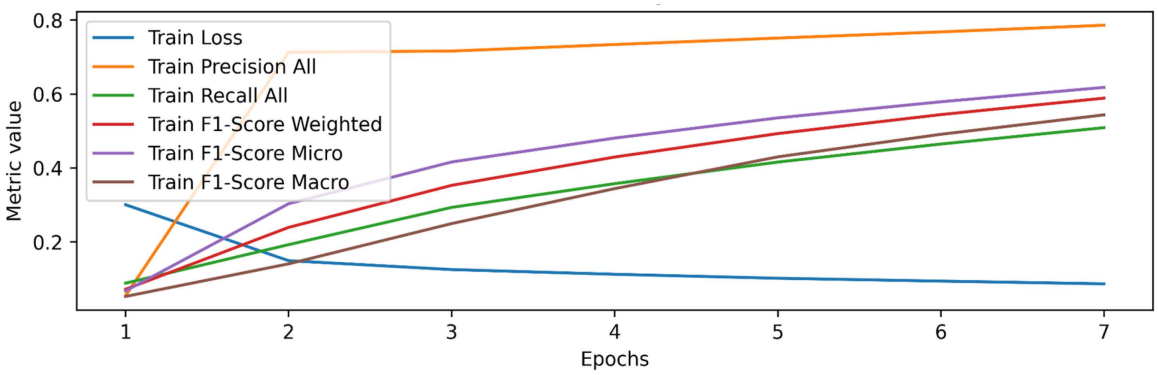

Figure 3. Change in values of the loss function, Precision, Recall, and F1-Score with micro, macro, and weighted averaging for all classes on the training dataset over train epochs (source: author's development).

Similarly, changes in the values of the same metrics and the loss function on the validation dataset are presented in Figure 4.

As we can see, the value of the loss function decreased while all of the F1-Score metrics increased with each epoch. Thereafter, the model gained performance and did not tend to overfit.

Finally, the authors ran the same metrics on the test dataset, which consists of the data that the model has not seen before, and analyzed the results. Having conducted a series of experiments, the authors present the best results in Section 6 of this study.

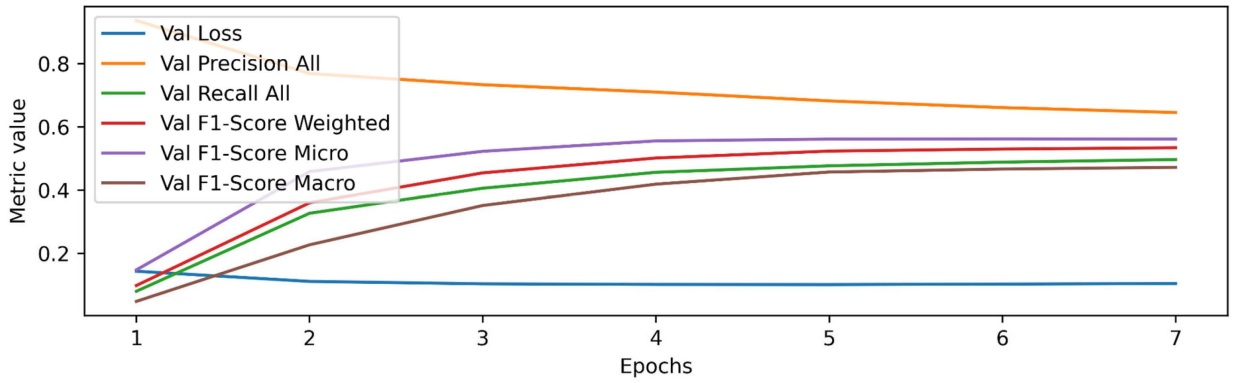

Figure 4. Change in values of the loss function, Precision, Recall, and F1-Score with micro, macro, and weighted averaging for all classes on the validation dataset over train epochs (source: author's development).

6. Results

This section is divided into two parts. In the first sub-section, the authors present the metrics the model has reached on the testing dataset, while in the second one, the authors provide the analysis of the result and ideas for further research. Additionally, in the third sub-section authors provide a comparison with the baseline approach. Finally, sub-section four contains the practical implications of the study.

6.1. Metrics

Table 1 presents the micro-, macro-, and weight-averaged F1-Score for all classes. Additionally, the authors calculated the micro- and macro-averaged F1-Score after converting true and predicted emotions labels to sentiment labels according to the map supplied with the dataset (Table 2). Table 3 shows precision and recall for each emotion category class, as well as for all classes. All of the metrics were reached on the testing dataset, and all of them were set up to have an activation threshold of 0.5.

Table 1. Precision, Recall, and F1-Score weight-, micro- and macro-averaged metrics for all classes reached on the testing dataset (source: author's development).

Precision	Recall	F1-Score (Weighted)	F1-Score (Micro)	F1-Score (Macro)
0.61322	0.55658	0.56341	0.58353	0.50704

Table 2. Precision, Recall, and F1-Score weight-, micro- and macro-averaged metrics for all classes reached on the testing dataset (after converting) (source: author's development).

F1-Score (Micro)	F1-Score (Macro)
0.7760	0.7349

In Figure 5 we can see plots for micro and macro-averaged ROC curve. The area under the micro-averaged curve is 0.9361, while the macro-averaged one is 0.9136. In addition to this, Appendix B contains ROC curves with the area under the curve calculated for each of the emotion classes.

Table 3. Precision and Recall metrics for each emotion category reached on the testing dataset (source: author's development).

Emotion	Precision	Recall
admiration	0.68873	0.73473
amusement	0.76567	0.88365
anger	0.47297	0.52239
annoyance	0.31169	0.32432
approval	0.49838	0.39386
caring	0.51351	0.42222
confusion	0.43307	0.31609
curiosity	0.55866	0.71174
desire	0.75676	0.35443
disappointment	0.31034	0.0625
disapproval	0.52717	0.33333
disgust	0.81395	0.35354
embarrassment	0.72	0.46154
excitement	0.7619	0.16495
fear	0.71186	0.48276
gratitude	0.91268	0.90251
grief	0.33333	0.625
joy	0.63448	0.55758
love	0.71765	0.9037
nervousness	0.33333	0.35294
optimism	0.57065	0.61404
pride	0.53333	0.72727
realization	0.63889	0.18254
relief	0.16667	0.17647
remorse	0.64935	0.75758
sadness	0.56	0.5283
surprise	0.60976	0.53191

One more approach to evaluate the classifier's performance is calculating a confusion matrix. Appendix C contains confusion matrixes for each emotion category class.

Finally, the authors have added the confusion matrix for all classes using the all-vs-all strategy in Appendix D. It is necessary to outline the process of its calculation. This matrix can be created only for single-labeled multiclass classifiers, as it requires a comparison of only one predicted class with only one true class for each element. To create this matrix, the authors have set up the testing dataset to include only single-labeled texts, while the predicted class was always selected with the maximum prediction value.

6.2. Discussion

First of all, the model shows non-zero precision and recall for all emotion categories, as we can see in Table 2 data. As mentioned before, the dataset is imbalanced, and some of the classes used have 0 precision and recall, meaning that the amount of data was not enough to fit the model. This issue was resolved by applying oversampling for low emotional categories on the training dataset (Section 4.2).

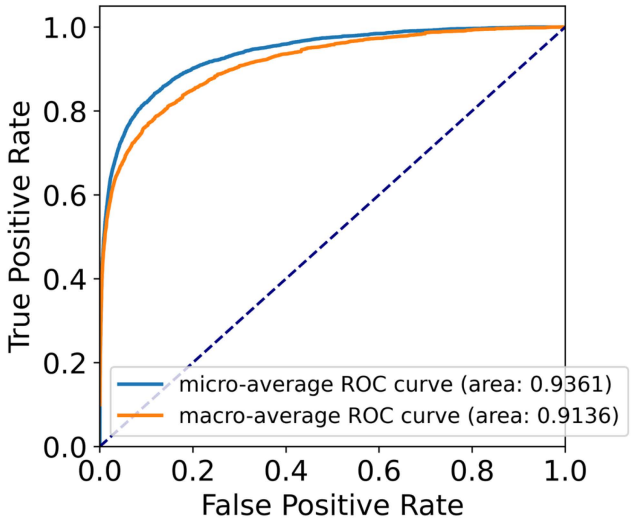

Figure 5. Micro- and macro-averaged ROC Curve (source: author's development).

The main metrics to compare models during the experiments were micro- and macro-averaged F1-Scores, giving priority to the macro-averaged score, as it works well on imbalanced data and treats all classes equally.

The all-vs-all confusion matrix, which is shown in Appendix D, represents a clear diagonal of correct predictions. This supports the previous conclusion that the model can distinguish and predict all emotion categories.

It is worth mentioning that the model sometimes confuses semantically close classes. As we can see, anger and annoyance, curiosity and confusion have significant confusion. This can be seen in the all-vs-all confusion matrix. The reason for this is that similar language can be used for expressing these emotions. For instance, aggressive words can be frequently used in the expression of both anger and annoyance, while question forms should be common for curiosity and confusion emotions.

As for the ability of the model to distinguish positive, negative, and ambiguous sentiments, higher F1-Scores (micro- and macro-averaged) show that it was achieved. This also supports the idea that even though sometimes it can be hard for the model to differ one emotion from another if they have close semantics, the model can determine whether positive, negative, or ambiguous emotions are expressed.

The achieved results give much room for further research by applying the emotion classification model. Apart from the exact data science, the model can also be used for various interdisciplinary studies. These can be in any area that involves human behavior. For instance, conducting marketing analysis and sociology research might require an examination of the emotions and sentiments of the target audience. The model can predict emotions expressed in text questionnaires and provide a source for the data mining process.

The quality of prediction significantly depends on the amount of data for this emotion category. This is a common rule for the machine learning industry and is supported by this study. As we can see from the all-vs-all confusion matrix presented in Appendix D, the classifier predicted more accurately classes with a higher amount of data: admiration, amusement, gratitude, and love. At the same time, it made less accurate predictions for embarrassment, relief, excitement, and remorse, which have comparably fewer labels.

It is also important to mention that a lower number of elements in class not only provides the model with less data to train on but also results in less data to test the model. In Appendix B, we can see ROC curves for each emotion category class. Curves for classes with a low qty of test data, such as grief, relief, pride, and nervousness, demonstrate explicit sleepiness. The test dataset class distribution can be seen in the third plot of Appendix A.

The original BERT study [6] reports that increasing the model size results in increasing its accuracy in solving GLUE tasks, which means that a more complex model with a higher amount of layers, hidden units, and attention heads shows better language under-

standing. Thus, training a classifier described in this study with a more complex BERT model, for instance, BERT-large, can lead to better emotion prediction. In addition to this, according to an article [26], a lower learning rate also results in better performance. However, it is important to mention that proof of both of these concepts requires more computational resources.

Finally, having more data for training and testing the model also should increase the accuracy of the classifier. It can be achieved by either adding more data to the current dataset or using another dataset.

6.3. Comparison with a Baseline Approach

Authors of the GoEmotions dataset paper validated if the dataset is suitable for modeling purposes in the "6. Transfer Learning Experiments section" [7]. They presented results of baseline fine-tuning of the pre-trained BERT-base model (12 layers and 768 hidden units). The baseline model showed a macro-averaged score of 0.46 in emotions prediction and 0.69 in sentiments prediction.

The authors of this study describe a more complex model with better text pre-processing and dataset preparation. The authors reached a macro-averaged F1-Score of 0.50704 in emotions prediction and 0.7349 in sentiments prediction. Thus, the authors increased the quality of emotions prediction by 10.2% and sentiments prediction by 6.5%.

It is important to mention that, in this study, the dataset has a different train–test split. Consequently, this is not just the classification model that performs better but the method in general that achieves higher metrics on unseen data. The main improvement of the existing method is having an additional text standardization layer and different model configurations, which leads to achieving higher metrics on the testing dataset.

For direct comparison, the model was trained and validated on the original GoEmotions dataset without changes to splits. The neutral class is used "as is", although, in the writers' opinion, it should be excluded. Table 4 shows metrics reached in emotions prediction, while Table 5 presents sentiments prediction results.

Table 4. F1-Score micro- and macro-averaged metrics for all classes reached on the testing dataset on original GoEmotions data in emotions prediction (source: author's development).

F1-Score (Micro)	F1-Score (Macro)
0.4684	0.3442

Table 5. F1-Score micro- and macro-averaged metrics for all classes reached on the testing dataset on original GoEmotions data in sentiments prediction (source: author's development).

F1-Score (Micro)	F1-Score (Macro)
0.6711	0.6367

The model shows zero precision and recall metrics for classes "embarrassment", "grief", "nervousness", "pride", and "relief".

These scores are lower than those presented in the original GoEmotions study. This can be explained by using only a tiny-BERT model (2 layers, 128 hidden units) due to computing resource limitations. Employing a more complex BERT model, for instance, BERT-base, should positively impact the classification quality. As mentioned before, the GoEmotions research refers to the BERT-base model as a baseline.

Another factor to consider is different splits. Re-splitting the dataset with fraction by class, as well as oversampling low classes, helps the model detect and distinguish all emotion categories.

Finally, excluding the neutral class leads to increasing the model's performance. The GoEmotions research describes it as: "If raters were not certain about any emotion being expressed, they were asked to select Neutral. We included a checkbox for raters to indicate

if an example was particularly difficult to label, in which case they could select no emotions. We removed all examples for which no emotion was selected" [7]. We question this approach as the emotions are either expressed in a piece of text or not. In case no emotions are expressed, the model should not trigger any emotion categories by threshold. Apart from that, the neutral class, being the largest one, causes model overfitting toward itself. Thus, the neutral class is excluded from the current study.

For supplementary comparison, the approach proposed in this study was limited to training the vanilla tiny-BERT model (2 layers, 128 hidden units) without text pre-processing and additional layers on the presented splits. Although hyper-parameter tuning might be required, these were used as presented before (Section 5.2 Train Process). The neutral class was also excluded. Table 6 shows metrics reached in emotions prediction, while Table 7 presents sentiment prediction results.

Table 6. F1-Score micro- and macro-averaged metrics for all classes reached on the testing dataset for limited model in emotions prediction (source: author's development).

F1-Score (Micro)	F1-Score (Macro)
0.5648	0.4909

Table 7. F1-Score micro- and macro-averaged metrics for all classes reached on the testing dataset for limited model in sentiments prediction (source: author's development).

F1-Score (Micro)	F1-Score (Macro)
0.7423	0.7060

As we can see, these results are slightly lower than presented in Section 6.1 Metrics. This demonstrates that text pre-processing and additional model layers improve the model's performance by 3.27% in emotions predictions and 4.1% in sentiments prediction, according to F1-Score macro-averaged metrics. The main contribution is achieved by imbalance treatment and exclusion of neutral class.

To additionally verify this, the proposed model (Section 5.1) was trained on original GoEmotions splits, excluding the neutral class. Table 8 shows metrics reached in emotions prediction, while Table 9 presents sentiments prediction results.

Table 8. F1-Score micro- and macro-averaged metrics for all classes reached on the testing dataset on original GoEmotions data in emotions prediction excluding neutral class (source: author's development).

F1-Score (Micro)	F1-Score (Macro)
0.5677	0.4609

Table 9. F1-Score micro- and macro-averaged metrics for all classes reached on the testing dataset on original GoEmotions data in sentiments prediction excluding neutral class (source: author's development).

F1-Score (Micro)	F1-Score (Macro)
0.8144	0.7604

As we can see, the macro-averaged F1-Score is lower by 9% than when using the proposed splits. Although the model performs better in sentiment prediction, the all-vs-all confusion matrix (Appendix E) shows that the model tends to predict "sadness" or "fear" in a true "grief" class. All of these classes belong to a negative sentiment. It is important to mention that the all-vs-all confusion matrix may not be completely representative in

this case because it considers the predicted class as the one with the highest prediction score, while the testing part of the original GoEmotions split includes items with two or more labels.

Moreover, the model showed zero precision and recall metrics for the classes "grief", "nervousness", and "relief", which means it cannot distinguish them.

This supports the idea that imbalanced datasets, such as GoEmotions, require special misproportion treatment on split for achieving good model performance. Despite different split algorithms, the model's performance is always evaluated on a test dataset, which presents "unseen" data with a meaningful fraction.

In light of the above, dataset preparation with custom training/validation/testing split, exclusion of neutral class, enhanced text standardization, and a more complex classifier structure leads to better emotions and sentiment classification quality.

6.4. Practical Implications

The authors present an emotions classifier that shows a higher quality of emotions and sentiment prediction from the text information. The ready-to-use model can process raw text messages in English written in a chat-style language without any additional preprocessing. This model can be used for sentiment analysis in a batch job, as well as a part of an information system.

Additionally, all modules developed by the authors can be used for further modeling and future improvement by running experiments using different BERT models and parameter fine-tuning to increase classification quality. All code presented in this study is published in the GitHub repository: https://github.com/Shoomaher/sentiment-analysis (accessed on 25 February 2024).

7. Limitations

There is a list of limitations of the current study that needs to be addressed. First of all, the main limitation is the data used for training the model. The GoEmotions dataset consists of texts written in English, which means that the model is not able to predict texts written in any other language. This also applies to the style of writing. Although the authors added a text standardization layer, completely different types of writing will lead to questionable results. For example, applying the model to a big blog post instead of a small comment. It is also worth mentioning that the model operates only with text data. This study does not involve speech or facial emotions analysis.

Secondly, the authors of the GoEmotions paper outline that the Reddit platform, which is a source of text comments, is biased towards young male users. This may cause the model to perform worse on texts written by completely different audiences.

Thirdly, as mentioned before (Section 4.1), the dataset is imbalanced, which means that the model is less accurate in predicting classes with a low amount of data.

One more limitation is the list of emotions and sentiments that the model can predict. Obviously, it is strictly defined by the dataset used for training the model. Predicting different sets of emotion categories requires training the model on a different dataset and additional analysis.

Finally, the authors were limited in computational resources and data for training and evaluating the model. The trend nowadays is toward large language models. These models serve multi-purposely and are likely to provide better emotion classification results. The main reason for this is the vast amount of computation resources and training data used for fitting such models.

8. Conclusions

This paper researches a method for emotional analysis of text data using deep learning. Modern deep learning models can detect complicated patterns within the data. This is proved by the GLUE [2] benchmark, which evaluates the model's ability for human language understanding.

In this study, the authors trained and analyzed the emotions classifier using the pre-trained BERT model [7] fine-tuning approach. The modeling was performed on the GoEmotions dataset [6]. In addition to this, the authors described a text-standardization layer for performing data cleansing inside of the model so that it can handle raw text data. The proposed text-standardization layer can be used in other models working with text data in English written in chat style.

According to the results achieved aiming to the F1-Score metric with macro-averaging, the model shows good performance. The presented model increased the quality of emotions prediction by 10.2% and sentiments prediction by 6.5% in comparison to the baseline approach.

In addition to this, it was proven that the model could predict all of the emotion category classes in the dataset, even though some of the classes consist of a significantly lower quantity of elements.

Finally, the authors outline some concepts for further research to increase the model's performance. Those are the following:

- Training model using a more complex BERT model, for example, BERT-large;
- Conduct experiments with a lower learning rate;
- Increase the amount of data for training and testing.

The aim of future work will be to improve the emotions classification of the text data using a deep learning approach. This can also include either research of BERT modifications, such as ALBERT [27], RoBERTa [28], and DistilBERT [29], or other deep learning models capable of language understanding, for example, XLNet [5].

The trend nowadays is toward large language models, which are multi-functional and can be used for various language understanding tasks. Fine-tuning a large language model for emotions and sentiment classification is also a promising direction for further research.

Author Contributions: Conceptualization, I.B. and O.M.; Methodology, M.S. and O.M.; Software, M.S. and O.M.; Validation, I.B. and M.S.; Formal analysis, M.S. and O.M.; Investigation, M.S. and O.M.; Data curation, I.B. and M.S.; Writing—original draft, M.S.; Writing—review & editing, I.B.; Visualization, I.B.; Supervision, I.B.; Project administration, I.B.; funding, I.B. All authors have read and agreed to the published version of the manuscript.

Funding: Research project supported by the program "Excellence initiative—research university" for AGH University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All code presented in this study is published in the GitHub repository: https://github.com/Shoomaher/sentiment-analysis (accessed on 25 February 2024).

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A Classes Distribution for Training, Validation, and Test Datasets

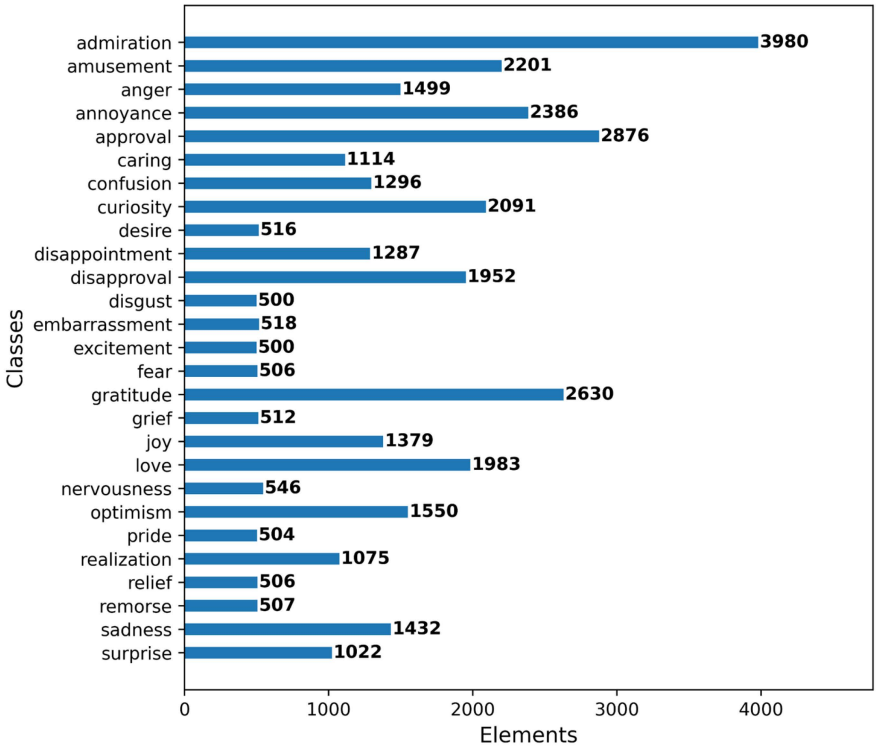

Figure A1. Classes distribution in the training dataset.

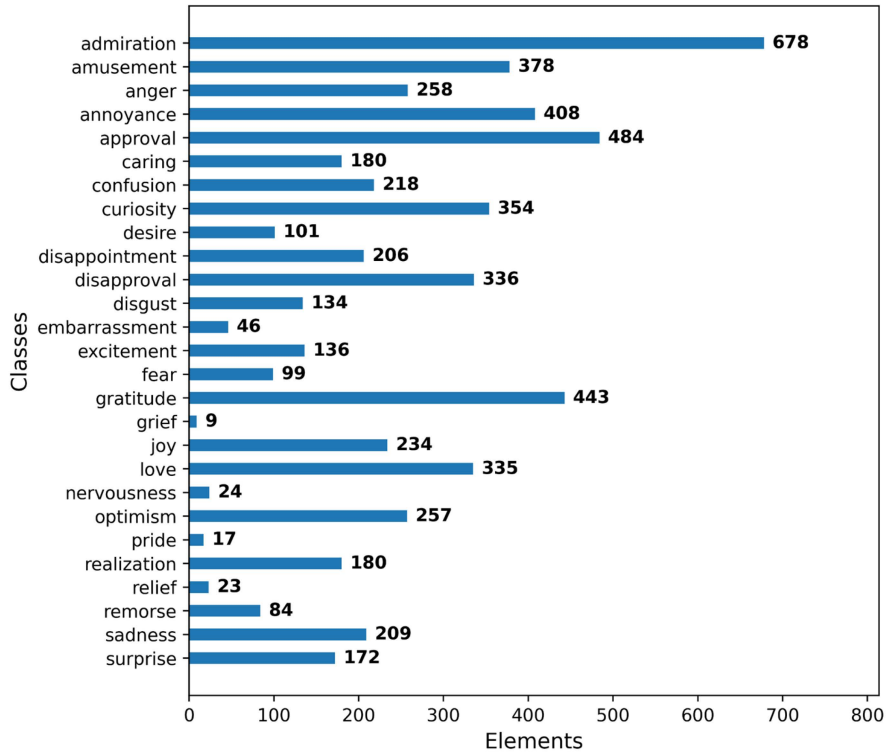

Figure A2. Classes distribution in the validation dataset.

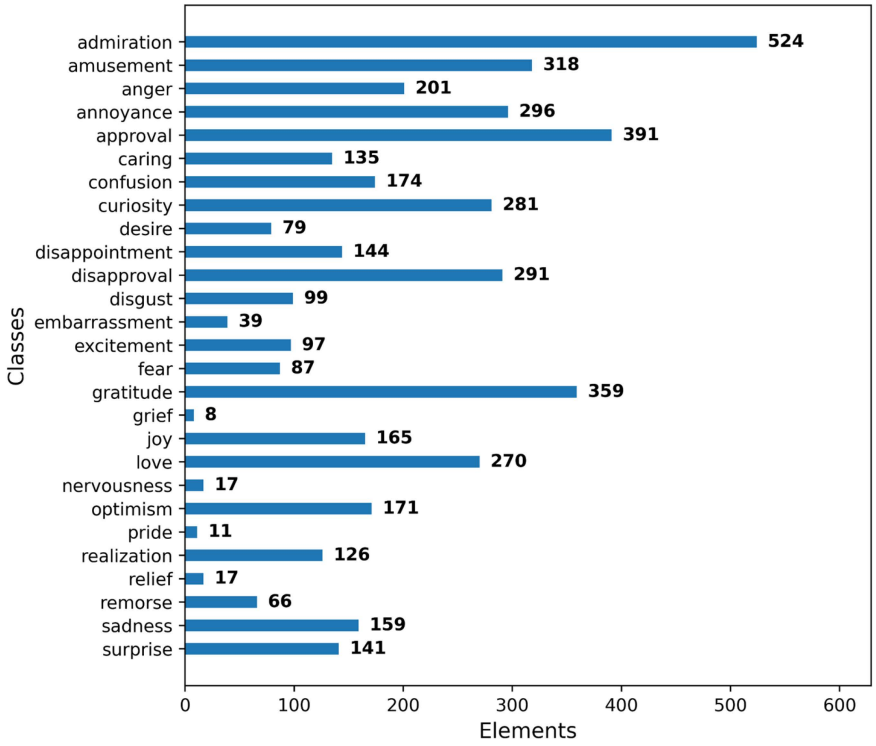

Figure A3. Classes distribution in the testing dataset.

Appendix B

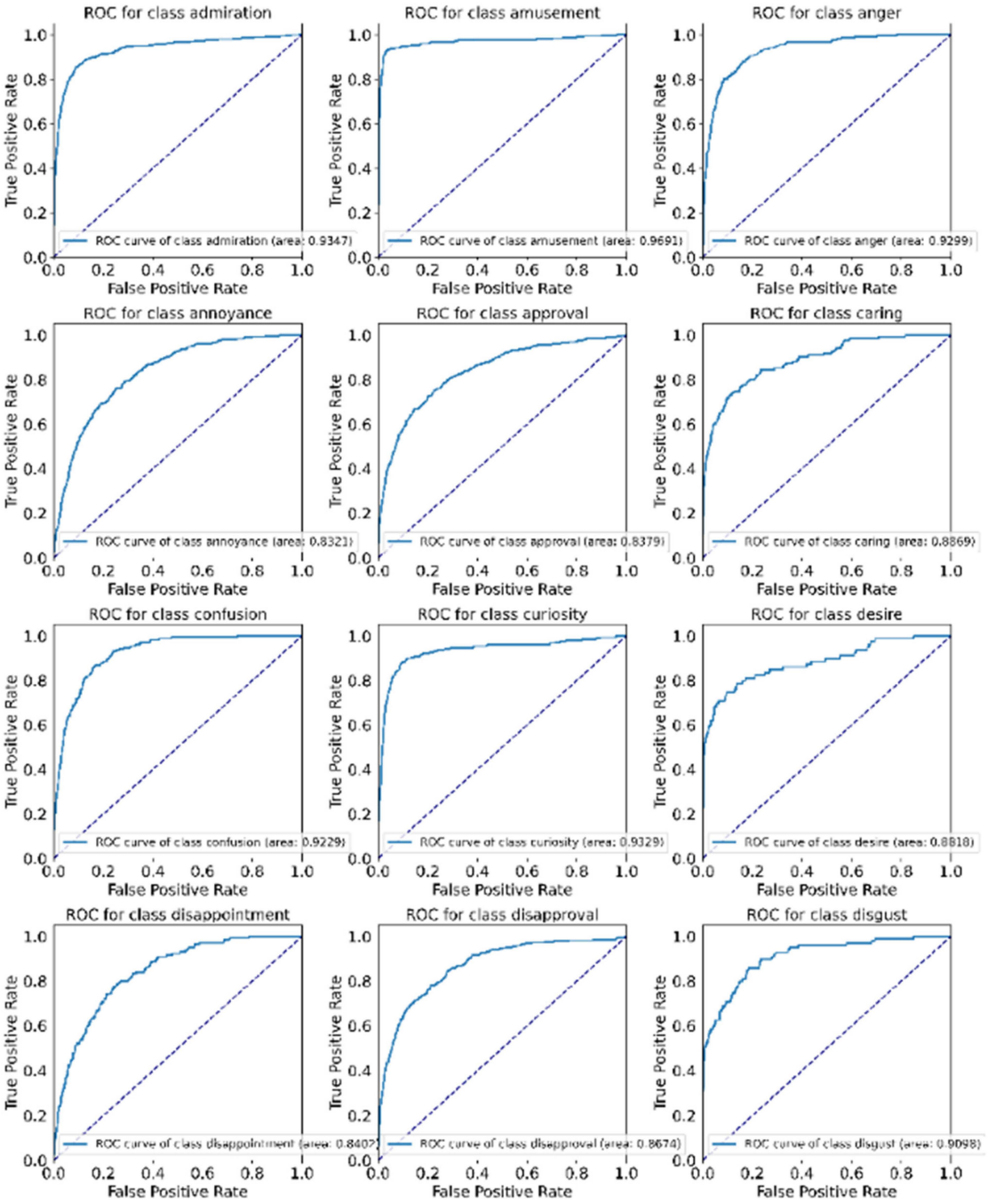

Figure A4. *Cont*.

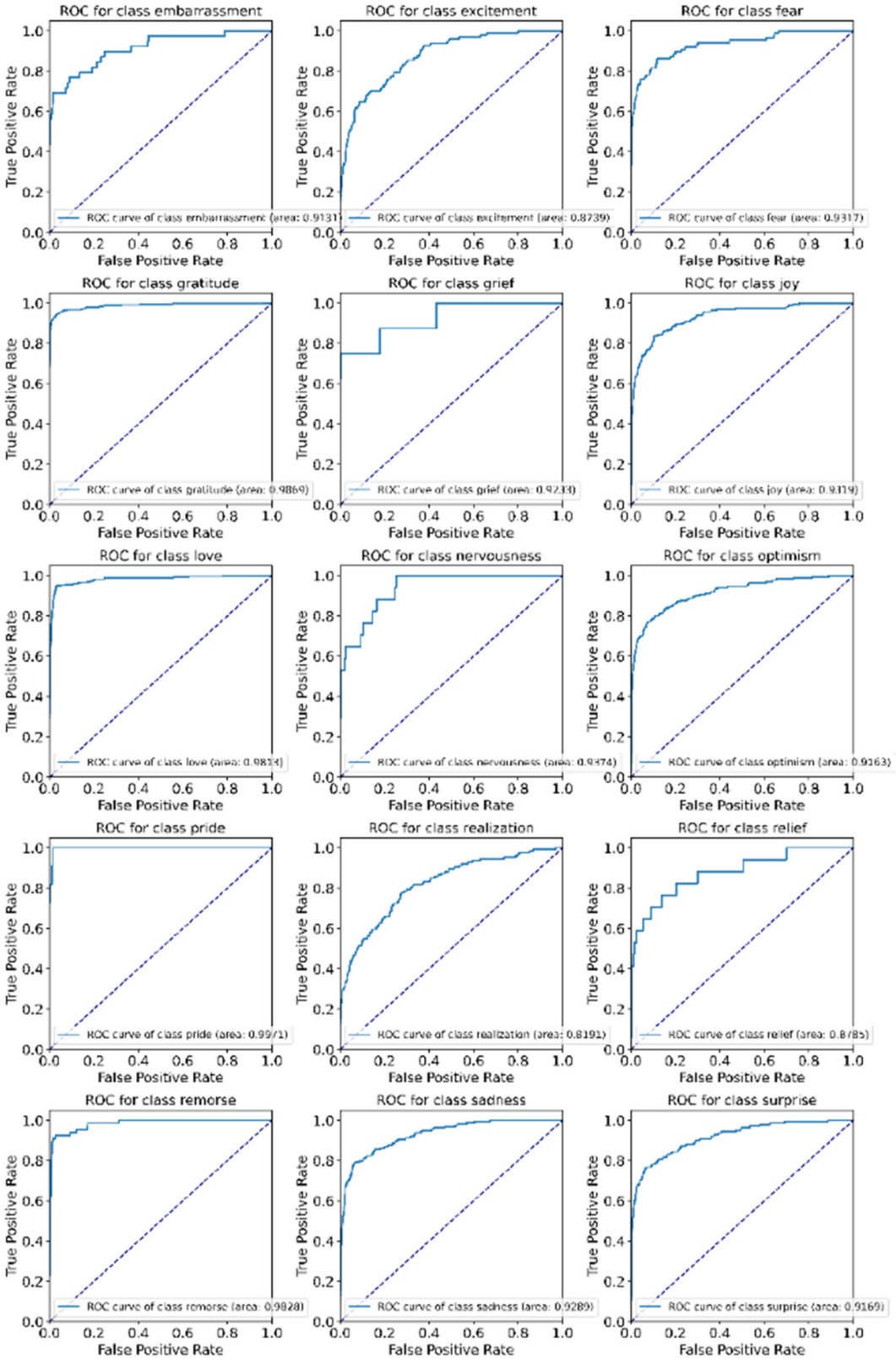

Figure A4. ROC curves for each emotion category class.

Appendix C

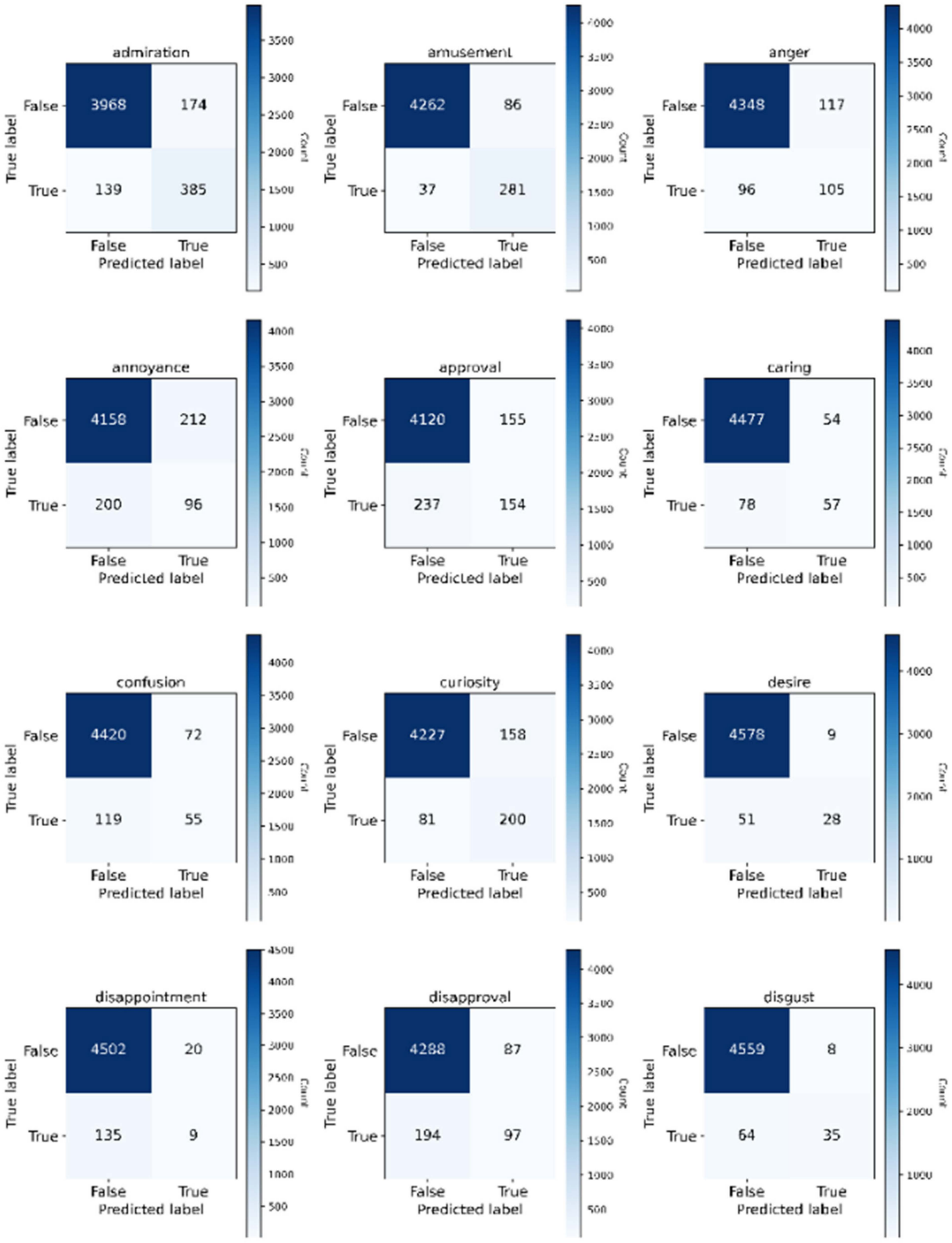

Figure A5. *Cont*.

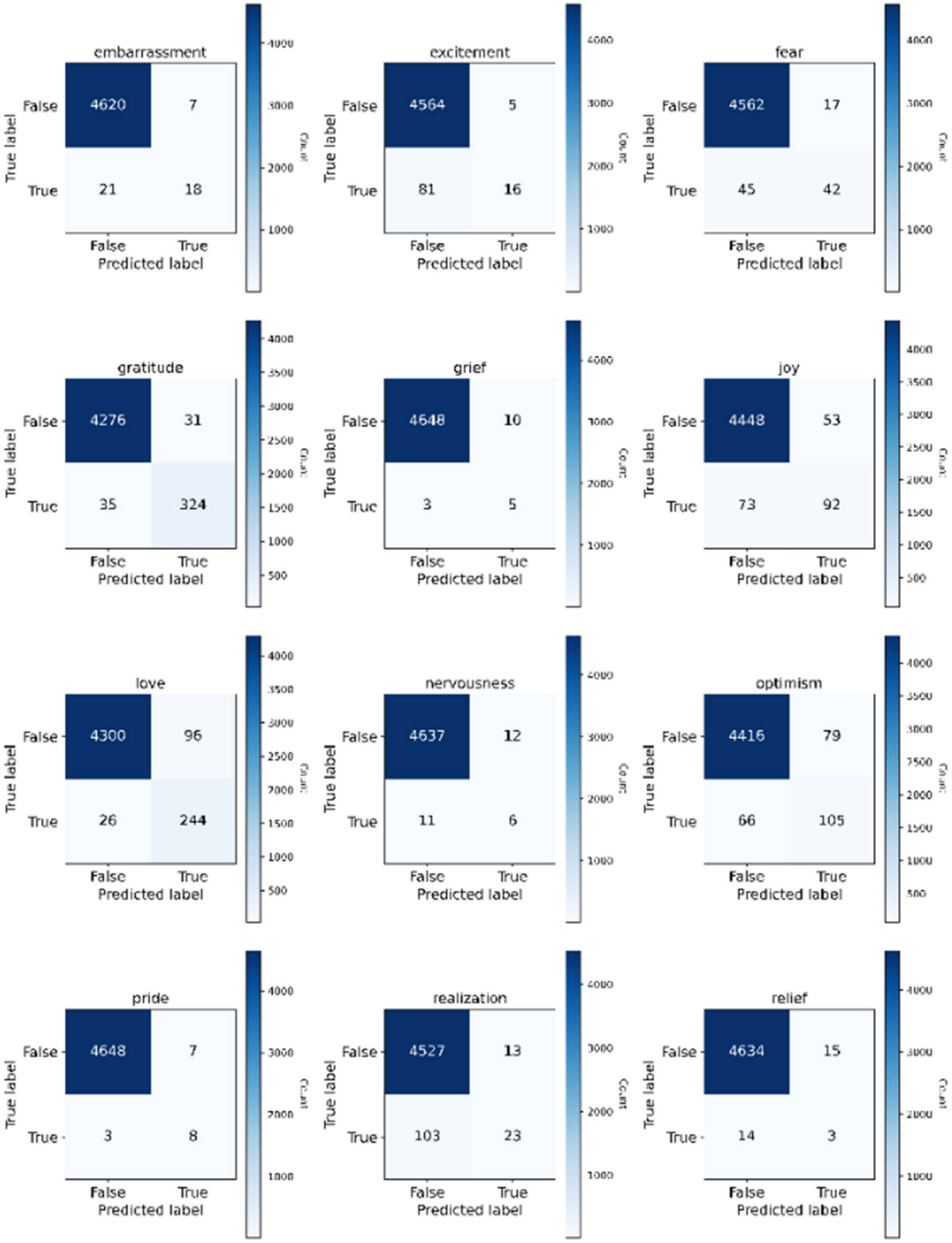

Figure A5. Confusion matrixes for each emotion category class.

Appendix D

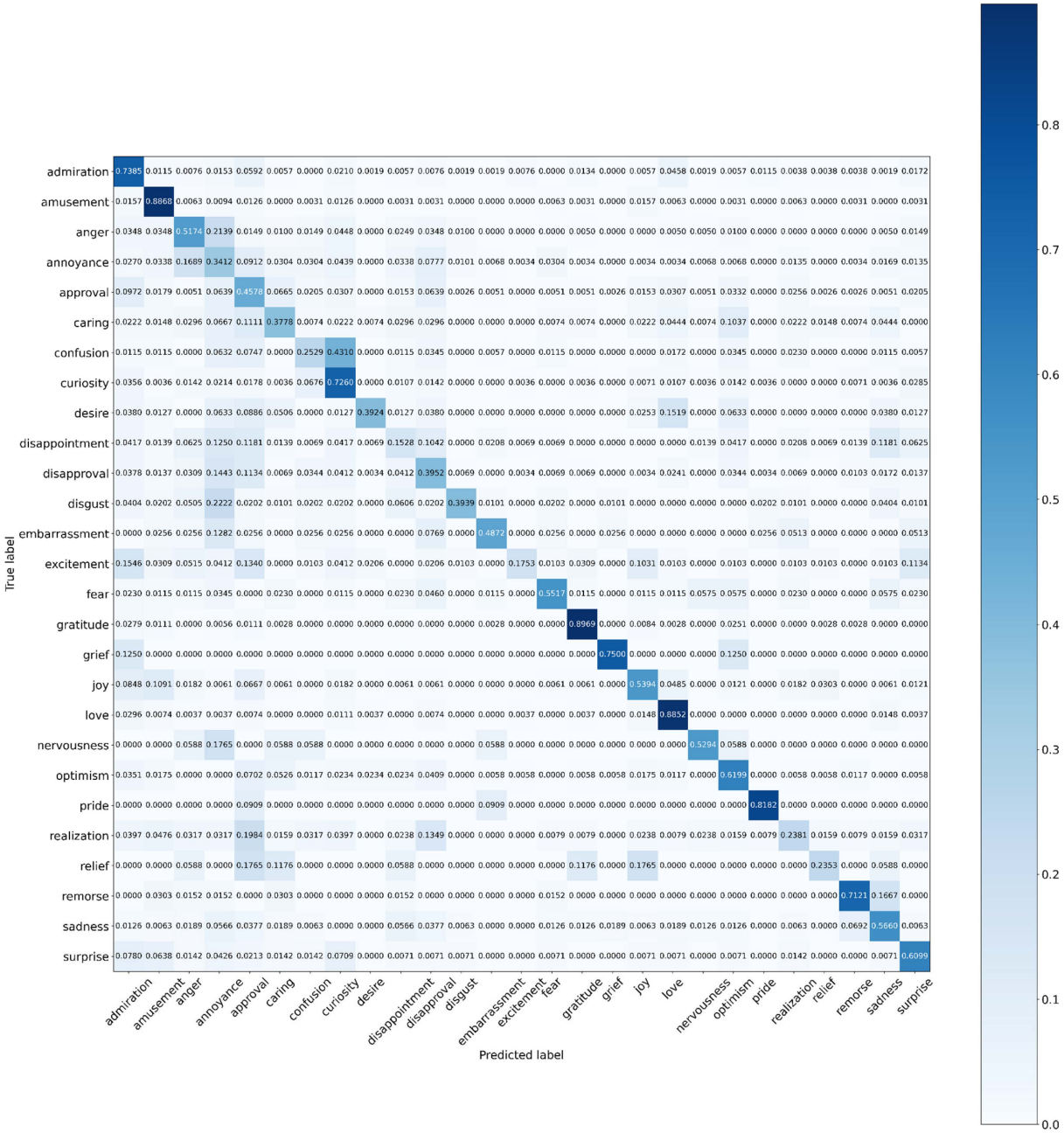

Figure A6. Confusion matrix for all classes (all-vs-all strategy).

Appendix E

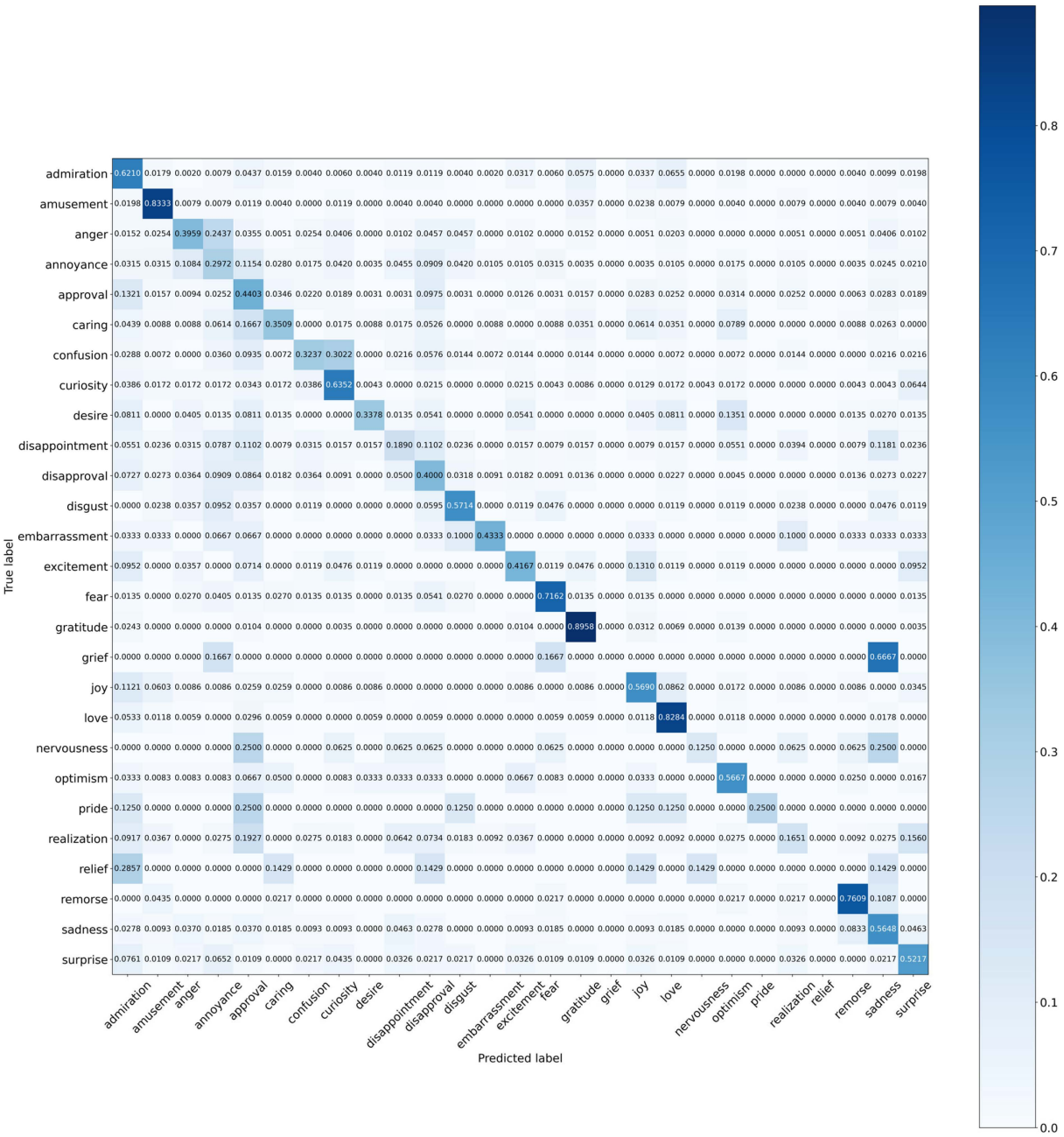

Figure A7. Confusion matrix for all classes (all-vs-all strategy) on original GoEmotions train/validation/test split.

References

1. TensorFlow API Documentation. TextVectorization: A Preprocessing Layer Which Maps Text Features to Integer Sequences. 2023. Available online: https://www.tensorflow.org/api_docs/python/tf/keras/layers/TextVectorization (accessed on 15 November 2023).
2. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
3. Lample, G.; Conneau, A. Cross-lingual language model pretraining. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 7059–7069.
4. Patra, B.; Singhal, S.; Huang, S.; Chi, Z.; Dong, L.; Wei, F.; Chaudhary, V.; Song, X. Beyond English-centric bitexts for better multilingual language representation learning. *arXiv* **2022**, arXiv:2210.14867.

5. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized autoregressive pretraining for language understanding. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 5753–5763.
6. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the NAACL HLT 2019—2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies—Proceedings of the Conference, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
7. Demszky, D.; Movshovitz-Attias, D.; Ko, J.; Cowen, A.; Nemade, G.; Ravi, S. GoEmotions: A dataset of fine-grained emotions. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 4040–4054.
8. Reddit. Dive into Anything. Available online: https://www.reddit.com/ (accessed on 15 November 2023).
9. Pandas. Python Data Analysis Library. Available online: https://pandas.pydata.org/ (accessed on 15 November 2023).
10. Numpy. The Fundamental Package for Scientific Computing with Python. Available online: https://numpy.org/ (accessed on 15 November 2023).
11. Matplotlib. Visualization with Python. Available online: https://matplotlib.org/ (accessed on 2 December 2023).
12. Unidecode. ASCII Transliterations of Unicode Text. Available online: https://pypi.org/project/Unidecode/ (accessed on 12 November 2022).
13. TensorFlow. An end-to-End Machine Learning Platform. Available online: https://www.tensorflow.org/ (accessed on 12 November 2022).
14. TensorFlow. Classify Text with BERT. Available online: https://www.tensorflow.org/text/tutorials/classify_text_with_bert (accessed on 12 November 2022).
15. Scikit-learn. Machine Learning in Python. Available online: https://scikit-learn.org/ (accessed on 15 November 2023).
16. Cortiz, D. Exploring Transformers in Emotion Recognition: A comparison of BERT, DistillBERT, RoBERTa, XLNet and ELECTRA. *arXiv* **2021**, arXiv:2104.02041. [CrossRef]
17. Shen, W.; Chen, J.; Quan, X.; Xie, Z. DialogXL: All-in-One XLNet for Multi-Party Conversation Emotion Recognition. *arXiv* **2020**, arXiv:2012.08695. [CrossRef]
18. Read, J.; Pfahringer, B.; Holmes, G.; Frank, E. Classifier chains for multi-label classification. *Mach. Learn.* **2011**, *85*, 333–359. [CrossRef]
19. Sharaf, S.; Anoop, V.S. An Analysis on Large Language Models in Healthcare: A Case Study of BioBERT. *arXiv* **2023**, arXiv:2310.07282. [CrossRef]
20. Saravia, E.; Liu, H.-C.T.; Huang, Y.H.; Wu, J.; Chen, Y.S. CARER: Contextualized Affect Representations for Emotion Recognition. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; Association for Computational Linguistics: Brussels, Belgium, 2018; pp. 3687–3697. [CrossRef]
21. Tucker, A. Contractions. San José State University Writing Center. 2011. Available online: https://www.sjsu.edu/writingcenter/docs/handouts/Contractions.pdf (accessed on 20 February 2024).
22. Kaggle. Getting Started with Text Preprocessing. 2019. Available online: https://www.kaggle.com/code/sudalairajkumar/getting-started-with-text-preprocessing (accessed on 10 November 2023).
23. TensorFlow. BinaryCrossentropy. TensorFlow API Documentation. Available online: https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy (accessed on 15 November 2023).
24. TensorFlow Hub. BERT (Bidirectional Encoder Representations from Transformers). Available online: https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/3 (accessed on 15 November 2022).
25. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
26. Morris, J. Does Model Size Matter? A Comparison of BERT and DistilBERT. Weights & Biases. 2022. Available online: https://wandb.ai/jack-morris/david-vs-goliath/reports/Does-Model-Size-Matter-A-Comparison-of-BERT-and-DistilBERT--VmlldzoxMDUxNzU (accessed on 20 November 2022).
27. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv* **2019**, arXiv:1909.11942. [CrossRef]
28. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692. [CrossRef]
29. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108. [CrossRef]