From Data to Insights: Developing an NLP Model for Banijay Benelux

Max Meiners

Dániel Elek

Yuliia Bobrovytska

Breda University of Applied Sciences

Mentor: Dean van Aswegen

Table of Contents

Abstract

Natural Language Processing (NLP) has significantly advanced, enabling computers to analyze and generate human language effectively. Emotion classification, a subfield of NLP, is crucial for understanding sentiments and behaviors in text, with applications ranging from analyzing social media sentiments to multimedia content. This task is challenging as human emotions are influenced by context, tone, and cultural background, and can vary from subtle to overt expressions.

This project centers on emotion classification in the TV series "Expeditie Robinson" ("internationally known as Survivor"), utilizing data from Banijay Benelux and 3Rivers media consultants. It aims to enhance the understanding and use of NLP in multimedia, focusing on viewer engagement.

The introduction outlines the project's goals, methods, and significance. It describes the phases of data collection, model development, and evaluation, and discusses the practical outcomes, including a pipeline for emotion tagging in multimedia content. This work not only advances NLP applications but also offers insights into human cognition through computational analysis. (OpenAI, 2024)

Introduction

Emotion classification in natural language processing (NLP) is essential for understanding human sentiment and behavior. This project aims to develop models for emotion classification, specifically for the TV series "Expeditie Robinson", provided by Banijay Benelux, to analyze viewer engagement.

The project begins with an exploration of emotion theories, particularly Paul Ekman's six core emotions: happiness, sadness, anger, surprise, fear, and disgust. It acknowledges the complexities of interpreting emotions from linguistic data, considering factors like tone and context.

A phased approach includes acquiring annotated datasets and developing models trained on general emotion data, before integrating and fine-tuning with "Expeditie Robinson" data. The methodology employs NLP techniques for data processing and classification, including feature extraction and sentiment analysis, using machine learning algorithms from Naïve Bayes to Transformer models.

The project also explores speech recognition, extracting insights from the series' episodes and developing a pipeline for emotion classification from speech-to-text conversion to emotion tagging.

Finally, the project culminates in submitting models to Kaggle challenges, producing a technical report and a final presentation that covers data processing, feature engineering, model selection, and evaluation, offering a detailed overview of the methodologies, findings, and implications for content analysis and viewer engagement in television. (OpenAI, 2024)

Data Processing and Exploration

In this section, we delve into the crucial preliminary steps of any data-driven project: data processing and exploration. Before diving into model building, it's essential to thoroughly understand the datasets at hand, clean and preprocess them appropriately, and explore the underlying patterns and insights they contain. Data processing involves tasks such as handling missing values, removing noise, and transforming the data into a suitable format for analysis. Exploration, on the other hand, entails visualizing the data, identifying correlations, and gaining a deeper understanding of its characteristics.

Effective data processing and exploration lay the foundation for building robust and reliable models for emotion classification. By uncovering hidden relationships and nuances within the data, we can make informed decisions about feature selection, model architecture, and training strategies. Additionally, thorough exploration helps us identify potential biases or limitations in the data, enabling us to address them proactively and ensure the fairness and integrity of our model. (OpenAI, 2024)

Data Collection and Preprocessing:

We began by collecting and preprocessing multiple datasets listed on the "Datasets" page (given to us from the university). This includes; GoEmotions, Smile Twitter Emotion, Friends (Emotion labeled dialogue), MELD, CARER, Affective Text, Daily Dialogue, Affect data, and the Emotion Dataset for Emotion Recognition Tasks from Kaggle)

We understood, that we did not have enough data for some emotions and we decided to generate the additional data with the help of the API of OpenAI. Each dataset underwent extensive cleaning and preprocessing to ensure uniformity and consistency across the combined dataset. This involved removing the duplicates, standardizing text formats, checking if all the sentences had capital letters and punctuations at the end of the sentence, and converting tertiary and secondary emotions into Paul Ekman's six core emotions (happiness, sadness, anger, surprise, fear, disgust)[1]. (OpenAI, 2024)

Data Integration and Augmentation

After preprocessing, we started integrating the datasets into a unified corpus for emotion classification. This involved combining the cleaned data from different sources while maintaining a balance between the number of instances for each emotion category. Augmentation aims to increase the variability of training instances and improve model generalization. To enhance the diversity and richness of the dataset, we performed a data augmentation technique - synonym replacement, but it gave us a worse score, so we decided to stay with the dataset that we had. Augmentation aims to increase the variability of training instances and improve model generalization. (OpenAI, 2024)

Feature Extraction and Representation

We were able to use various suitable feature extraction techniques. We implemented a pre-trained word embedding model (Word2Vec, created by Google) and extracted embeddings as features for the model's input, we performed TF-IDF, we extracted Part-of-Speech tags from the data, and we created a word embedding model ourselves. Unfortunately, all of these features did not help us to improve the score of the model's performance and we stuck with only using the sentences as a feature for future training.[2] (OpenAI, 2024)

[1] Paul Ekman's six core emotions and the composition of our final dataset can be seen on the "Figures" page in this report. (Figures 3 and 4, respectively)

[2] The final dataset we used for training our best-performing models. This can be seen on the "Figures" page in this report. (Figure 5)

Model Selection and Implementation

To get to our best-performing model, we started working on several other models that led us to our end goal. In this section, we will talk about how that process went and what steps we took to create our best-performing model.

Model Overview:

In the development of our NLP model aimed at emotion classification, we did an extensive evaluation of several machine learning and deep learning models to get the most effective approach for our dataset and task. The initial phase of model selection involved experimenting with a broad number of different models, including traditional machine learning models such as Random Forest Classifier (RFC), Naïve Bayes, Logistic Regression, and XGBoost. We also worked on more complex neural network architectures like Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTM). Furthermore, the implementation of Term Frequency-Inverse Document Frequency (TF-IDF) vectorization has been used as a method for transforming text data into a suitable format for model training.

RFC, while generally well-performing across a variety of tasks, demonstrated a bad performance in our specific application of emotion classification. Typically, RFC is praised for its ability to handle tabular data with high accuracy. However, its design proved less effective for the analysis required for distinguishing between human emotions in text.

After working on the RFC models, our focus moved towards more advanced neural network architectures capable of capturing the contextual relationships within text. This led us to implement and evaluate several transformer-based models, including BERT (Bidirectional Encoder Representations from Transformers), FINBERT (a BERT model pre-trained on financial texts), RoBERTa (a robustly optimized BERT approach), and Electra (Efficiently Learning an Encoder that Classifies Token Replacements Accurately). Transformers are particularly well-suited for NLP tasks as they excel in understanding the context and relationships between words in a sentence, so they should provide a significant advantage in emotion classification.

Our findings revealed that both BERT and RoBERTa delivered impressive results, outperforming most of the traditional models and other neural networks we tested. Between those two transformer models, RoBERTa emerged as the slightly more effective model for our specific task. This is likely because RoBERTa was trained with more data and on longer pieces of text, giving it a deeper grasp of the subtleties in language.

With these results, we decided to further focus our attention on experimenting with RoBERTa, refining our model through hyperparameter tuning and evaluation. The following sections will detail the implementation process, hyperparameter tuning, and the adjustments made to fully leverage RoBERTa's capabilities.

Table 1: Machine Learning Models with their corresponding F1 score

Machine Learning Models	F1 score
Random Forest Classifier	0.406
Naïve Bayes	0.741
Logistic Regression	0.796 (0.829 using TF-IDF vectorizer)
XGBoost	0.457
Recurrent Neural Network	0.487
Long Short-Term Memory network	0.590
Electra	0.680
BERT	0.761
RoBERTa	0.884

Modifications and Adaptations:

To improve RoBERTa's performance for our emotion classification task, we focused on several key modifications. We adjusted key hyperparameters, including the learning rate, batch size, and epoch count, to optimize RoBERTa's accuracy and efficiency. This was achieved through manual tweaking based on performance feedback. We also played around with a learning rate scheduler, which means that it gradually adjusts the learning rate throughout the training process. By implementing this, we ensured a systematic increase in the learning rate from a lower starting point to a specified higher rate during the initial epochs.

At first, we had a major class imbalance. In the first datasets we used, there was a lack of sentences with the emotions 'anger', 'fear', and 'disgust'. To address this, we made use of the OpenAI API to generate sentences for those specific emotions. We then trained our model again, with the updated dataset, and plotted a Confusion Matrix. Whichever emotion was predicted the worst, we generated new sentences for. With the use of this API, we gathered around 20.000 sentences per emotion that we were lacking. When training our model again, with the implementation of these generated sentences, the accuracy improved immensely.

Furthermore, we incorporated an early stopping callback and a custom F1 score metric[3]. Early stopping helped against overfitting by monitoring the validation loss and stopping training when the loss did not improve anymore over a time of three epochs.

At some point, we struggled with predicting on the test dataset we had received. After a lot of tweaking, it turned out that we were predicting the results with a different tokenizer than what we trained our model on. After fixing this, we were back on track to building our best model possible.

[3] The figures for the Early Stopping callback and the custom F1 score metric can be seen on the "Figures" page of this report. (Figures 6 and 7 respectively)

Hyperparameter Tuning:

To fine-tune our RoBERTa model, we took an approach to setting the hyperparameters, using both automatic tools and our own trial-and-error process. We started with a batch size of 128, which gave us a good mix of speed and learning stability while also depending on the capacity of the server we were using. For our best-performing model, we set the learning rate at 2e-5, which was slow enough to let the model learn thoroughly without missing the best solution. We played around with the learning rate, but we came to the conclusion that a learning rate of 2e-5 worked best for our specific model. We decided on 100 epochs for training, allowing the model to learn as much as possible without memorizing the data, but with the Early Stopping callback we implemented, it usually stopped around epoch seven because by then the validation loss would increase slightly.

We used 20% of our data as a test set to check how well the model was doing. To make sure we could get the same results again, we used a fixed random state of 42. We also kept our input text lengths to 128 tokens, which matched RoBERTa's training and was long enough to get the full meaning of most sentences without wasting computational power.

A special part of our setup was a custom learning rate scheduler that gradually increased the learning rate at the start of training. This approach helped the model start learning slowly and then pick up speed, which made it more likely to find the best solution. However, the learning rate scheduler was not implemented in our best-performing model, since the F1 scores seemed to not increase as much as we wanted to when using this approach.

Training Process:

During the training process of our RoBERTa model, we paid attention to the composition of our training and testing datasets to ensure that they accurately represented the distribution of all six unique classes of emotional expression. To this end, when splitting the data into training and testing sets, we used stratification. This is a technique that ensures that each split of the dataset contains approximately the same percentage of samples of each target class as the original dataset. We utilized 80% percent of our data for training and the remaining 20% for testing, stratifying both splits to reflect the overall composition of our dataset.

Training each model usually took around half a day, since we trained the model on our entire dataset, which contained a total of around 740.000 rows. If the server allowed it, we processed the data in batches of 128 samples. For faster training results, we utilized the server’s GPU capabilities.

To conclude, the combined use of a stratified approach, careful monitoring of training, and optimal utilization of high-end GPUs enabled us to conduct a series of training iterations. These iterations were key to refining our model’s accuracy.

Transcription

After we have created a robust model that we were happy with, we could start working on implementing it to the data provided by the client. We received MOV files of an entire season of the series, meaning it was a massive dataset. To make sure we wouldn’t run into resource exhaustion errors, we decided to first strip the audio from the image data, for which we used the open source software FFmpeg. This was a straightforward process, but transcribing it proved to be more challenging.

We decided to use OpenAI’s Whisper model, at first just the base version, to complete this task. Setting up the model was not the hard part, as transcribing the episodes went without problems when testing it out. What caused the complications, however, was that we needed to first split the episodes into separate segments, predefined by Banijay. The CSV file they provided us was very disorganized and contained mistakes too; the start times of some segments in a few episodes were all the same, so we first needed to correct these issues. Afterward, we could begin splitting the audio into the required segments to create separate transcriptions. At this point, even though we tried avoiding it, we ran into resource exhaustion errors, specifically VRAM errors. The logical next step was to try running the code on the provided server that had much more GPU memory, but this also did not yield the expected results, still, the same error message popped up. We tried splitting the audio into even smaller pieces, but the problem remained.

The breakthrough came when we had a conversation with an expert in the field, who suggested taking apart the code into smaller steps, meaning first splitting into segments, saving the files to a directory, and then loading them back in for transcription. With this approach, we managed to run the code. Here we decided to shift to using OpenAI's large-v3 model, to have the best possible quality of data. Seeing how long each transcription took, we moved the process to the server provided by the university and changed the code to save the files when an episode is fully transcribed, just in case the server shuts down.

After the transcriptions and translations were done, we only had to merge this new DataFrame with the one provided to us by Banijay, containing the segments and the labeled emotions.

Final pipeline:

After 8 weeks of hard work, we built the entire pipeline for the client. The pipeline is able to receive MOV files, convert them into MP3s, split them into predefined segments, transcribe and translate them, and finally predict what emotions are present in which segment.

What we did notice, however, is that there was a gap between what our machine-learning model was trained to do, and the structure of the dataset provided by Banijay. We trained our algorithm to receive one sentence as an input and give an emotion as an output, but Banijay labeled their data differently. They labeled entire paragraphs, reliably containing more than 20 sentences with either one or multiple emotions[4], handling this proved very difficult for our classifier model. Our approach was to

[4] The figures either one or multiple emotions can be seen on the "Figures" page in this report. (Figure 8)

replace these detailed emotions with their core counterparts, and then only keep one emotion as label, the one that appeared the most from those core emotions. If there were equal amounts of multiple motions, we chose one randomly. We will go more in-depth about this in the error analysis section.

Assessment Methods and Outcomes

In emotion classification, the effectiveness of prediction models is critical. Our analysis focuses on the types of errors made by our model across different datasets, such as questionable labels, hard-to-classify sentences, and incorrect predictions. By exploring these errors, we aim to identify limitations and guide improvements in our model's ability to interpret human emotions accurately.

Error Analysis:

When training our final model on our local dataset, it worked remarkably well, however, it still produced some errors. We found 3 categories of mistakes: questionable labels, hard-to-classify sentences, and simple wrong predictions.

<u>Questionable labels</u>: Since the data was collected from many different sources, different labeling techniques might have been used, but also because some of the dataset was not hand labeled, we found such misclassifications:

Table 2: Questionable labels category

Sentence	True Label	Predicted Label
"No negotiations with terrorists."	Sadness	Happiness
"No sorry still looking."	Happiness	Anger
"Do you have any tinder picture or bio advice? I have no selfies or mirror pics but still not doing very well."	Disgust	Happiness

Hard to classify sentences: The sentences in this category could easily be interpreted as a different emotion or require some context to really tell what is happening exactly.

Table 3: Hard to classify category

Sentence	True Label	Predicted Label
"What a nasty goal! Gives you chills!"	Surprise	Happiness
"I I'm so honored."	Surprise	Happiness
"Does this attitude go for anything where you accidentally end up killing someone via stupid behaviour?"	Anger	Surprise

These mistakes were found in all emotions. To fix this issue we would need to manually inspect all 740821 rows, which is of course not feasible. Another fix would be to only use hand-labeled datasets, but we found that our model generalized a lot better when we added the non-hand-labeled data too, but it comes with this tradeoff.

Model Performance and Dataset Challenges

Due to the unbalanced dataset, the classifier is better at recognizing some emotions than others, even if the difference is small. For example, it is the best at identifying happiness, as 31.4% of the dataset is happiness. On the other end of the spectrum is surprise, it comprises 11.4% of the dataset, and it is the emotion that the model is the least confident in. It is interesting to see that disgust is the least represented label in the dataset (9.5%), yet it is more accurate than surprise. We think that the most likely explanation is that happiness and surprise often go together in sentences, so the model often cannot decide which one it should predict. It is supported by the fact that, as visible in the Confusion Matrix (Figure 1), the most common wrong label that the model assigns "surprise" sentences is "happiness", more than five times as often as with the second most common wrong label for this emotion: anger.

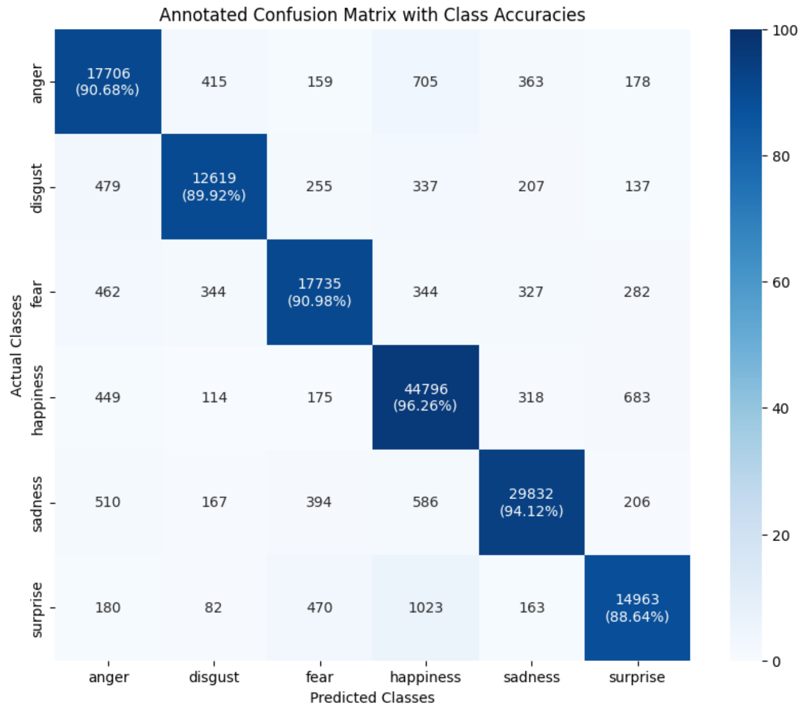

Figure 1: Confusion Matrix

When looking at the mistakes in the Robinson dataset, we have a very different picture. Because of the unusual structure of the dataset mentioned in the pipeline section, our model does not do very well on it. It had an accuracy of 62%, but this was mostly due to the fact that most labels were happiness. When looking at the accuracy of each specific emotion, it is obvious that they all do terribly, except for happiness, as visible in figure 2.

For example, the model could not predict a single disgust label correctly and only predicted one surprise label well.

As this is an enormous difference in performance when compared to our local dataset or even to the Kaggle dataset, we had to find the answer that could explain this big gap in accuracy. We found two factors that are most likely responsible for it.

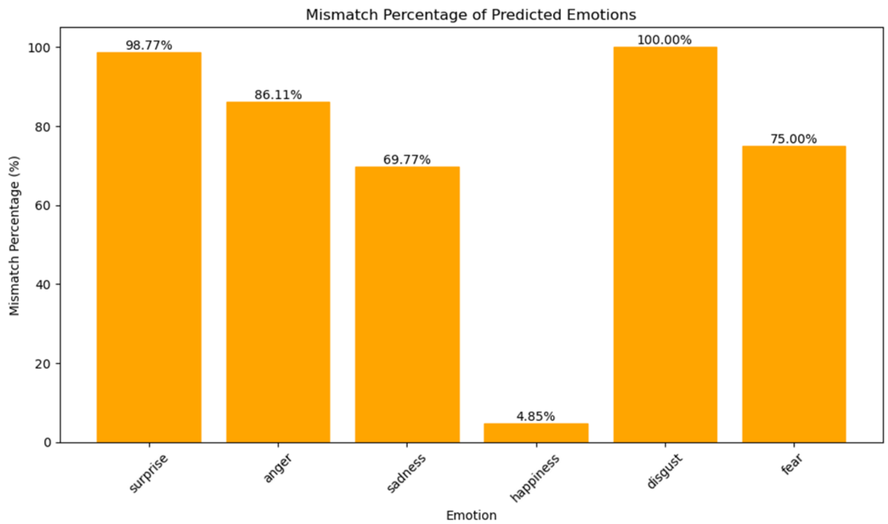

Figure 2: Accuracy of each emotion

The first and biggest one is that the model is not built for the task that this data structure requires. The task we trained it for is to output one emotion when given one sentence as an input. This dataset requires it to make predictions on many sentences at once, making it a lot more difficult to pay attention to the nuances.

The second factor is that we did not have a balanced dataset[5]. An interesting correlation is that the order of the accuracy from worst to best almost exactly matches the order of the emotions in the dataset from least to most represented. Improving the balance of labels could help improve performance, but because of the previous reason, it would most likely be a small improvement.

[5] Figure 4: The composition of the final dataset. This can be seen in the “Figures” page in this report

Discussion and Conclusion

In this final section, we delve into the discussion and conclusion of our project on emotion classification in natural language processing (NLP). Here, we reflect on the methodologies, findings, challenges, and implications of our work, aiming to provide a comprehensive overview of our journey from data processing to model implementation. (OpenAI, 2024)

Project Reflection:

Reflecting on our journey through the development of an NLP model for emotion classification in the context of Banijay Benelux's TV series "Expeditie Robinson", we encountered both successes and challenges that contributed to our growth and learning throughout the project.

One of the key successes of our project was the comprehensive exploration and integration of multiple datasets for emotion classification. By combining diverse sources of data, we were able to create a rich and extensive corpus that captured a wide range of emotional expressions. This approach not only enhanced the robustness of our model but also provided valuable insights into the nuances of human emotions as expressed in textual data.

However, this amalgamation of datasets also presented challenges, particularly in terms of data cleaning, preprocessing, and integration. We encountered inconsistencies in data formats, missing values, and class imbalances, which required meticulous attention to detail and careful handling to ensure the quality and integrity of our dataset. Despite these challenges, overcoming them strengthened our data processing skills and underscored the importance of thorough data exploration and preparation in the model development process.

Another success of our project was the systematic evaluation and selection of machine learning and deep learning models for emotion classification. Through experimentation with various algorithms, architectures, and techniques, we gained valuable insights into the strengths and weaknesses of different approaches. This iterative process of model selection and implementation allowed us to identify the most effective solution for our task while deepening our understanding of NLP techniques and methodologies.

One unexpected finding during the project was the effectiveness of transformer-based models, particularly RoBERTa, in capturing the contextual nuances of emotional expressions in text. While we initially explored traditional machine learning models and recurrent neural networks, the superior performance of transformer models highlighted the importance of contextual understanding in emotion classification tasks. This unexpected insight informed our decision to focus on RoBERTa and refine its implementation through hyperparameter tuning and experimentation. (OpenAI, 2024)

Future Recommendations:

As we deploy our model into real-world scenarios, the necessity of fine-tuning it on domain-specific data becomes apparent, aiming to enhance its performance and adaptability. This ongoing process involves not only collecting and incorporating feedback from users but also retraining the model to grasp evolving language patterns and sentiments. To deepen the model's comprehension of emotions, integrating multimodal data sources like images, audio, and video could be incredibly beneficial. Given that emotions are often conveyed through non-verbal cues, such an approach promises a richer understanding of context and sentiment.

Building a feedback loop where users can critique and comment on the model's predictions allows for a dynamic refinement process, honing the accuracy and personalizing responses over time. It's crucial, however, to navigate the ethical landscape carefully, ensuring the model operates transparently, equitably, and without bias, thus respecting user privacy and promoting algorithmic fairness.

Another way of making final predictions on the test dataset is by predicting on each sentence separately per segment, and then determining which emotion appears most often in each segment. This method could enhance the accuracy and relevance of the model's predictions in complex datasets like the Robinson dataset.

The challenge of emotion classification extends beyond linguistic boundaries, urging a cross-lingual and cultural adaptation to cater to diverse users. This could involve leveraging translation services, employing cross-lingual learning techniques, or collaborating with experts in local languages and cultures. A continuous cycle of monitoring and evaluation is pivotal to maintain and elevate the model's real-world efficacy, guided by a thorough analysis of performance metrics and user insights.

Engagement with the broader academic and professional community is also beneficial, sharing insights and resources to collectively tackle challenges within emotion classification and NLP. Such collaboration fosters innovation and addresses emerging issues more effectively. Ensuring the infrastructure's scalability and efficiency underpins the model's success, necessitating a vigilant approach to resource management and possibly cloud-based solutions for enhanced demand handling.

Comprehensive documentation and open sharing of research findings and methodologies encourage replication and further exploration within the community, contributing significantly to collective knowledge. Finally, empowering users through education about the model's capabilities and boundaries ensures informed interactions and mitigates potential misconceptions, fostering a responsible and enlightened user base. (OpenAI, 2024)

Concluding Remarks:

Overall, our project provided us with valuable hands-on experience in the application of NLP techniques to real-world problems, particularly in the domain of emotion classification. Despite the challenges encountered along the way, our perseverance, teamwork, and dedication ultimately led to the development of a robust and effective NLP model. Moving forward, we are excited to apply the lessons learned from this project to future endeavors in NLP research and development, contributing to advancements in understanding human emotions through computational methods. (OpenAI, 2024)

References

OpenAI. (2024). *ChatGPT* (Mar 14 version) [Large language model]. https://chat.openai.com/chat

Figures

Figure 3: Paul Ekman's six core emotions

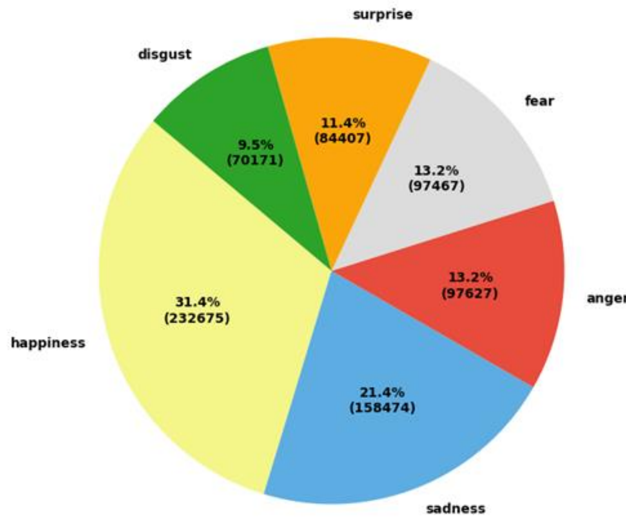

Figure 4: The composition of the final dataset

	sentence	emotion
0	That game hurt.	sadness
1	Man I love reddit.	happiness
2	Right? Considering its such an important docum...	happiness
3	He isn't as big, but he's still quite popular....	disgust
4	That's crazy; I went to a super [RELIGION] hig...	happiness
...	...	...
745723	My heart falters at the realization of my limi...	fear
745724	The unease over my own capabilities falling sh...	fear
745725	I'm jittery with apprehension as I confront my...	fear
745726	Oh no, the fear of my limitations hindering me...	fear
745727	The realization that my limitations might defi...	fear

740821 rows × 2 columns

Figure 5: The final dataset

```python
# Establishing early stopping to mitigate overfitting
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    verbose=1,
    restore_best_weights=True
)

# Initiating model training with early stopping and learning rate scheduler callbacks
history = model.fit(
    training_dataset,
    validation_data=validation_dataset,
    epochs=EPOCHS,
    callbacks=[early_stopping, warmup_lr_scheduler],
    verbose=1
)
```

Figure 6: Early Stopping Callback

```python
# Defining a warm-up learning rate scheduler to improve training.
class WarmUpLearningRateScheduler(Callback):
    def __init__(self, warmup_epochs, initial_lr, final_lr):
        super(WarmUpLearningRateScheduler, self).__init__()
        self.warmup_epochs = warmup_epochs
        self.initial_lr = initial_lr
        self.final_lr = final_lr
        self.increment = (self.final_lr - self.initial_lr) / self.warmup_epochs

    def on_epoch_end(self, epoch, logs=None):
        # Skip if outside the warmup period
        if epoch > self.warmup_epochs:
            return

        # Calculate and set the new learning rate
        new_lr = self.initial_lr + (epoch * self.increment)
        self.model.optimizer.lr = new_lr
        print(f"\nEpoch {epoch+1}, LR = {new_lr:.2e}")

# Usage:
warmup_epochs = 5
initial_lr = 1e-5
final_lr = 5e-5

warmup_lr_scheduler = WarmUpLearningRateScheduler(warmup_epochs, initial_lr, final_lr)
```

Figure 7: Learning Rate Scheduler

Fear, Amusement, Optimism, Anticipation, Disappointment	3, 2, 1, go! As team Orange we immediately have a strategy. That is to get me into the tower as soon as possible. I hold on to the legs of Ferry. And that went very hard. With one arm I paddle along, with the other arm I hold it. I'm not the first to get to the rope. But I have the first to get up at the rope. So that's where we're in for. And I'm actually going in a very nice elevator. Almost like a showgirl I just go up. Before I know it, I'm standing on top of the tower. Go! Go down! Help me! There goes Jeeva for 10 horses up. I can go up first. Then we agreed that Rose would go up. Because she's also light. And then we throw the rope down together. And I have taken the lead not to look at the other teams. I'm just doing what I have to do. Ready? Go! Lisa has her first button loose and throws it straight down. We go quite smoothly. We help to keep the net a bit tight. For the first few climbers that really makes it much easier. But then they are also faster up. What the fuck are these buttons? We have to loosen 12 buttons. But I don't see the buttons. And I don't understand what they mean. And in the meantime, Kirsten is also being lifted up. And she comes up and I help her up. I say, Kirsten, I don't see it. I thought, here we go. It's my first thing I have to do. And I don't see it right away. Then we stand at that net. Just throw those ropes off, I think. Because in the meantime, I see next to me. That they have that rope ladder. That they already have that down. No, I don't want that. Okay guys, I'm going to throw it down. Team purple has the rope down. And also team green has the rope down. Come on! Team orange is up. Well, for your information. Everyone is up there. And we can start at the lock. Climbing against that net, I found that very difficult. Yes, and then I got a hand from London. And also one from Niek. But Niek slipped. I thought, oh, I'll fall over later. Hold on tight. Hold on tight. Come on. Yes. That went just right. Orange releases the bar. First bar in the sea. And also the second bar of team orange is in the water. And while we have to jump, I think. Oh, I really don't dare to do this. And everyone is jumping. I think, okay, go. And I jump into it. Well, I survive. Thiijs is the first to set a foot on the water. And I grab that pole and I see that I'm first on the beach. So I'm going to do the left pole into the box first. And we had agreed, half, half. Half and half, man and woman, we're going to lift. Come on expedition members! Then we came to the second obstacle. A kind of net of all ropes, but we have to pull a box of I don't know how many kilos through it. And I can already see that the bottom left is open. So of the eight ropes, six ropes are already open on the left and then you just have to go a little bit. All teams start at the rope net. We had agreed that the men were going to carry everything and that the women were going to hold all the ropes well. And actually that went very smoothly. The orange on top. I see you, we have a good team, we have a good team, very nice, really super happy with it. It goes so fast, you know. You look left, you look right and I see that we are really ahead. Or no, we still have a lot of time.

Figure 8: Banijay labeled data with one or multiple emotions to multiple sentences