# NLP insights in TV Programming

Neil Ross Daniel (221270)

Vincent van der Beek (223377)

Maikel Boezer (220755)

Shan Oomes (220231)

2023-24 Y2C

Banijay Benelux

12/4/2024

## Contents

## Introduction

Natural Language Processing (NLP) is an interdisciplinary field that merges linguistics, computer science, and artificial intelligence (AI) to decode human language for diverse applications. Among these applications, emotion classification presents a big challenge due to the contextual nuances and subjective interpretations in emotional expression. Paul Ekman's seminal research in the 1970s identified six core emotions—happiness, sadness, fear, anger, surprise, and disgust—as foundational to human experience. These emotions, along with their related secondary and tertiary counterparts, provide a comprehensive framework for annotating and analyzing emotional content within text.

Our collaboration with Banijay Benelux and 3Rivers media consultants focuses on enhancing emotion classification within television programming, with a particular emphasis on the popular TV series "Expeditie Robinson" (Survivor). By manually annotating episodes, our goal is to train models capable of automatically tagging emotional expressions, thereby facilitating deeper insights into viewer engagement and content analysis.

The dataset from “Expeditie Robinson” has information on various TV shows, including details on segments like explanations, games, and voting sessions, along with when each segment aired and viewer metrics such as Kdh and Madl, indicating viewer demographics and popularity across different age groups. It also includes content specifics like locations, props, actors, emotions, and types of games played, enabling a deeper understanding of show themes and storytelling.

Throughout this report, we will discuss various NLP methods that we have tried for emotion classification, emphasizing the significance of robust training data for model development and evaluation. By leveraging theoretical insights alongside practical implementations, we aim to unlock new avenues for understanding and interpreting emotional cues embedded within textual data.

## Data Processing and Exploration

The training dataset contains the sentences and their corresponding emotions from GoEmotions. SMILE, MELD-master, Friends emotion-labeled dialogues, ijcnlp_dailydialog and the CARER dataset. These datasets are sampled for various sources like Reddit comments, tweets, and dialogues . For these datasets we extracted features that we thought would help with the accuracy of the model. The features extracted from the sentences include:

- Document Entities: Extracted named entities present in the document.
- Document Noun Chunks: Identified noun phrases within the document.
- Document 2-Grams: Pairs of consecutive words occurring in the document.
- Document 3-Grams: Triplets of consecutive words occurring in the document.
- Token Part of Speech: Part-of-speech tags assigned to each token.
- Token Lemmatized: Lemmatized forms of each token.
- Token Normalized: Normalized versions of each token.
- Token Dependency: Dependency relationships between tokens.
- Token Sentiment: Sentiment scores or labels associated with each token. (This was done using the spaCy: doc._.blob.polarity)

The Test dataset contains detailed information about different TV shows, including their names, episodes, and parts like acts and chapters. These parts cover various segments such as explanations, games, and voting sessions. It also includes details about when each segment aired on live TV. Viewer metrics like Kdh (Kijkdichtheid in % is the percentage of people in the Netherlands in that age group that were watching) and Madl (Marktaandeel, reflects the amount of people in that age group divided by the total amount of people that was watching TV at that moment for any given channel in total) are provided, showing how many people watched each segment and their demographics. Analyzing these metrics helps us see which parts of the shows are popular with viewers of different ages. The dataset also tells us about the content of each segment, like where it takes place, props used, actors involved, emotions shown, and types of games played.

Exploring this content helps us understand the themes and stories of the shows better. We can also look at how locations, props, and games are used across segments to get an idea of what viewers enjoy. Plus, analyzing how viewer metrics change throughout each episode helps us see trends in audience engagement over time. Overall, by looking at this data closely, we can learn a lot about what makes TV shows appealing to viewers and how their content is structured. This information can help TV creators make better decisions about what to include in their shows and when to air them.

The data is manually labeled and annotated by taggers based on observations while watching the episodes and the viewership metrics are provided by the NMO.
The Reliability of the sources can vary. The viewership metrics provided by the NMO are dependable, while annotations by taggers may have subjectivity based on their interpretations.

To understand the data, we started by analyzing the viewership patterns across different age groups to understand which demographics are more engaged with the show. After analyzing the data, we found that the age group with the highest viewership was 35- 49 years. As this group had the highest viewership, we decided to further research this group.

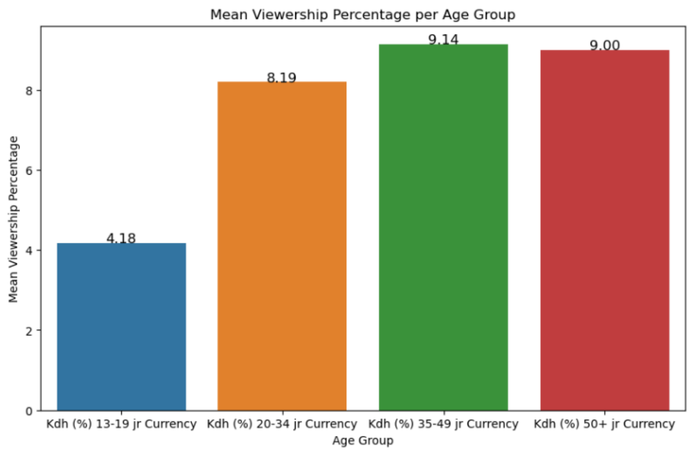


*Figure 1 Mean Viewership percentage per age group.*

We then did a content analysis to explore the types of locations, emotions and game types that are associated with higher viewership percentages. This could help in understanding viewer preferences and interests.

Doing the content Analysis, we found that the location of the fragment with the highest viewership percentage was “Thuis” (at home) and the lowest was “Duel arena afvallerseiland.”

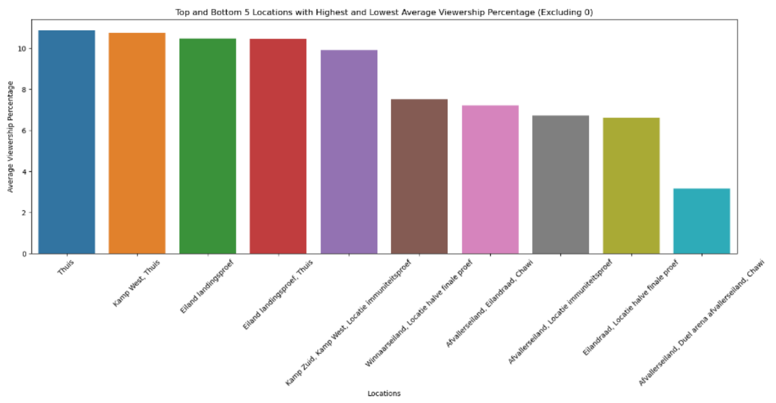


*Figure 2*

The emotion with the highest viewership percentage were “Disgust, Amusement” and the lowest were “Surprise, Amusement, Confusion.”

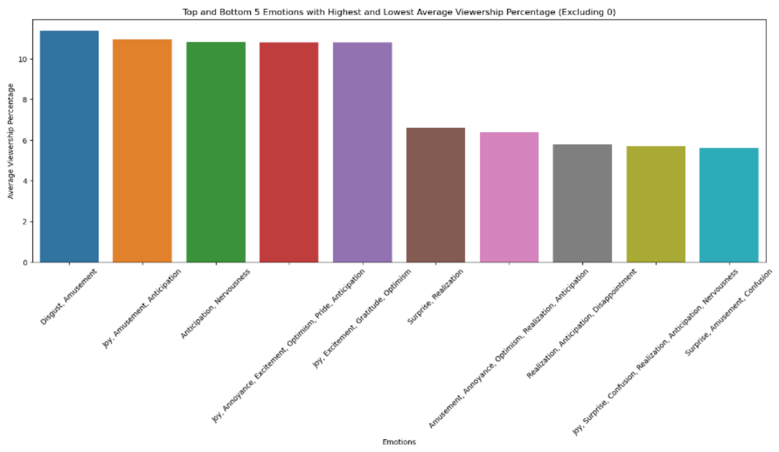


*Figure 3*

The type of game with the highest viewership percentage were “cooperative games, endurance games, puzzles” while the lowest was “skill games, board games, strategy games.”

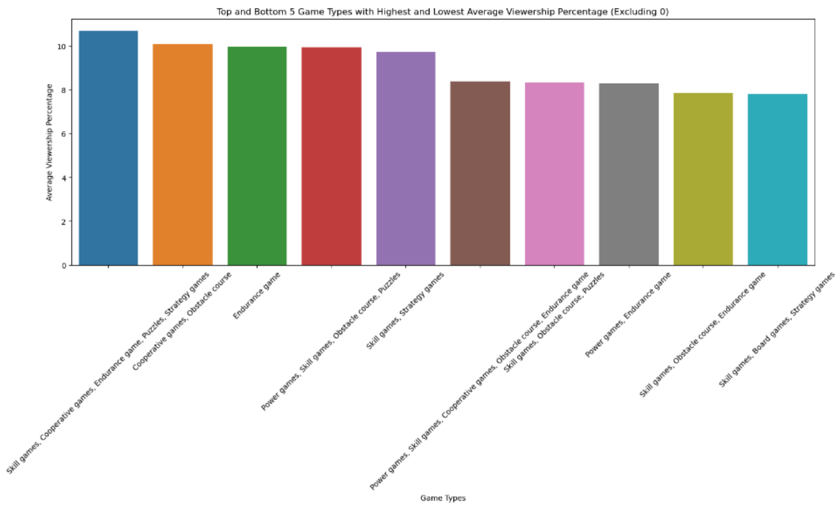


*Figure 4*

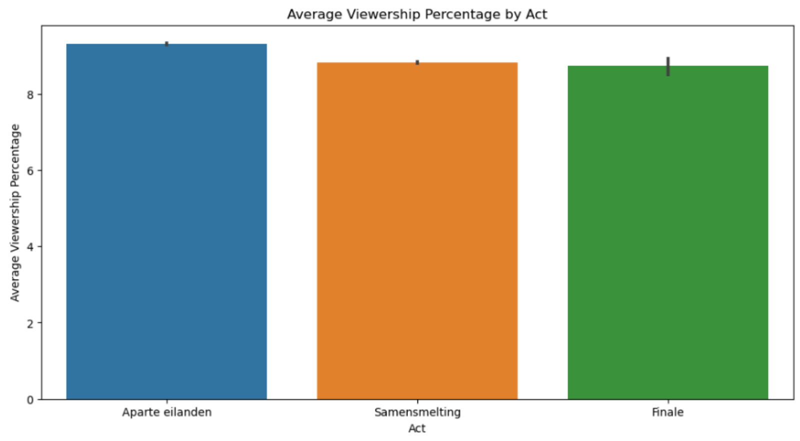


*Figure 5*

Finally, we compared viewership metrics between different acts, chapters, and segments to identify patterns and trends.

Doing the comparison we found that as the viewership declined as the acts progressed.

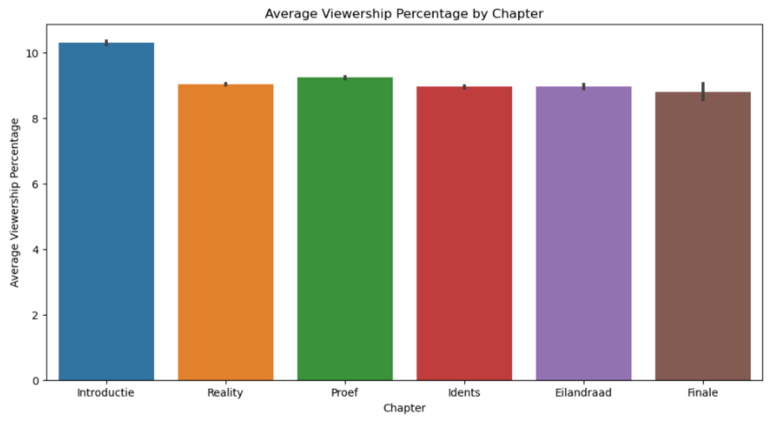


*Figure 6*

The viewership is highest at the "introduction" chapter but drops during the "reality" chapter, increases slightly at the "proef" (Games) chapter then gradually declines for the following chapters.

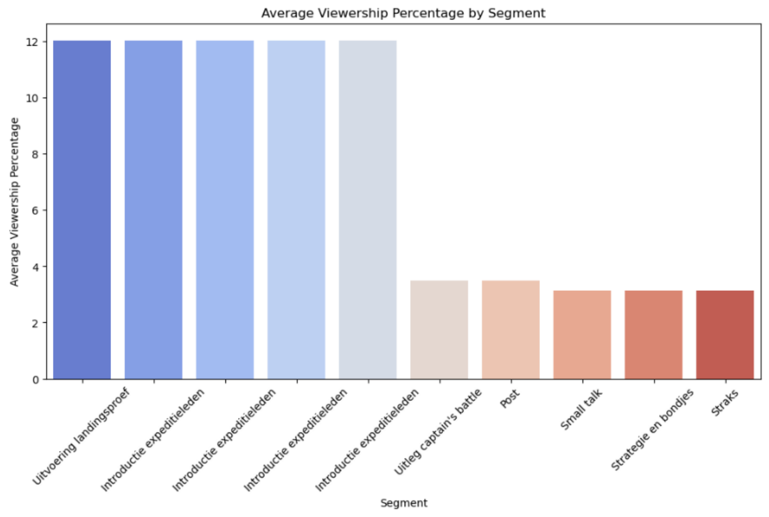


*Figure 7*

The viewership is highest is in the "uitvoering landingsproef" segment and is lowest in the "straks" segment.

## Preprocessing and Feature Engineering

When working with a lot of data and converting it to sentences we could use for emotion detection, we must use some case of preprocessing for the training dataset. All of us tried different things, but the most common ones that we used for most models were the following.

**Text cleaning**, text often has either exceptionally large or has noise inside it. Words or symbols that can be removed and it would allow the training process to be easier for the model. Removing special characters, punctuation marks or irrelevant symbols are a key part of the preprocess.

**Tokenization** is like breaking a sentence into smaller pieces. Instead of looking at the whole sentence at once, we split it into words or units. This helps us understand each part of the sentence separately, making it easier to study and analyze.

**Stemming**, A technique employed to reduce words to their root form, standardizing variations and reducing redundancy within the dataset. By condensing words to their base form, stemming enhances the efficiency of subsequent analyses, ensuring consistency across the dataset. It also made it easier for models to identify the data more clearly.

**Vectorization**, with different models you would have to use more preprocessing than with others. We used vectorization for the neural models that we have done like RNN, HMM, LSTM and CRF. Using TF-IDF or TensorFlow vectorizer we used this method to enable our neural models by converting textual data to numerical. This is to take in the data and to provide a good training set for our results.

**Handling imbalanced data**, when acquiring data from various sources. It became evident that a lot more of a certain emotion came back. Happiness and sadness became the number one and two most common emotions. Them being about 70% of the data. The least was “disgust,” there were about 6000 sentences total what would account for barely 1% of the total dataset. To solve this taking an x amount of each emotion instead of the whole dataset would alleviate some of the bias in data.

Reprocessing methods vary based on the needs of different models used for emotion detection in sentences. Below, we will explore simplified explanations of preprocessing techniques tailored to several types of models.

**Traditional Machine Learning Models**:

Preprocessing: Basic steps include breaking text into words, making everything lowercase, and removing unnecessary words and punctuation. Sometimes, words are trimmed to their base form. Vectorization: Text becomes numbers using techniques like CountVectorizer or TF-IDFVectorizer. This converts words into numerical values based on how often they appear or their importance in the text. Feature Engineering: Extra features such as word combinations, sentiment scores, or word types like nouns or verbs can be added to improve understanding.

**Deep Learning Models:**

Preprocessing: Text is turned into fixed-length sequences. Padding or trimming ensures they are all the same length. Words are usually made lowercase, and unimportant words may be removed. Vectorization: Text becomes dense numbers using methods like Word2Vec or GloVe. These methods convert words into numbers that represent their meaning in a numerical space. Feature Engineering: Features like word types or relationships between words can be added to give more context to the model.

**Transformer Models (e.g., RoBERTa) :**

These models have shown impressive performance: Preprocessing: Text is broken into smaller parts using special methods like WordPiece or Byte Pair Encoding. Sequences are then made uniform in length through padding or trimming. Vectorization: Text is turned into numbers using pre-trained embeddings that capture complex relationships between words. Feature Engineering: These models often do not need extra features because they are good at understanding context. But sometimes, extra features like positional information are added to help in specific tasks.

## Model Selection and Implementation

### Model Selection

After preparing a dataset and deciding what features we were going to use in our models, we all started making models. We started with making simple models for binary classification or "statistical models" such as naive Bayes and logistic regression. Logistic regression can become a lot more complicated it can even (also) be used for multi-class classification, but we kept it simple (for binary classification). In this early stage of making models, it was interesting to see that for most people in the group, their naive Bayes models were outperforming the logistic regression models on the test data we got from the individual Kaggle competition. The difference was minimal, but the simpler naive Bayes model performed slightly better than the more complex, logistic regression model was unexpected. There was one in our group where the logistic regression model worked better than naïve Bayes and that is because his logistic regression model was a lot more complicated compared to the rest of the group.

Next, we started working on more complicated models this time for multiclass classification or "neural models." The models we tried are a Hidden Markov Model (HMM), a Conditional Random Fields (CRF) model, a Recurrent Neural Network (RNN) model, and a Long Short-Term Memory (LSTM) model. The CRF model and the HMM model were difficult to get working so we decided to not finish them and put our effort toward the remaining models. An LSTM model is a form of RNN model , and it combats an RNN's long-term dependence issue, what this means is that when a normal RNN model is making a prediction on Figure 8 it looks at the text around it to make a prediction, but the text around it does not help with making a prediction. What LSTM does is cut the sentence down to the critical data which is " ______ down my budget" and forget the rest so now it only must predict on " ______ down my budget" which makes it easier to produce a result like "Cut down the budget." So, in theory, an LSTM model will work better than an RNN model, but on the team Kaggle test data the RNN work better than the LSTM this has something to do with the structure of the neural networks as this has an excessively substantial impact on the performance of a neural network. However, the score of our best RNN model was not one we were satisfied with.

"I am going to buy a table that is large in size, it'll cost more, which means I have to ______ down my budget for the chair," (Mohak, 2022).

Figure 8

We attempted to improve our score with word embedding models both pre-trained and making one ourselves. A pre-trained model has already been trained on a large dataset, which means that when you train your model it does not have to start from zero. To avoid confusion when talking about a word embedding model, we are not talking about statistical or neural models that can predict an output. Word embedding models are used to give more meaning to the words in your input data you can visualize it like in Figure 8. The performances of these models were worlds apart, the word embedding model that we trained ourselves performed very poorly not just in comparison to the pre-trained but in general. In comparison, the pre-trained word embedding model performed the best so far, but we were still not satisfied.

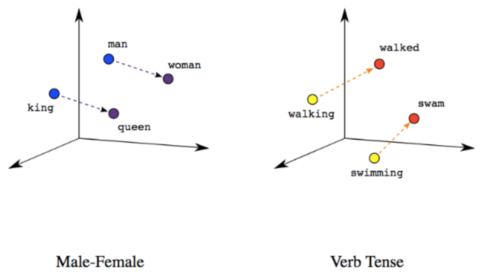


*Figure 8*

The last thing we had to try was a transformer model, more specifically a pre-trained BERT model and a different version of BERT, RoBERTa. These transformer models have performed the best out of all the models we have tried, with the RoBERTa model slightly outperforming the BERT models. The model we selected for our final report is the RoBERTa model and for more details about the performance of this model check the Evaluation Metrics and Results section.

The architecture

RoBERTa, an acronym for "Robustly Optimized BERT Pre-training Approach," represents a significant advancement in the domain of natural language processing (NLP). Developed by researchers at Facebook AI and Washington University, RoBERTa stands as a variant of the renowned BERT (Bidirectional Encoder Representations from Transformers) model. While sharing the foundational principles of BERT, RoBERTa distinguishes itself through a series of enhancements and optimizations, resulting in superior performance across various NLP tasks.

RoBERTa, an evolution of BERT, operates on the transformative architecture of transformers, which revolutionizes natural language processing through its unique mechanisms. At its core, transformers employ a self-attention mechanism, allowing each word in a sequence to weigh its importance against every other word, enabling the model to capture intricate contextual relationships. Multi-head attention further enhances this capability by enabling simultaneous attention to different parts of the input, facilitating a holistic understanding of the relationships between words, and capturing the broader context within sentences. Additionally, transformers utilize positional encoding to provide crucial information about the sequential order of words, ensuring that the model can discern temporal dynamics and capture long-range dependencies. Together, these features allow RoBERTa to excel in tasks ranging from language modeling to translation by comprehensively understanding the nuanced semantics and relationships within natural language sequences.

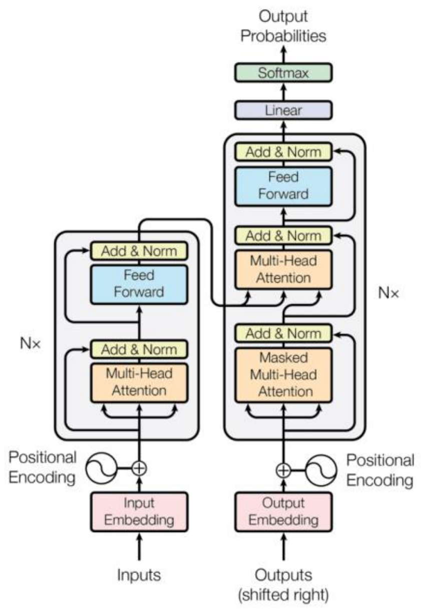


*Figure 9*

## Implementation

After selecting what model, we were going to use. We started working on the rest of the pipeline. The purpose of the pipeline is to be able to feed it an episode of Expedition Robinson and give a prediction of emotion that each sentence has in the episode. The first step in this process is to split up the episodes into fragments that we can deal with in this case, into sentences. We do this by using the start and end time data of each fragment we got with the episodes and then saving the audio of that fragment for speech-to-text later. A limitation of doing it this way, is that this pipeline cannot be used with episodes that we do not have data for.

The second step in this process is to run the fragment through a speech-to-text model because our model does not work with audio. The speech-to-text model we used is called whisper which is a general-purpose automatic speech recognition model. While the fragments were being transcribed, they were also being translated from Dutch to English. After being transcribed and translated the data is stored in a panda's data frame and saved as a CSV file so that this step can be skipped the next time it is run.

The third step is to predict what emotion is present in the transcript but before that, the data needs to be tokenized so that the model will accept it. For this, we used a tokenizer from the same library as the model, a RoBERTa tokenizer. Then import the model and let it predict on the tokenized data and save the predictions to the data frame.

The fourth and last step is to put everything together, including checks to see if the fragments already exist and a check to see if the transcript file already exists. These checks are there to make sure that the pipeline runs efficiently and does not do everything again if a prediction is made with a different model. The entire pipeline can be seen in Figure 10.

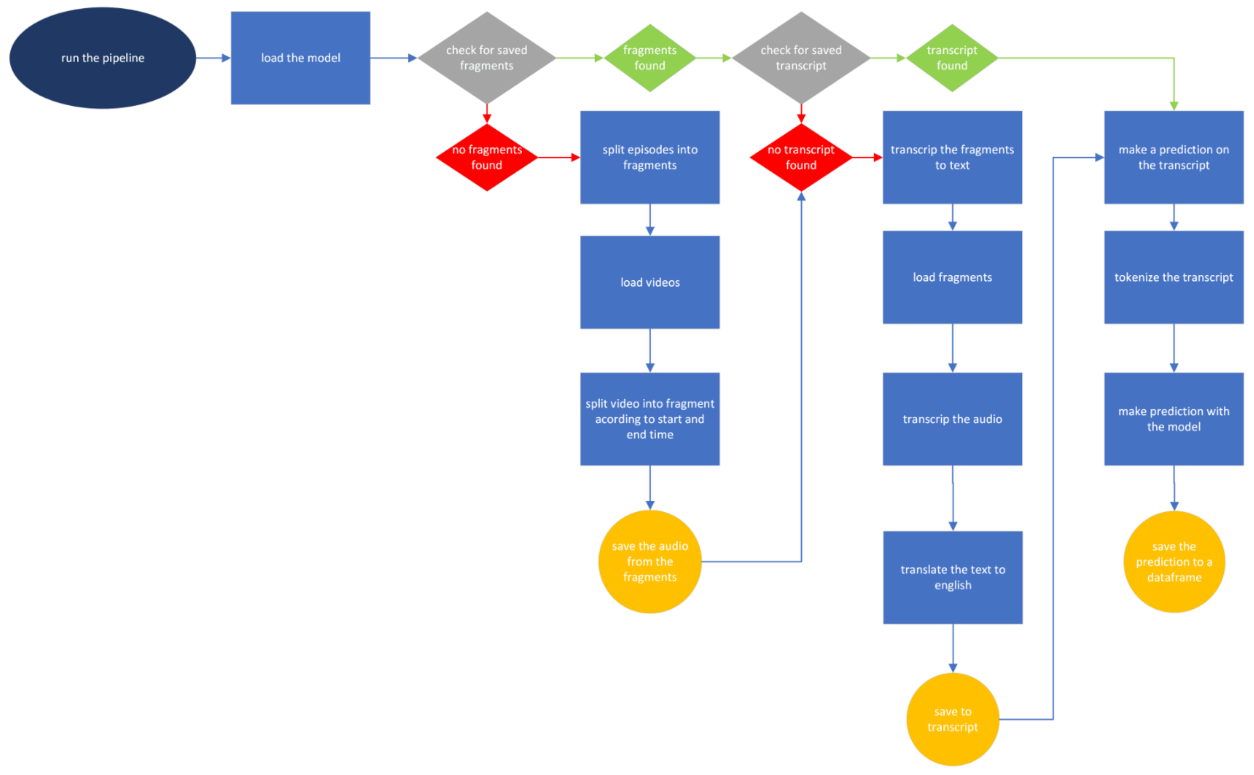

run the pipeline
load the model
check for saved fragments
fragments found
no fragments found
split episodes into fragments
load videos
split video into fragment acording to start and end time
save the audio from the fragments
check for saved transcript
transcript found
no transcript found
transcrip the fragments to text
load fragments
transcrip the audio
translate the text to english
save to transcript
make a prediction on the transcript
tokenize the transcript
make prediction with the model
save the prediction to a dataframe

*Figure 10*

## Evaluation Metrics and Results

These evaluation metrics represent key measures chosen to assess the performance of the model using the test or validation dataset. They provide valuable insights into the model's effectiveness across different facets of prediction accuracy. Here is a breakdown of the metrics and their interpretation:

Accuracy: The model achieves an accuracy of 0.5597, indicating that approximately 55.97% of the predictions made by the model are correct. While accuracy offers a broad view of the model's correctness, it may not fully capture nuances in performance, particularly in scenarios with imbalanced datasets.

Precision: With a precision of 0.5581, the model demonstrates the proportion of correctly predicted positive cases among all instances predicted as positive. This implies that around 55.81% of the instances classified as positive by the model are true positives, reflecting a low rate of false positives.

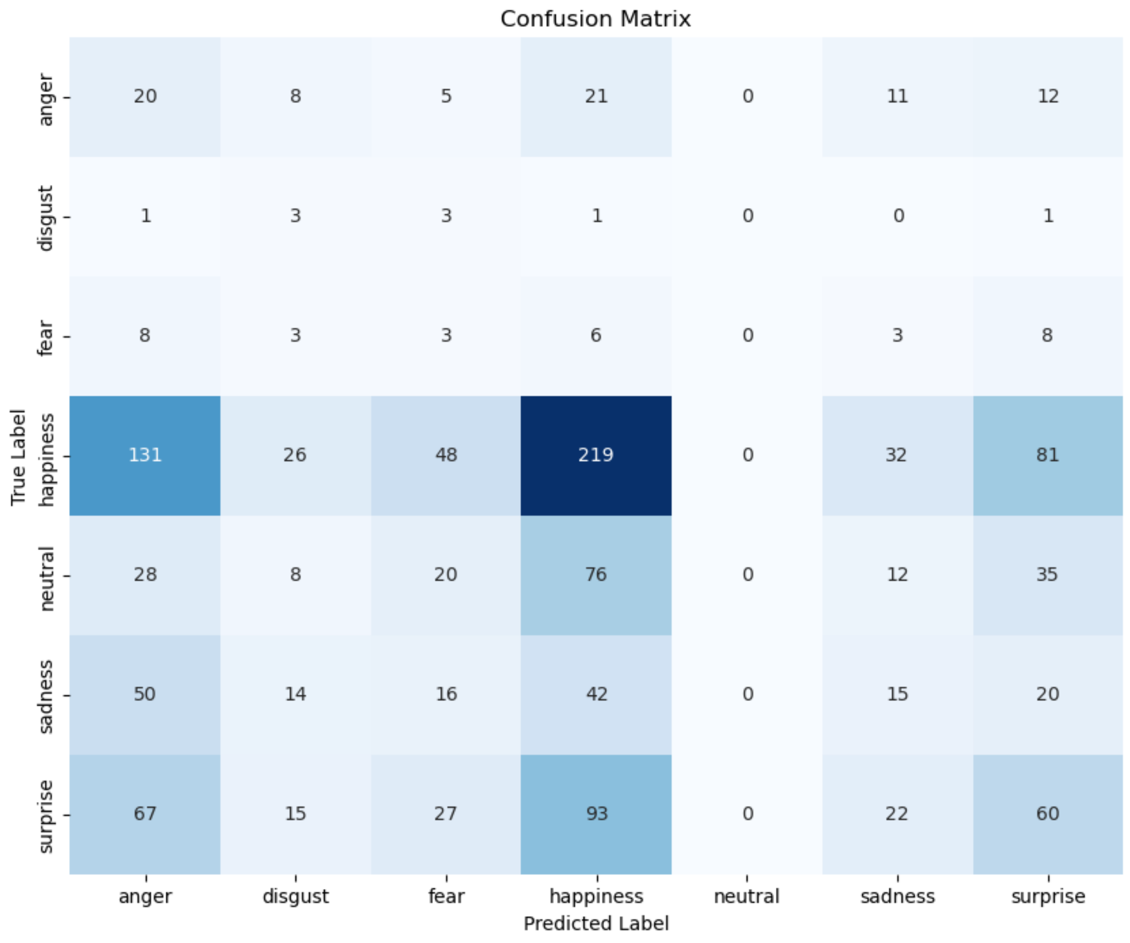
Recall: The recall metric, also equal to the accuracy value, suggests that approximately 55.97% of the actual positive instances in the dataset are correctly identified by the model. This indicates the model's ability to capture a sizable portion of positive instances, thereby minimizing false negatives.

F1 Score: The F1 score, computed as 0.5573, represents the harmonic mean of precision and recall, providing a balanced assessment of the model's performance. A higher F1 score indicates better overall performance, considering both false positives and false negatives.

## Results

Now that the pipeline is finished and capable of predicting the emotions within fragments of the TV show "Survivor" (Expedition Robinson), we can examine the results. Here, we conducted an error analysis comparing the actual labels with the predicted ones.

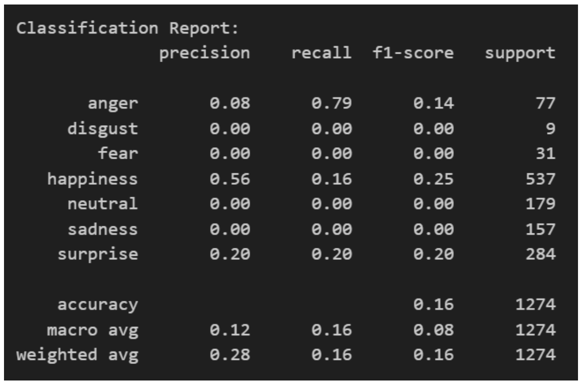
### Error analysis



| True Label \ Predicted Label | anger | disgust | fear | happiness | neutral | sadness | surprise |
|---|---|---|---|---|---|---|---|
| anger | 20 | 8 | 5 | 21 | 0 | 11 | 12 |
| disgust | 1 | 3 | 3 | 1 | 0 | 0 | 1 |
| fear | 8 | 3 | 3 | 6 | 0 | 3 | 8 |
| happiness | 131 | 26 | 48 | 219 | 0 | 32 | 81 |
| neutral | 28 | 8 | 20 | 76 | 0 | 12 | 35 |
| sadness | 50 | 14 | 16 | 42 | 0 | 15 | 20 |
| surprise | 67 | 15 | 27 | 93 | 0 | 22 | 60 |

*Figure 11*

Figure 11 shows us the confusion matrix, this was part of the error analysis. Here we examine the mistakes made by our best performing RoBERTa model, gaining insights into its performance and areas of improvements. It shows the number of true positives, true negatives, false positives, and false negatives for each class.



```
Classification Report:
              precision    recall  f1-score   support

       anger       0.08      0.79      0.14        77
     disgust       0.00      0.00      0.00         9
        fear       0.00      0.00      0.00        31
   happiness       0.56      0.16      0.25       537
     neutral       0.00      0.00      0.00       179
     sadness       0.00      0.00      0.00       157
    surprise       0.20      0.20      0.20       284

    accuracy                           0.16      1274
   macro avg       0.12      0.16      0.08      1274
weighted avg       0.28      0.16      0.16      1274
```

*Figure 12*

In Figure 12, the classification report presents a detailed breakdown of the model's performance metrics for each emotion category. Notably, the precision and recall scores for emotions such as Anger, Disgust, Fear, Sadness, and Surprise indicate significant difficulty in accurately predicting these states. Conversely, the model demonstrates better precision and recall for the Happiness class. However, an alarming observation arises from the complete failure to correctly identify instances of the Neutral class, as evidenced by zero precision, and recall scores. Overall, these findings underscore the urgent need for improvements to enhance the model's ability for more precise and reliable emotion classification.

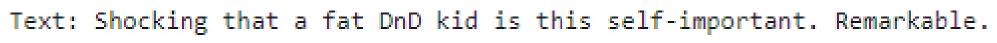
Here below we look at some sample predictions, reviewing individual predictions made by the model. Allowing to understand how the model performs on different types of inputs.

**The model used.**

The RoBERTa (Robustly Optimized BERT Pre-training Approach) Transformer model is a state-of-the-art natural language processing (NLP) model. It utilizes the Transformer architecture and pre-training techniques on large corpora to learn powerful language representations.

**Successful Classifications:**

Happiness:

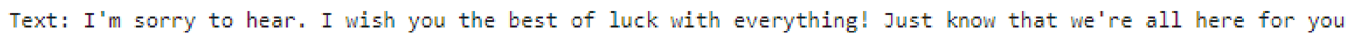


```
Text: Shocking that a fat DnD kid is this self-important. Remarkable.
```

*Figure 13*

*Analysis: The model adeptly identifies the expression of happiness, highlighting its proficiency in capturing positive emotional states.*

Sadness:



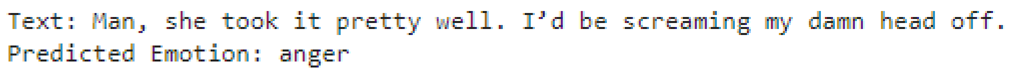
```
Text: I'm sorry to hear. I wish you the best of luck with everything! Just know that we're all here for you
```

*Figure 14*

*Analysis: Despite the unconventional expression, the model accurately discerns the underlying sadness, highlighting its ability to capture nuanced emotional nuances.*

**Challenges and Patterns:**

***Surprise:***

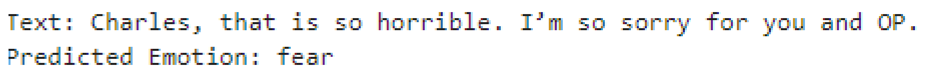


```
Text: Man, she took it pretty well. I'd be screaming my damn head off.
Predicted Emotion: anger
```

*Figure 15*

*Analysis: The model frequently misclassifies surprise as fear, due to the presence of fear-associated words like "screaming". This highlights the challenge in distinguishing surprise from other negative emotions.*

***Sadness:***



```
Text: Charles, that is so horrible. I'm so sorry for you and OP.
Predicted Emotion: fear
```

*Figure 16*

*Analysis: The model might have associated words like "horrible" and "sorry" with fear-inducing situations, leading to a prediction of fear instead of 'other.' It may have interpreted the statement as expressing concern or empathy towards a potentially fearful event.*

## Strengths and Limitations

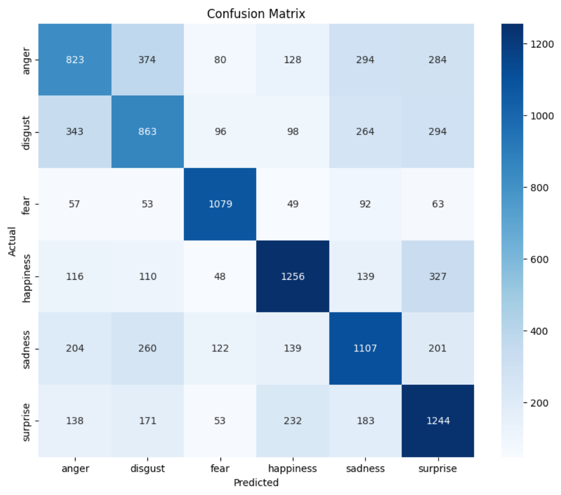
RoBERTa, a variant of the BERT (Bidirectional Encoder Representations from Transformers) model, exhibits strengths in capturing contextual information from text due to its bidirectional nature. With pre-training which considers both left and right contexts, RoBERTa can acquire an improved understanding of language semantics and relationships. This enables the model to perform well on a wide range of natural language processing tasks, including sentiment analysis and language understanding.

On the contrary, in terms of training time and computational resources, RoBERTa's strength might also be its weakness. The bidirectional nature of RoBERTa requires extensive pre-training on large datasets, leading to longer training times and higher computational costs compared to unidirectional models. In addition, RoBERTa's huge size could make implementation difficult, particularly in settings with limited resources. Despite these drawbacks, RoBERTa is a useful tool for a variety of natural language processing tasks. When computing resources allow it, then its effectiveness in gathering contextual information is strong.

## Discussion

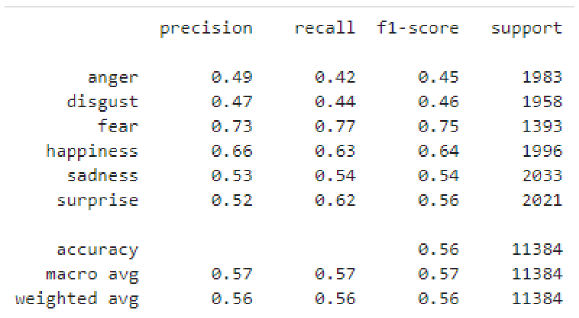
### Error Analysis of RoBERTa Model Predictions for Emotion Classification

Examining the test data reveals challenges in emotion classification. Common difficulties include distinguishing subtle emotional nuances, leading to misinterpretations. The model struggles with sarcasm and irony, especially for emotions like disgust and surprise. Biases in training data also impact predictions, leading to inaccuracies based on stereotypes. Addressing these challenges is vital for enhancing the model's accuracy and reliability in emotion classification tasks. Below you can see a confusion matrix and a classification report of the predictions.



| Actual \ Predicted | anger | disgust | fear | happiness | sadness | surprise |
|---|---|---|---|---|---|---|
| anger | 823 | 374 | 80 | 128 | 294 | 284 |
| disgust | 343 | 863 | 96 | 98 | 264 | 294 |
| fear | 57 | 53 | 1079 | 49 | 92 | 63 |
| happiness | 116 | 110 | 48 | 1256 | 139 | 327 |
| sadness | 204 | 260 | 122 | 139 | 1107 | 201 |
| surprise | 138 | 171 | 53 | 232 | 183 | 1244 |

*Figure 18*



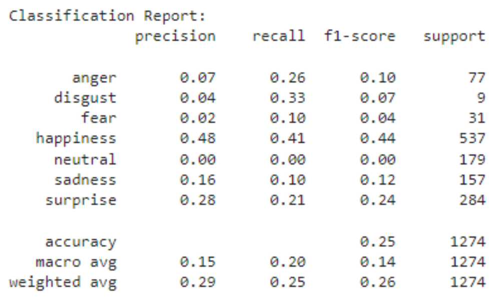
```
              precision    recall  f1-score   support

       anger       0.49      0.42      0.45      1983
     disgust       0.47      0.44      0.46      1958
        fear       0.73      0.77      0.75      1393
   happiness       0.66      0.63      0.64      1996
     sadness       0.53      0.54      0.54      2033
    surprise       0.52      0.62      0.56      2021

    accuracy                           0.56     11384
   macro avg       0.57      0.57      0.57     11384
weighted avg       0.56      0.56      0.56     11384
```

*Figure 17*

The precision, recall, and F1-score vary across different emotions in the training model. Notably, fear shows the highest precision at 0.73, indicating the model's accuracy when predicting instances of fear. Alternatively, other emotions such as anger, disgust, sadness, and surprise display lower precision values, suggesting a higher likelihood of false positives.

Similarly, recall values vary, with fear showing the highest recall at 0.77, while happiness, sadness, and surprise also show higher recall rates compared to other emotions. The support column underscores a class imbalance within the dataset, with disparities in sample sizes across emotions, potentially impacting the model's performance, particularly for minority classes. Despite an accuracy of 0.56, reflecting better performance than random guessing, the macro and weighted average F1-scores remain low at 0.57, indicating challenges in generalizing across all classes and limitations in the model's overall performance.

***Implementation in transcript:***



```
Classification Report:
              precision    recall  f1-score   support

       anger       0.07      0.26      0.10        77
     disgust       0.04      0.33      0.07         9
        fear       0.02      0.10      0.04        31
   happiness       0.48      0.41      0.44       537
     neutral       0.00      0.00      0.00       179
     sadness       0.16      0.10      0.12       157
    surprise       0.28      0.21      0.24       284

    accuracy                           0.25      1274
   macro avg       0.15      0.20      0.14      1274
weighted avg       0.29      0.25      0.26      1274
```

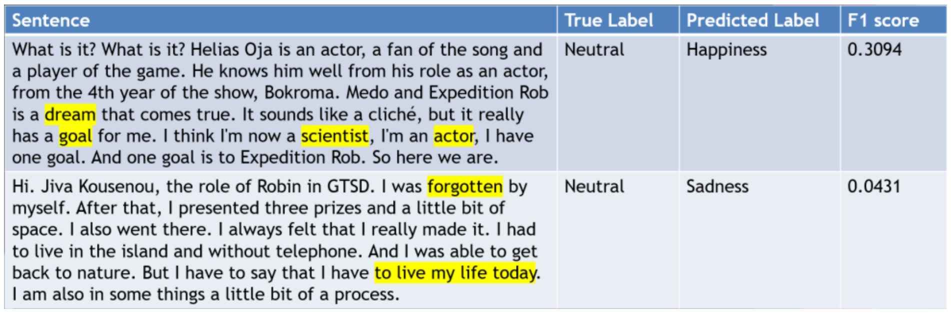
*Figure 19*

The precision, recall, and F1-score vary across different emotions. For example, the precision for happiness is high at 0.48, indicating that when the model predicts happiness. However, the precision for other emotions like anger, disgust, fear, and sadness is incredibly low, ranging from 0.02 to 0.16, suggesting a high rate of false positives.

The recall values also vary, indicating how well the model captures instances of each emotion from the dataset. The recall for happiness, sadness, and surprise is higher compared to other emotions.

The support column indicates the number of samples for each class in the dataset. The dataset is imbalanced, with a significantly higher number of samples for happiness and a lower number for disgust, fear, and sadness. This class imbalance could affect the model's performance, especially for minority classes.

The accuracy of the model is 0.25. While this accuracy is better than random guessing, it is low, suggesting that the model's performance is limited. The macro and weighted average F1-scores are both low at 0.14 and 0.26, respectively. This indicates that the model's performance is poor across all classes, and it struggles to generalize well to unseen data. To further analyze the results, we will examine two predictions with the lowest F1 score.



| Sentence | True Label | Predicted Label | F1 score |
|---|---|---|---|
| What is it? What is it? Helias Oja is an actor, a fan of the song and a player of the game. He knows him well from his role as an actor, from the 4th year of the show, Bokroma. Medo and Expedition Rob is a dream that comes true. It sounds like a cliché, but it really has a goal for me. I think I'm now a scientist, I'm an actor, I have one goal. And one goal is to Expedition Rob. So here we are. | Neutral | Happiness | 0.3094 |
| Hi. Jiva Kousenou, the role of Robin in GTSD. I was forgotten by myself. After that, I presented three prizes and a little bit of space. I also went there. I always felt that I really made it. I had to live in the island and without telephone. And I was able to get back to nature. But I have to say that I have to live my life today. I am also in some things a little bit of a process. | Neutral | Sadness | 0.0431 |

*Figure 20*

In prediction one it seems like the content of the sentence involves someone talking about their aspirations and goals (becoming an actor, participating in a game show). The presence of positive words like "dream," "goal," and "scientist" might have led the model to predict happiness.

Sentence two starts with a greeting followed by a statement about being forgotten by oneself, which could indicate Self-reflection or a sense of being overlooked. The model might have interpreted this self-reflective tone as indicative of sadness. Additionally, words like "forgotten" and "live my life today" might have contributed to the prediction of sadness.

## Conclusion

In conclusion, our project aimed to enhance emotion classification within television programming, focusing on the popular TV series "Expeditie Robinson" (Survivor). We employed a variety of NLP methods and models, selecting the RoBERTa transformer model for its superior performance. Our pipeline successfully predicts emotions within show fragments, providing valuable insights into viewer engagement and content analysis.

However, our evaluation revealed challenges in accurately classifying certain emotions, particularly those with subtle nuances like disgust and surprise. Addressing these challenges requires refining the model's training data, mitigating biases, and improving contextual understanding.

Future recommendations include expanding the dataset to include more diverse emotional expressions, refining preprocessing techniques to better handle imbalanced data, and exploring ensemble methods to improve classification accuracy. Additionally, incorporating multimodal features like audio and video data could further enhance emotion classification accuracy, offering a more comprehensive understanding of viewer engagement. Overall, our project lays a foundation for deeper insights into emotional cues within textual data, with potential applications across various domains beyond television programming.

## References

Mohak. (2022, June 9). *Recurrent Neural Networks (RNN) and LSTM: Overview and Uses*. https://www.turing.com/kb/recurrent-neural-networks-and-lstm

*NLTK Book*. (n.d.). https://www.nltk.org/book/

*Speech and language processing*. (n.d.). https://web.stanford.edu/~jurafsky/slp3/

*RoBERTa*. (z.d.). https://huggingface.co/docs/transformers/en/model_doc/roberta

*go_emotions · Datasets at Hugging Face*. (2023, 4 oktober). https://huggingface.co/datasets/go_emotions