YouTube Comments: MrBeast Classification

Gefei Zhao
Computer Science
undergraduate
gzhao46@wisc.edu

Abstract

This project aims to automatically classify user sentiment in MrBeast's YouTube comments, a setting where large-scale audience reactions are informative but impossible to manually analyze. We preprocess a labeled dataset of comments and evaluate three main model families. Our approach involves fine-tuning RoBERTa on the domain-specific dataset and training the CNN and MLP model from scratch for comparison. Experimental results show that while the off-the-shelf Twitter-based RoBERTa model performs moderately well, domain-specific fine-tuning dramatically improves accuracy to near-perfect levels, with TextCNN and embedding-based MLP offering a competitive, more efficient alternative. These findings highlight both the importance of adapting models to the YouTube comment domain and the trade-offs between accuracy and computational efficiency.

1 Introduction

User comments on large YouTube channels such as MrBeast's provide rich feedback about audience perception, satisfaction, and potential controversies. However, the volume of comments makes manual inspection infeasible. Automatically classifying comment sentiment into Negative, Neutral, and Positive categories can help summarize community reactions and monitor shifts in viewer sentiment over time.

The specific problem we address is: Given a text comment on a MrBeast video, predict its sentiment label. We use a labeled dataset of comments and sentiment annotations and explore both different architectures. This problem is interesting and important because sentiment on high-impact channels can influence brand perception, sponsorship decisions, and content strategy.

In this project, our goals are:

- To preprocess and clean the MrBeast sentiment dataset into a format suitable for modern NLP models.
- To fine-tune a large pretrained transformer model (RoBERTa) on this domain-specific data.
- To implement and evaluate a multi-kernel TextCNN and an embedding-based multilayer perceptron (MLP)—to study performance–efficiency trade-offs in this domain.

1.1 Related Work

Sentiment analysis of social media and user-generated content has been widely studied, with early work focusing on bag-of-words and classical machine learning models such as logistic regression and SVMs combined with n-gram features. More recent approaches employ deep learning, including convolutional neural networks for sentence classification and, more recently, large pretrained transformer models.

- Transformer-based models such as RoBERTa have achieved state-of-the-art performance on a wide range of NLP benchmarks by leveraging large-scale pretraining on diverse text corpora followed by task-specific fine-tuning.
- CNN-based architectures such as TextCNN provide a lighter alternative. They use multiple 1D convolutional filters over word embeddings to capture local n-gram features and have been shown to work well for short text classification tasks while being more computationally efficient than large transformers.
- Embedding-based MLP models serve as a strong baseline for text classification by combining fixed-length text representations with fully connected layers. While they lack explicit mechanisms for modeling word order or long-range dependencies, MLPs can perform competitively when paired with informative embeddings.

Unlike many generic sentiment analysis works, we specifically focus on a single creator's channel (MrBeast), which allows us to study performance on a narrower but highly relevant domain (short, informal, often meme-like comments).

2 Method

We use the Kaggle dataset `adilshamim8/mrbeast-youtube-comment-sentiment-analysis`:

- Comment — the raw YouTube comment text
- Sentiment — a categorical label in {Positive, Neutral, Negative}

2.1 Preprocessing Pipeline

2.1.1 Load and inspect data using `pandas`

- The dataset contains 6797 comments and sentiment labels.

2.1.2 Clean labels

- Strip leading/trailing whitespace from the `Sentiment` column to unify variants
- Keep only rows where the sentiment is one of the three valid classes.

2.1.3 Split into train/validation sets

- Training size = 5437
- Validation size = 1360

2.1.4 Label encoding

- For RoBERTa, we align our dataset labels with the model's built-in label space.
  - We load the configuration of `cardiffnlp/twitter-roberta-base-sentiment-latest` and read `config.id2label`, which for this checkpoint maps {0: "negative", 1: "neutral", 2: "positive"}.
  - We then build a dictionary `model_label2id` from these labels (lowercased), e.g. `{"negative": 0, "neutral": 1, "positive": 2}`.
  - Our dataset `Sentiment` values (e.g., "Negative", " Neutral ", "Positive") are stripped and lowercased, and then mapped through this dictionary so that `df["label"]` uses exactly the same IDs as the RoBERTa model.
- For the TextCNN and embedding-based model, we apply `sklearn.preprocessing.LabelEncoder` to obtain label IDs.
  - After cleaning the `Sentiment` strings (stripping whitespace), we fit a `LabelEncoder` on `df["Sentiment"]`.
  - We then create `df["label"] = le.fit_transform(df["Sentiment"])` , which maps the three sentiment classes to integer IDs 0, 1, 2 in the order learned by the encoder.

  - The number of classes for the final linear layer is obtained as `num_classes = len(le.classes_)`.

2.1.5 Text tokenization

- For RoBERTa, we use its built-in subword BPE tokenizer via `AutoTokenizer`, with truncation and padding to a fixed `max_length` for efficient batching.
- For TextCNN, we only reuse a transformer tokenizer (DistilBERT-base-uncased) to turn text into token IDs, with padding to length 80, truncation beyond length 80, and pad token ID = tokenizer's pad token. Padding uses the tokenizer's pad token ID, which is passed to the embedding layer so that padded positions do not affect the model's outputs.
- For the embedding-based MLP model, we use the same tokenizer and tokenization settings as the TextCNN for consistency across lightweight models. Comments are converted into token IDs using the DistilBERT-base-uncased tokenizer, padded or truncated to a fixed length of 80 tokens. The padding token ID is passed to the embedding layer.

2.2 Model 1: RoBERTa-base Transformer

2.2.1 Base model

- We use `cardiffnlp/twitter-roberta-base-sentiment-latest`, a publicly available 3-class sentiment classifier with labels {negative, neutral, positive}.
- All comments are tokenized using the model's tokenizer with a maximum sequence length of 80 tokens; longer inputs are truncated and shorter ones padded to this length.
- Before any fine-tuning, we evaluate this pretrained model directly on our validation split to establish a zero-shot baseline.

2.2.2 Fine-tuning model

- We use the Hugging Face `Trainer` API with:
  - The model is loaded with AutoModelForSequenceClassification, and the tokenizer configuration is identical to the baseline evaluation (same tokenizer, truncation, and maximum length of 80).
  - We use a data collator with dynamic padding (DataCollatorWithPadding) to support efficient batching.
  - The model is trained for 3 epochs with a learning rate of 5×10^{-5}, weight decay of 0.01, a batch size of 16 for training, and a batch size of 32 for validation.
  - The best checkpoint is selected based on macro-averaged F1 score on the validation set.
- We fine-tune the model for 3 epochs, monitoring validation metrics.

2.2.3 Prediction function

- We define a helper function `predict_sentiment(texts)` that:
  - Tokenizes incoming text(s) with the same tokenizer and `max_length`.
  - Runs the fine-tuned model in evaluation mode.
  - Applies `argmax` over logits to obtain label IDs.
  - Maps label IDs back to human-readable sentiment strings using `id2label`.

2.3 Model 2: Embedding + Multi-kernel TextCNN

2.3.1 Architecture

- Embedding layer: a trainable embedding layer maps each token ID in the vocabulary to a dense vector
  - Vocabulary size: equal to the size of the tokenizer's vocabulary
  - Embedding dimension: 128

  - The embedding for the padding token is fixed via a padding index, so that padded positions don't influence the output.
- Convolution layer: three one-dimensional convolutional layers with different kernel sizes to capture patterns of varying n-gram lengths.
  - Kernel sizes: 3, 4, and 5 tokens
  - Number of filters per kernel size: 64
  - Each convolution operates over the sequence of embeddings and produces feature maps that capture local patterns related to sentiment.
- Each convolution is followed by a non-linearity (ReLU) and max-pooling over time, which reduces each feature map to a single value.
- The outputs of the three convolution branches are concatenated into a 192-dimensional vector.
- Dropout layer with `p=0.5` for regularization.
- The final linear layer maps this representation to three output logits corresponding to the sentiment classes.

2.3.2 Training setup

- We build custom PyTorch `Dataset` and `DataLoader` objects for the tokenized sequences and labels.
- Both training and validation DataLoaders use a batch size of 32. The model is trained for 8 epochs using the Adam optimizer with a learning rate of 0.001 and no weight decay. Cross-entropy loss over the three sentiment classes is used as the training objective.
- During each epoch, the model prints the training loss and validation accuracy and macro-averaged F1 score. After each epoch, the validation performance is evaluated using the same validation split used for the RoBERTa model. The best model checkpoint is selected according to the highest validation macro-F1 score and saved to "textcnn_best.pt" for later evaluation and inference.

2.4 Model 3: Embedding + Multilayer Perceptron (MLP)

2.4.1 Architecture

- To establish a lightweight baseline and determine if complex sequence modeling is strictly necessary, we implemented a simple Multilayer Perceptron (MLP) model.
- Embedding Layer: Similar to the TextCNN, we use a trainable embedding layer (dimension 128) to convert token IDs into dense vectors.
- Global Average Pooling: Instead of using convolutions or attention mechanisms, this model computes the mean of the embeddings across the sequence length. This collapses the token sequence into a single fixed-size vector representing the "average" meaning of the comment.
- Hidden Layers: The pooled vector is passed through a feed-forward network:
  - Layer 1: Linear layer ($128 \to 128$) + ReLU + Dropout ($p = 0.5$).
  - Layer 2: Linear layer ($128 \to 64$) + ReLU + Dropout ($p = 0.5$).
  - Output: Linear layer ($64 \to 3$ classes).

2.4.2 Training Setup

- Optimizer: Adam with learning rate 1×10^{-3}.
- Loss Function: Cross-Entropy Loss.
- Epochs: 8 (saving the best model based on validation Macro-F1).

2.5 Model selection and implementation rationale

We selected these three model families to study the trade-off between model expressiveness and computational efficiency. RoBERTa represents a high-capacity pretrained transformer, while TextCNN captures local n-gram patterns with a lightweight convolutional architecture, and the embedding-based MLP serves as a minimal baseline without explicit sequence modeling. Beyond using existing libraries, we implemented custom preprocessing logic for label alignment with RoBERTa, designed and trained the TextCNN and MLP architectures in PyTorch, and built custom Dataset and DataLoader pipelines to support consistent training and evaluation across all models.

3 Experiments and Results

3.1 Evaluation Setup

We evaluate three models on the same held-out validation set. Because the dataset is imbalanced, we report accuracy as well as macro-averaged precision, recall, and F1 score, which weight each sentiment class equally. For all trained models, we select the best checkpoint based on validation macro-F1 to reduce overfitting and ensure fair comparison across architectures:

- Accuracy: overall fraction of correctly classified comments.
- Macro-averaged precision: average precision across the three classes, weighting each class equally.
- Macro-averaged recall: average recall across the three classes.
- Macro-averaged F1: harmonic mean of macro precision and macro recall.
- Per-class classification report: precision, recall, and F1 for each individual class. All metrics are computed using `sklearn.metrics` (`accuracy_score`, `precision_score`, `recall_score`, `f1_score`, and `classification_report`).

3.2 Baseline

3.2.1 Pretrained RoBERTa without Fine-tuning

We first evaluate the pretrained cardiffnlp/twitter-roberta-base-sentiment-latest model directly on the MrBeast validation set, without any additional training. This shows that although the off-the-shelf

Metric	Value
Accuracy	0.7838
Macro F1	0.5738
Macro Precision	0.5782
Macro Recall	0.7158

Table 1: Performance of off-the-shelf Twitter sentiment model on the YouTube dataset.

Twitter sentiment model is somewhat reasonable, it struggles with the domain shift to YouTube comments on a specific channel, especially in terms of macro-F1.

3.2.2 Fine-tuned RoBERTa on MrBeast Dataset

We fine-tune the same RoBERTa checkpoint on the MrBeast training split for 3 epochs. Fine-tuning dramatically improves performance across all metrics, with accuracy increasing by about 21 percentage points and macro-F1 increasing by over 0.42.

Metric	Value
Accuracy	0.9926
Macro F1	0.9941
Macro Precision	0.9955
Macro Recall	0.9927

Table 2: Performance of RoBERTa fine-tuned on the MrBeast dataset.

3.2.3 TextCNN Results

For the multi-kernel TextCNN, we train for 8 epochs and keep the best checkpoint based on validation macro-F1.

Metric	Value
Accuracy	0.9831
Macro F1	0.9863
Macro Precision	0.9899
Macro Recall	0.9830

Table 3: Performance of the TextCNN

3.2.4 MLP Results

For the embedding-based MLP, we train for 8 epochs and keep the best checkpoint based on validation macro-F1.

Metric	Value
Accuracy	0.9542
Macro F1	0.9610
Macro Precision	0.9654
Macro Recall	0.9588

Table 4: Performance of the embedding + MLP baseline.

Table 5: Performance Comparison of All Models

Metric	Pretrained RoBERTa	Fine-Tuned RoBERTa	TextCNN	MLP
Accuracy	0.7818	**0.9919**	0.9779	0.9542
Macro F1	0.5711	**0.9936**	0.9823	0.9610
Macro Precision	0.5754	**0.9929**	0.9832	0.9654
Macro Recall	0.7122	**0.9942**	0.9814	0.9588

4 Discussion

The experimental results highlight several key observations across the three evaluated model families.

1. **Importance of domain-specific fine-tuning**
   - The pretrained RoBERTa sentiment model, trained on Twitter data, performs reasonably on MrBeast YouTube comments but is clearly affected by domain shift.
   - After fine-tuning on the MrBeast dataset, the same architecture achieves near-perfect validation performance, demonstrating that domain-specific adaptation is crucial even between closely related social media platforms.
   - This result underscores the importance of in-domain labeled data for maximizing the effectiveness of large pretrained language models.
2. **Model complexity and performance trade-offs**
   - Fine-tuned RoBERTa achieves the strongest overall performance but requires substantially higher computational and memory resources.
   - The TextCNN model attains performance close to the transformer while using a significantly simpler architecture, offering faster training and inference.

- The embedding-based MLP, despite lacking explicit mechanisms for modeling word order or contextual interactions, also achieves competitive results, indicating that much of the sentiment signal in this dataset can be captured through aggregated embedding representations.
- These findings suggest that simpler models may be preferable in resource-constrained or latency-sensitive deployment scenarios, while transformers are best suited for offline analysis or GPU-backed systems.

3. **Generalization and overfitting considerations**
   - The extremely high validation scores observed across all models, particularly for fine-tuned RoBERTa, raise concerns about potential overfitting to the dataset's specific distribution or annotation style.
   - Because evaluation is performed on a single held-out split drawn from the same source, it remains unclear how well these models generalize to comments from unseen MrBeast videos or other YouTube channels.
   - Additional evaluation on a separate test set or cross-video splits would be necessary to more rigorously assess generalization.
4. **Dataset characteristics and failure modes**
   - The dataset is moderately imbalanced, with substantially fewer negative comments than positive or neutral ones. Despite this imbalance, all three models maintain strong performance on the minority class, suggesting that the learned representations are robust within this dataset.
   - Likely failure cases include extremely short or ambiguous comments, where sentiment may be unclear even to human annotators, as well as comments that rely heavily on sarcasm or contextual knowledge.
5. **Expectation**
   - Overall, the results largely meet expectations: fine-tuned transformers outperform simpler models, while TextCNN and MLP achieve strong performance given their lower complexity. Potential improvements include evaluating on comments from unseen videos, performing detailed error analysis, and incorporating techniques to better handle sarcasm and contextual ambiguity.

5 Conclusion

Extent to which goals were achieved:

- Preprocessing and understanding the dataset: Completed, with label cleaning and consistent encoding.
- Training and evaluating three model families (a pretrained and fine-tuned transformer, an embedding-based MLP, and a TextCNN): Completed, with detailed validation metrics and a clear performance comparison.
- Analyzing trade-offs and providing insight into model behavior: Completed; we discussed performance differences, computational trade-offs, and likely failure cases across models.

Future directions:

- Construct a separate test set (e.g., from different MrBeast videos) to better assess generalization.

Reflect on challenges:

- It is somewhat surprising that both TextCNN and the embedding-based MLP, despite their comparatively simple architectures, achieve performance close to transformer-level results with a modest dataset size.
- There are potential overfitting concerns that warrant deeper analysis, particularly given the very high validation performance.

References

[1] A. Shamim. MrBeast YouTube Comment Sentiment Analysis. *Kaggle Dataset*. Available at: `adilshamim8/mrbeast-youtube-comment-sentiment-analysis` (accessed 2025).

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

[3] Y. Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of EMNLP*, 2014.

[4] T. Wolf et al. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of EMNLP: System Demonstrations*, 2020. (Hugging Face Transformers library used for RoBERTa fine-tuning.)

[5] F. Pedregosa et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. (Used for metrics and preprocessing.)