Quantifying Sentiment: A Comparative Evaluation of Language Model Performance on the Rotten Tomatoes Corpus

Winter 2025 CS 545 Final Project Report

Alec Rogers, Luis Becerra, Mini Sengupta

Contents

Abstract

Sentiment analysis, the task of determining the emotional tone behind text, is a crucial application of Natural Language Processing (NLP). With the rapid advancement of Language Models (LM) and their opaque nature, researchers are attempting to develop methods, experiments and frameworks for assessment to characterize their performance in order to make sure their application is appropriate. This project conducted a comparative analysis of LM (a Custom baseline version against pre-trained models Flan-t5-small, RoBERTa and LLaMA-3.2-1B) on the Rotten Tomatoes movie review dataset. The Rotten Tomatoes dataset is a widely recognized resource for sentiment classification with a diverse range of expressions. By comparing their accuracy, F1-score, precision, or recall, we hope to gain insights and knowledge on LM-driven sentiment analysis.

Team members and roles

- Alec Rogers: Custom transformer algorithm owner and lead programmer.
- Luis Becerra: Flan algorithm owner and custom prompt engineering.
- Mini Sengupta: Bert and Llama algorithm owner and documentation lead.

1.0 Introduction

The adoption of Large Language Models (LLM) has been rapidly climbing since 2022. OpenAI reports that within 5 days of releasing ChatGPT (GPT-3.5) as a free research preview, 1 million users signed on, a record-breaking adoption rate. A wide range of expected use cases are emerging, spanning personal, professional, and educational domains. However, there have been findings such as hallucination and bias that have raised doubts about its accuracy. One such use-case is accurately quantifying sentiment, which has emerged as a critical challenge for applications spanning from market research to social media monitoring. Sentiment analysis is the method of identifying semantic orientation. It can be applied in an unsupervised machine learning mode. This project presents a systematic comparison of various language models in their ability to detect and classify sentiment within the Rotten Tomatoes corpus—a benchmark dataset of movie reviews that presents diverse linguistic patterns, nuanced expressions, and the inherent complexities of human opinion. By evaluating model performance across this standardized dataset, we aim to not only identify the most effective approaches for sentiment analysis but also to illuminate the strengths and limitations of current language technologies when tasked with understanding the subtleties of human sentiment.

2.0 Literature review

Sentiment analysis, also known as opinion mining, has become a cornerstone of natural language processing (NLP), driven by the explosion of user-generated content online. Early approaches used lexicon-based (pre-defined set of words and their associated sentiment scores [1]) and machine learning (e.g. Support Vector Machines [2]) approaches. With the rise of Deep Learning, Recurrent Neural Networks and Convolutional Neural Networks [3], more capabilities were available to derive sequential dependencies in text and learning features and apply them to sentiment analysis. The introduction of the Transformer architecture [4] revolutionized NLP. BERT and the follow-on, RoBERTa language models revolutionized sentiment analysis in language model studies by introducing contextualized embeddings, improving pre-training approaches, and achieving state-of-the-art results. And currently, LLMs such as the Llama family, and GPT family of models have shown promise in zero-shot and few-shot sentiment analysis. To summarize, we can see that there are many tools available to perform sentiment analysis that it behooves us to be able to compare them. Given the expected continued rise of content generation online and reliance on applications to use that information to personalize offerings to humans, this study attempts to study some of the common ways that sentiment analysis can be performed today both by developing a sentiment analysis engine from scratch along with contrasting it with leveraging open-source LMs. We reviewed example methods to perform comparisons of LLM [5], but given the project time available to us, we chose to focus on the accuracy of predictions using labeled datasets as the method to compare.

3.0 Methodology

The code was built upon the exercises found in the book "Hands-On Large Language Models", by Jay Alammar and Maarten Grootendorst [5]. One custom implementation (Transformer) was developed for a baseline and for the rest we leveraged pre-trained models that are available at huggingface (https://huggingface.co/models). The transformer codebase was written with the assistance of GPT-4o.

3.1 Dataset & Metrics Evaluation

The dataset that we used is the Rotten Tomatoes dataset (https://huggingface.co/datasets/cornell-movie-review-data/rotten_tomatoes). The dataset contains raw textual information provided by the user and the label being the sentiment expressed in the information. The dataset is derived from a database consisting of movie reviews from the online platform Rotten Tomatoes (https://www.rottentomatoes.com). The dataset details are:

- Target Labels = {0,1}
- Size of training set = 8530
- Size of validation set = 1066

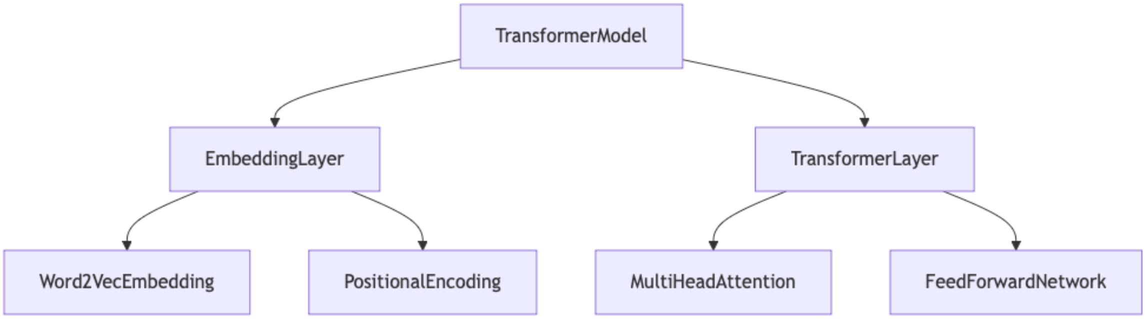
- Size of testing set = 1066

For the metrics, the confusion matrix is the basis and includes the following:

- Precision (True Positive/(True Positive + False Positive))
- Recall (True Positive/(True Positive + False Negative))
- F1-score (2 *((Precision*Recall)/ (Precision+Recall))
- Accuracy ((True Positive + False Positive)/(True Positive + False Positive + True Negative + False Negative)) and their averages
- Support provides information on the count of data points

3.2 Transformer

The transformer architecture can be broken down into two main parts, an embedding Layer and a transformer layer:



To keep things simple, only one transformer layer was used: however, modern GPT networks typically have dozens of transformer layers stacked on top of one another.

3.2.1 Embedding

- Word2Vec represents words as vectors in a similarity space, capturing semantic meaning based on context within some corpus.

- The embedding corpus in this case comes from Wikipedia (enwiki_20180420_100d.txt)
- The dimensionality of the input space is 100

3.2.2 Positional Encoding

- Absolute sinusoidal encoding was chosen.
- As I am not a fan of conflating the data with the positional encoding, the position was used to augment the embedding space.

3.2.3 Attention

- In a Transformer model, the first layer weight matrix (the first self-attention layer) transforms the input embeddings (e.g., from Word2Vec) by applying linear projections followed by self-attention. Thus, a word “apple” has an embedding $ x \in \mathbb{R}^{d} $, where d is the embedding dimension.
- Transformers modify embeddings based on sentence context (e.g., “apple” in “I ate an apple” vs. “Apple Inc. is a tech company” will have different contextual embeddings), whereas Word2Vec gives fixed representations (e.g., “apple” always has the same embedding).
- Word2Vec encodes meaning based on co-occurrence statistics, while self-attention refines this by modulating embeddings dynamically based on actual sentence structure.
- Word2Vec embeddings lie in a space optimized for distributional similarity, while the Transformer projects them into a new space.

The self-attention mechanism used here is called MultiHeadSelfAttention. multiple heads allow attention to pay attention to four different things or locations within weight space. Self-attention relies crucially on the notion of Q, K, and V vectors, which are matrix projections of the input embedding space.

1. Keys Represent Encoded Memory
2. Queries Retrieve from Memory
3. Values are the actual memories

$$\text{Attention} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_h}}\right)V = AV$$

$$Q = W^Q x \quad K = W^K x \quad V = W^V x$$

Where:

- A is the transition probability matrix, governing how information flows between tokens.
- QK^T captures pairwise attention relationships in the input (or context window).
- Softmax ensures probabilities sum to 1, making attention function as a probabilistic transition process.

Having multiple attention heads allows the model to maintain and retrieve different types of information simultaneously. Each head may store and retrieve different aspects of the sequence (e.g., syntactic relationships, semantic meaning), effectively increasing the short-term memory capacity.

Short-Term Memory Analogy

- The keys function like a dynamically updated short-term memory that holds a contextualized representation of tokens in a sequence.

- Since attention mechanisms operate within a fixed-length context window, the stored key representations serve as a temporary memory buffer that helps the model decide which past information is most relevant to the current token.
- Unlike long-term memory (e.g. which is stored in the network weights), this short-term memory is refreshed for each new input.

Connectionist Analogy

In a connectionist network, knowledge is stored in the weights of connections between neurons, which dictate how activation flows between units. These weights can be interpreted as transition probabilities, determining how likely it is that activation moves from one state (or neuron) to another. In the multi-head attention mechanism of transformers, the attention matrix (A) plays a similar role by dynamically encoding how strongly different elements (tokens or states) influence each other.

The Attention matrix (A) in a transformer attention mechanism is similar to the connections within a connectionist network. By computing dynamic attention scores via QK^T and normalizing them into a short-term attention matrix A, transformers generalize and improve upon connectionist models by enabling asymmetric associations (in virtue of separate Q an K matrices) and allowing context-aware transitions between states in a sequence.

3.2.4 Feedforward Network

The output of the embedding is typically transformed with multiple feedforward layers: here, only one layer is used. The operation of the feedforward neural network (FFN) can be written:

$$y = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$$

where:

- x and y are the input and output, respectively.
- W_1, W_2 are learned weights.
- b_1, b_2 are biases.

This network layer is just a MultiLinearPerceptron with an ReLU activation function, with an added linear output layer.

3.2.5 Training Parameters

Because overtraining was problematic on this dataset, a stopping criterion was introduced when the performance on the validation data set decreased by more than 0.05. The training parameters used:

Error	= MSE
Embedding dimension	= 100
Number of epochs	= 100
Learning Rate	= 0.001
Batch Size	= 100
Stopping Criterion	= 0.05

3.3 Flan-t5-small

Flan-t5-small is an open-source large language model (LLM) developed by Google [6]. Flan-T5 is designed for natural language processing (NLP) tasks through instruction fine-tuning, which is a technique where a pre-trained LLM is further trained on a dataset consisting of instruction-response pairs. This allows it to better understand and follow human instructions. Flan has an encoder-decoder (Seq2Seq) architecture, which means its neural network is designed for sequence-to-sequence tasks where an input sequence is mapped to an output sequence. These characteristics make Flan-T5 a good algorithm to classify our Rotten Tomatoes dataset. As we will see, Flan-T5 had the highest performance metrics compared to the other LLMs tested. However, it does not perform as well as other LLMs (like ChatGPT) for conversational uses and creative text generation.

3.3.1 Prompt Engineering

Another subject of interest for this project is prompt engineering, which is a technique of carefully crafting the prompts to guide the LLM towards a desired output. In this project a default prompt (prompt 1) was used with each LLM; however, to increase the prediction metrics the following prompts were tested:

- **Prompt 1 (Default):** Is the following movie review positive or negative?
- **Prompt 2:** Is the following movie review from rotten tomatoes positive or negative?
- **Prompt 3:** Pretend you are a cinephile. Is the following movie review from rotten tomatoes positive or negative?
- **Prompt 4:** Pretend you are a highly educated movie critic who likes dramas and documentaries. Is the following movie review from rotten tomatoes positive or negative?

3.4 RoBERTa

The RoBERTa pre-trained model was accessed from Huggingface [7]. A high-level overview of the execution flow in RoBERTa is shown in Figure 3.4.1.

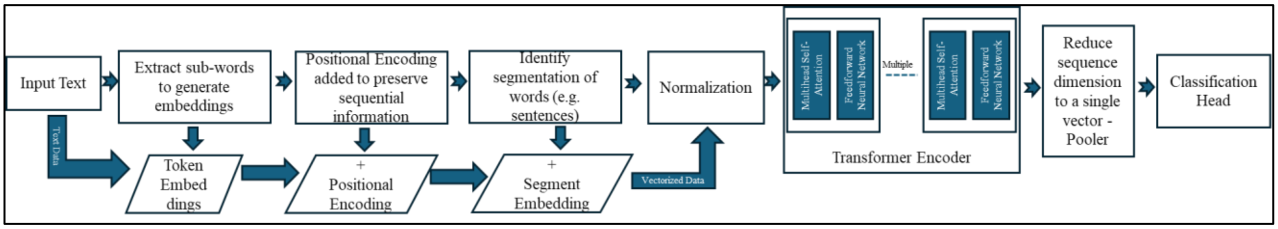

Input Text
Extract sub-words to generate embeddings
Positional Encoding added to preserve sequential information
Identify segmentation of words (e.g. sentences)
Normalization
Multihead Self-Attention
Feedforward Neural Network
Multiple
Multihead Self-Attention
Feedforward Neural Network
Transformer Encoder
Reduce sequence dimension to a single vector - Pooler
Classification Head
Text Data
Token Embed dings
+ Positional Encoding
+ Segment Embedding
Vectorized Data

Figure 3.4.1: RoBERTa overview

Encoder only models such as RoBERTa excel at contextualizing word relationships within input sequences. However, due to the missing Decoder component RoBERTa is not well suited for generative capabilities. Given that RoBERTa is a pre-trained model, the training task was not performed.

3.5 Llama-3.2-1B

The LLaMA-3.2-1B model was chosen as the smallest of the current LLaMA models for this experiment. The model was accessed from Huggingface [8]. An access token was obtained in order to use the model. The model includes pre-trained weights for instruction-tuned text only. The model is optimized for multilingual dialogue use cases, including agentic retrieval and summarization tasks which doesn't seem to cover sentiment analysis as we found during experimentation. It contains a total of 1 billion parameters and is trained using a masked language modeling objective. A high-level overview of the execution flow in LLaMA-3.2-1B is shown in Figure 3.5.1.

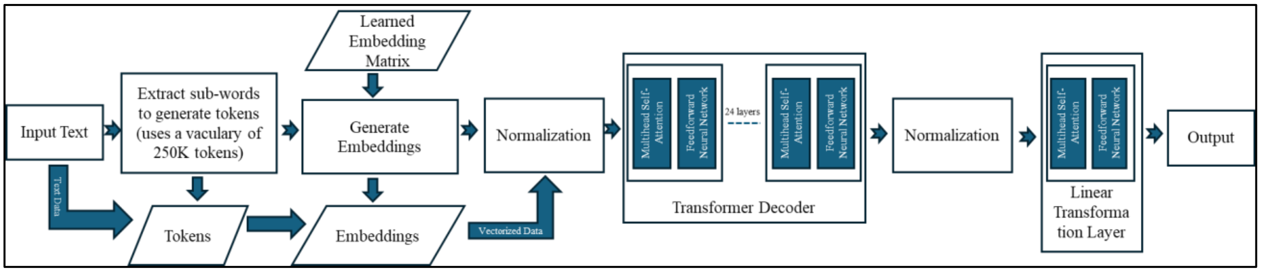
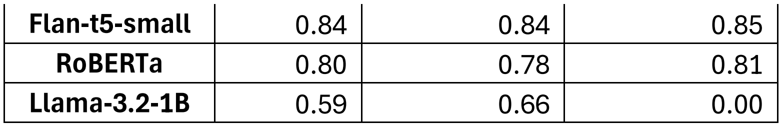


Figure 3.5.1: LLaMA-3.2-1B overview

4.0 Results and Discussion

4.1 Overall Results

		f1-score	
	Accuracy	Positive Review	Negative Review
Transformer	0.74	0.75	0.73



Flan-t5-small	0.84	0.84	0.85
RoBERTa	0.80	0.78	0.81
Llama-3.2-1B	0.59	0.66	0.00

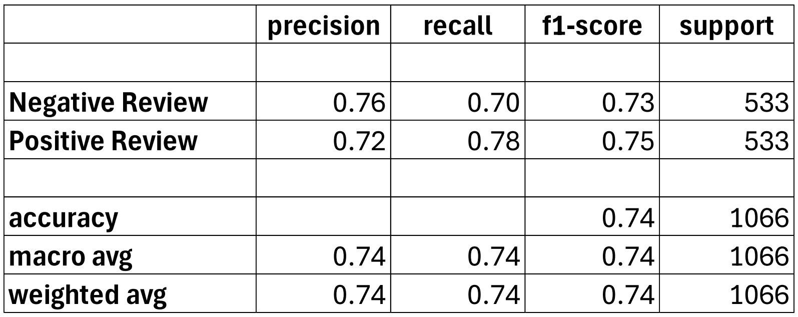
Table 4.1. Metrics summary for the 4 algorithms in this study.

The custom build Transformer performed competitively to the open-source models on this specific dataset and task. While literature indicated that RoBERTa performs excellently in sentiment analysis, Flan-t5-small performed better. Llama-3.2-1B performed the worst of all; accuracy that we were able to obtain is analogous to fair chance. This is an indication that training for the model is in order, more on that in the discussion of the results.

4.2 Transformer Results

The Transformer program performance is shown in Table 4.2.1 and the program execution exhibited the following metrics:

- Epoch [99/100]
- Train Loss: 0.1613
- Validation Loss: 0.2205
- Test Loss: 0.1632
- Training stopped due to overfitting.



	precision	**recall**	**f1-score**	**support**
Negative Review	0.76	0.70	0.73	533
Positive Review	0.72	0.78	0.75	533
accuracy			0.74	1066
macro avg	0.74	0.74	0.74	1066
weighted avg	0.74	0.74	0.74	1066

Table 4.2.1. Transformer performance metrics

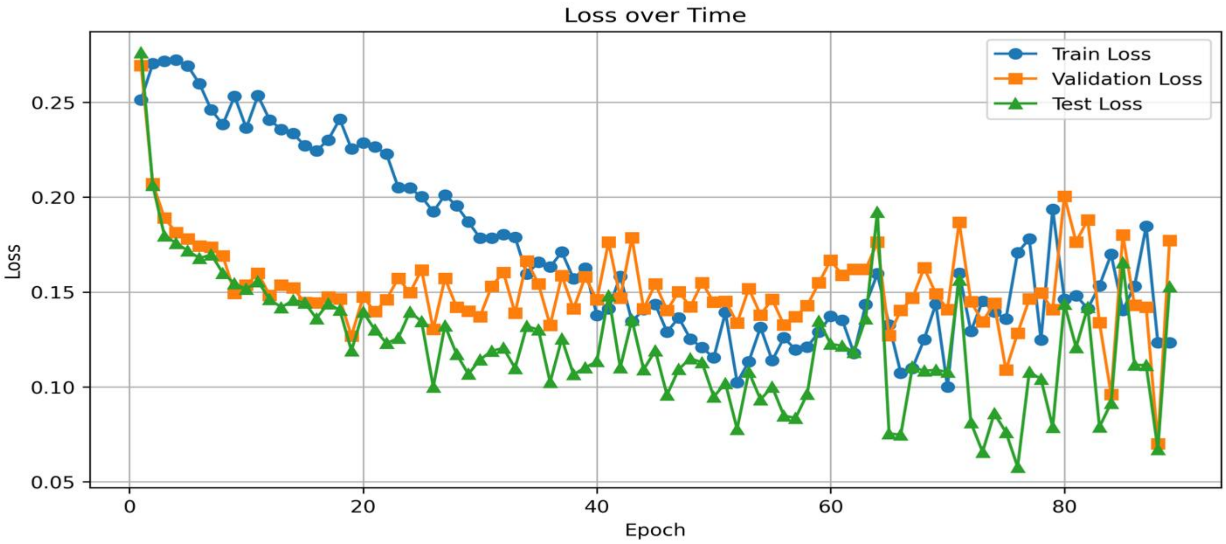


Figure 4.2.1 256 hidden layer neurons

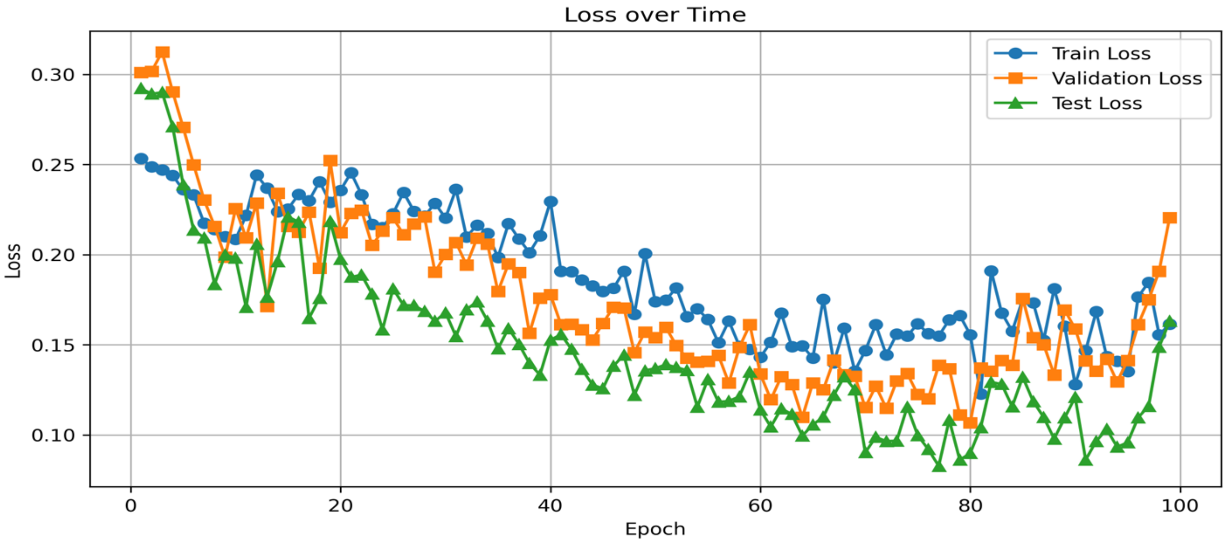


Figure 4.2.2 64 hidden layer neurons

Preventing overfitting was a main concern when tuning the algorithm parameters. The use of a validation set as a stopping criterion helped somewhat, but it also prevents training from happening for very long. Reducing the complexity of the network by using a smaller number of hidden layer neurons helped somewhat (see figures 4.2.1 and 4.2.2), although that did not prevent the network from triggering the validation stopping criterion. It also seemed to help reduce the inter-epoch variation in the data, which might also be addressed by a more intelligent schedule of learning rate decrease.

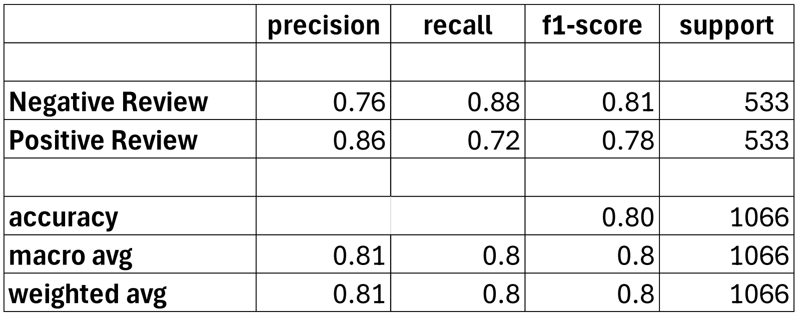
4.3 Flan-t5-small Results

While the Flan-T5-small LLM yielded high prediction metrics (e.g. 84% accuracy), Table 4.3.1 shows that the different prompts resulted in little to no effect on the classification metrics. This could be because the LLM is overly sensitive to the sentimental classification task, causing it to essentially ignore the additional context added. It could also be because the prompts are simply not providing enough additional context.

		precision	recall	f1-score	support
Prompt 1 (Default)	**Negative Review**	0.83	0.86	0.85	533
	Positive Review	0.86	0.83	0.84	533
	accuracy			0.85	1066
	macro avg	0.85	0.85	0.85	1066
	weighted avg	0.85	0.85	0.85	1066
Prompt 2	**Negative Review**	0.83	0.86	0.85	533
	Positive Review	0.86	0.82	0.84	533
	accuracy			0.84	1066
	macro avg	0.84	0.84	0.84	1066
	weighted avg	0.84	0.84	0.84	1066
Prompt 3	**Negative Review**	0.83	0.87	0.85	533
	Positive Review	0.86	0.82	0.84	533
	accuracy			0.84	1066
	macro avg	0.84	0.84	0.84	1066
	weighted avg	0.84	0.84	0.84	1066
Prompt 4	**Negative Review**	0.85	0.83	0.84	533
	Positive Review	0.84	0.85	0.84	533
	accuracy			0.84	1066
	macro avg	0.84	0.84	0.84	1066
	weighted avg	0.84	0.84	0.84	1066

Table 4.3.1. Classification report for Flan-t5-small and the different prompts tested.

4.4 RoBERTa Results

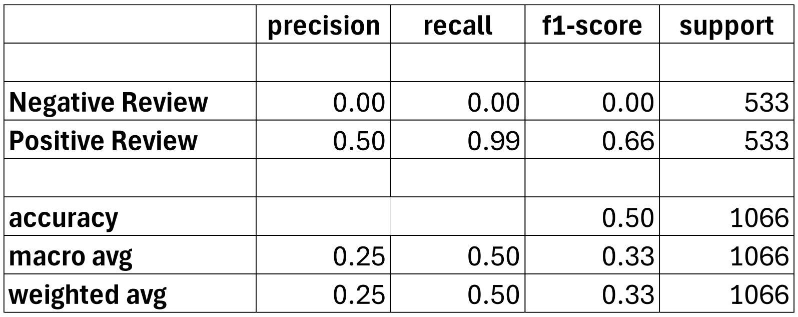


	precision	recall	f1-score	support
Negative Review	0.76	0.88	0.81	533
Positive Review	0.86	0.72	0.78	533
accuracy			0.80	1066
macro avg	0.81	0.8	0.8	1066
weighted avg	0.81	0.8	0.8	1066

Table 4.4.1. RoBERTa performance metrics

4.5 Llama-3.2-1B Results

The results for this model are shown in Table 4.5.1. As training looked to be required in order to improve the performance, it was attempted. First, the Trainer class from the transformers library and then the AdamW optimizer from the torch.optim library. However, both types of training ran into the error “RuntimeError: CUDA error: out of memory”. We could not locate ways to resolve the error in time and perform the training.



	precision	recall	f1-score	support
Negative Review	0.00	0.00	0.00	533
Positive Review	0.50	0.99	0.66	533
accuracy			0.50	1066
macro avg	0.25	0.50	0.33	1066
weighted avg	0.25	0.50	0.33	1066

Table 4.5.1. Llama-3.2-1B performance metrics

5.0 Conclusions, Limitations and Future Work

We were able to successfully complete the comparison of accuracies between the models that were selected in the initial project proposal (Appendix A). The effort gave us a better appreciation on how text is handled in the context of language models in Machine Learning (ML). It also exposed us to the skills, advanced libraries, repositories, prompt engineering, etc., that are needed to interact with LMs similar to the ML practitioners in the industry.

Our findings indicate that much of the work of the analysis of the review is done by the embedding itself. Anecdotally, predicting good or bad reviews simply by doing a cosine similarity and using that vector as a predictor achieves a best-in-class accuracy of 84%, which suggests that we were not able to take much advantage of any training above and beyond that embedding (which is the first step in the use of any transformer architecture).

Transformers need lots of data; training them on limited datasets adds to the difficulty of an already difficult problem. Training a transformer from scratch, as opposed to using a GPT (Generative Pre-trained Transformer), is probably not a great solution to this prediction problem since transformers generally rely on vast amounts of data, and the rotten tomatoes dataset is not large. That said, the simple transformer developed here does serve as a good reference implementation to understand the cognition of LLMs in general.

Using existing Generative Pretrained Transformers seems to be adequate, with careful prompt engineering.

Training is not always required for pre-trained models. However, attention needs to be paid to account for performing training after reviewing the results as noticed in our comparison. Also, if training is needed, careful research is needed to understand what training method will work best for the model and what resources will be needed. As we found in this project, our available computers were not adequate for the training.

We expect that a sentiment analysis engine on different datasets other than rotten tomatoes may need a different approach. For example, when reviewing technical reviews of products, what additional capabilities would be needed for the analysis.

Ideally, if we had additional time we would have built a framework that would harness and present the data side-by-side and can be run at any time and with different datasets [9]. There are a few other angles that we did not have time to look at such as multi-lingual sentiment analysis, enhanced emotion detection and making the predictions explainable.

References

[1] P. Turney, "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Philadelphia, 2002.

[2] B. Pang, L. Lee and S. Vaithyanathan, *Thumbs up? Sentiment Classification using Machine Learning Techniques,* 2002.

[3] Y. Kim, *Convolutional Neural Networks for Sentence Classification,* 2014.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Poloshukin, *Attention Is All You Need,* 2023.

[5] J. Alammar and M. Grootendorst, Hands-On Large Language Models: Language Understanding and Generation, O'Reilly Media, 2024.

[6] "Hugging Face Models - google/flan-t5-small," [Online]. Available: https://huggingface.co/google/flan-t5-small. [Accessed 2025].

[7] "Hugging Face Models - cardiffnlp/twitter-roberta-base-2021-124m," [Online]. Available: https://huggingface.co/cardiffnlp/twitter-roberta-base-2021-124m. [Accessed 2025].

[8] "Hugging Face Models - meta-llama/Llama-3.2-1B," [Online]. Available: https://huggingface.co/meta-llama/Llama-3.2-1B. [Accessed 2025].

[9] M. Kahng, I. Tenney, M. Pushkarna, M. X. Liu, J. Wexler, E. Reif, K. Kallarackal, M. Chang, M. Terry and L. Dixon, "LLM Comparator: Visual Analytics for Side-by-Side Evaluation of Large Language Models," 2024. [Online]. Available: https://arxiv.org/html/2402.10524v1.

Appendix : A – Initial Project Proposal

CS545_group_project_proposal.docx