Product Review Sentiment Analysis with GPT3.5-Turbo and Other Large Language Models

Rasmus Rynell
Rynell.Rasmus@gmail.com

Course: TDDE16
LiU-ID: rasry945

Abstract

This paper explores the effectiveness of five different large language models and one Naive Bayes baseline classifier in sentiment analysis of product reviews. Two of the language models were previously fine-tuned, one was fine-tuned on this specific dataset, and the remaining two large language models were the OpenAI models *text-Curie-001* and *gpt-3.5-turbo*. Additionally, random oversampling and few-shot prompting were also explored for relevant models. The results show that the overall best model was the custom fine-tuned one, with a macro average f1 score of 87.87%, compared to the Naive Bayes model with 77.09% and the *GPT3.5-turbo* with 77.99%. The experiments conducted also concluded that random oversampling and few-shot prompting were not effective methods at yielding overall better results for this particular problem and dataset.

1 Introduction

The ability to analyze and understand emotions and opinions expressed in text is called sentiment analysis. The technique is often used to classify a corpus of text into classes such as, positive, neutral, or negative, but can also include classes such as emotions. Sentiment analysis can be used by many entities, such as businesses, governments, and researchers, to analyze and give insights into people's opinions and moods about different topics (Birjali et al., 2021).

One important topic that can be analyzed using sentiment analysis is the intention and meaning behind reviews of products. The analysis could be utilized not only by companies to better understand the meaning behind reviews, but also to help customers get an overview of what other people think about the product. By learning more about effective ways of doing sentiment analysis for product reviews, one can hopefully gain knowledge on how such a system can be implemented and used by many entities.

This paper investigates how a baseline NB classifier (NB) compares against five different Large Language Models (LLMs) for sentiment analysis of product reviews. Specifically, two already fine-tuned LLMs named *twitter-roberta-base-sentiment-latest* (LLM1) and *distilbert-base-multilingual-cased-sentiment-2* (LLM2). One custom fine-tuned, on the specific dataset used tested, *DistilBERT base model (uncased)* model (LLM3). And lastly, two models from OpenAI, *text-Curie-001* (OA1) and *gpt-3.5-turbo* (OA2). The paper also conducts experiments on few-shot prompting for the (OA1) and (OA2) models and how randomly oversampling the training data effects (NB) and (LLM3).

All experiments were carried out on the same sentiment analysis dataset, using the same train/test split, both with and without random oversampling where necessary. However, some experiments regarding (OA1) and (OA2) were done on a subset of the test data, since these models are hosted by OpenAI and are pay-to-use. Additionally, to better understand the structure of the dataset, some initial analysis was conducted on both the entire dataset, and on the train/test sets.

2 Theory

This chapter presents theory about the main areas in the paper.

2.1 Large Language Models

Large Language Models are deep learning models trained on a large corpus of text for tasks such as translation, summarization, generation, and/or classification. BERT (*Bidirectional Encoder Representations from Transformers*) and GPT (*Generative Pre-Trained Transformer*) are two of the most successful families of LLMs to date (Min et al., 2021). Both models are based on the transformer architecture, which uses a self-attention mechanism to process sequential data (Vaswani et al., 2017).

While BERT models are bidirectional and built from the encoder part of the transformer (Devlin et al., 2019), GPT models are unidirectional and built from the decoder part of the transformer (Radford et al., 2018). Two model variants spawned from BERT are the *DistilBERT* and *RoBERTa* models. *DistilBERT* is a smaller and faster version of the original BERT models, created by a technique called distillation (Sanh et al., 2019). While *RoBERTa* is an optimized version of the original BERT models, improved by both its training process and the amount of data for training.

Fine-tuning

Fine-tuning a model is the process of continuing to train an already trained model for a specific downstream task. The process can take many forms, but for LLMs the most common setup is that the model has been pre-trained on a large text corpus of unlabeled data. The model can then be trained for a more specific task, such as sentiment analysis, by continuing training on a labeled dataset (Ruder, 2021). Depending on what type of model that is being fine-tuned and for what task, one could either add *head*, a small new network on top of the then *backbone* model or if the model architecture allows, skip the head.

Few-Shot prompting

Prompts are the initial text given to a LLM on which it bases its output. As Gao writes, "A prompt is a piece of text inserted in the input examples, so that the original task can be formulated as a (masked) language modeling problem" (Gao, 2021). In other words, a prompt is the way to interact with a LLM model by giving it input. As Gao mentions, carefully crafting a prompt and analyzing the result allows the LLM to perform tasks it may not have originally been trained for. As an example to, finish the text in such a way as to answer a question or carry out a sentiment analysis. Few-shot prompting is a technique where a model is given a small number of examples, typically between two and five, in the prompt to quickly adapt to a new task. An example of this technique can be seen in Table 4.

2.2 Fine-tuned models

The first model tested, (LLM1)[1], is a version of the *roberta-base* model, which has been fine-tuned for the sentiment analysis task on the *tweet_eval* dataset (Barbieri et al., 2020). While two of the other models, (LLM2)[2] and (LLM3)[3] on the other hand, are both *DistilBERT* models. (LLM2) has already been fine-tuned for the sentiment analysis task on the *amazon_reviews_multi* dataset (philschmid, 2023) and (LLM3) is fine-tuned in this paper.

Field name	Description
ProductName	The product name
ProductPrice	The price of the product
Rate	A number rating
Review	A rating in text form
Summary	A written product review
	(positive, negative, or neutral)

Table 1: Fields in the used dataset.

2.3 OpenAI models

Very little information is currently known about the OpenAI models (OA1) and (OA2). However, what is known, is that the two models comes from the GPT-3 family of models which builds on the GPT architecture. (OA1), the *text-curie-001* model, is a text-completion model described as "Very capable, faster and lower cost than Davinci" (models, 2023), Davinci being the best performing model in the GPT-3 family. While (OA2), also known as *GPT3.5-turbo* is a chat optimized version of the GPT-3 models (models, 2023). Both models are available via OpenAI's API[4].

3 Data

The dataset used for the experiments is the *171k product review with Sentiment Dataset* (vaghani, 2023) downloaded 2023-03-04. The specific data used is located in the sentiment.csv which has the original field presented in Table 1.

The dataset is currently being updated regularly by its author, scraping the Flipkart[5] website using the BeautifulSoup Python library.

3.1 Pre-Processing

The following preprocessing steps, along with explanations, were done to clean up the dataset before any use:

1. Dropped row 17301 *due to it being corrupted*
2. Removed all fields except "Summary" and "Sentiment" *since they are not used*

[1]Found at: https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest

[2]Found at: https://huggingface.co/philschmid/distilbert-base-multilingual-cased-sentiment-2

[3]Original non-fine-tuned model found at: https://huggingface.co/distilbert-base-uncased

[4]OpenAI's API: https://platform.openai.com/

[5]Flipkart website: https://flipkart.com

3. Dropped rows with any field being NA *to only keep data that can be worked with*
4. Converted the "Sentiment" field to integers *(2:positive, 1:neutral, 0:negative) to be compatible with all models*
5. Removed rows with a summary longer than 450 words *(removed $\approx$ 1%) due to VRAM limitations*
6. Randomly oversampled the training data *(Done only for relevant experiments)*

4 Method

This section explains each part of the experiment's methods. To keep results reproducible, all experiments, and data splits were carried out using a set seed of 42. In the experiments involving oversampling of training data, the *RandomOverSampler* method from the *imblearn* Python library was used with default parameters and set seed.

4.1 Analyzing the dataset

The dataset analysis was conducted by analyzing 3 key data properties:

1. Word count
2. Text length
3. Sentiment distribution

To generate the word count, stop-words were removed using Spacy's built-in English *en_core_web_sm* pipeline, and each remaining word was converted to lowercase. Text length and distribution of sentiments were calculated using built-in features from the pandas package.

4.2 Naive Bayes

The Naive Bayes model was created using the scikit-learn Python package. A scikit-learn pipeline object was constructed by combining the *CountVectorizer* feature extractor and *MultinomialNB* model, both from the same package. The *CountVectorizer* converts the text into a matrix of token counts that can be utilized by the *MultinomialNB* model. To ensure that optimal hyperparameters were used, a 5-fold cross validation grid search was also done over the parameters presented in Table 2.

Name	Values
vec_binary	[False, True]
vec_ngram_range	[(1,1), ..., (5,5)]
MNB__alpha	[0.1, 0.5, 1.0, 1.5, 2.0]

Table 2: Grid searched hyperparameters for the Naive Bayes baseline classifier model.

4.3 Already fine-tuned LLMs

The Large Language models, (LLM1) and (LLM2), which were already fine-tuned for sentiment analysis, were downloaded and processed using the python library called *Transformers*. The library offers an abstraction layer to handle both the model and its tokenizer. This was done through the library special sentiment-analysis pipeline, which takes the text to classify as input and automatically converts it using the supplied tokenizer to then be sued by the model. In this paper, all the LLMs use their own original tokenizers to maintain fairness in the comparisons, since they were trained using their own tokenizers.

4.4 Custom fine-tuned LLM

To fine-tune (LLM3) the *Trainer* class of the *Transformers* library was used. The model was created using the *DistilBertForSequenceClassification* class. This class implements a simple multi-class classification head on top of the supplied backbone, which in this case is the *distilbert-base-uncased* model. Since the default hyperparameters for multi-class classification head looked reasonable for this problem, they were kept.

The hyperparameters for fine-tuning (LLM3), both with and without oversampling, can be seen in Table 3. Many hyperparameters were tested, but these were the once changed from the defaults in the trainer class. Some hyperparameters, such as fp16, were not chosen based on producing the overall best results, but instead to strike a balance between training time and results. Finally, early stopping and selecting the model with the lowest validation score during training were used to keep training times as low as possible while still achieving the *best*[6] results. When searching for the best hyperparameters, the training data was split into another 90/10 training/validation split. This split was also used for early stopping and ultimate model selection during the fine-tuning.

4.5 OpenAI's models

As stated in the introduction, some tests done on OpenAI's models, (OA1) and (OA2) were smaller in scope due to them being pay-to-use. These smaller experiments were conducted on 1000 randomly sampled test examples. Therefore, the results of these experiments cannot be directly compared to the results obtained from the larger models. They can however be interpreted as general

[6]Not guaranteed to be global best

Name	no os	os
learning rate	$1.5e^{-4}$	$2e^{-4}$
train batch size	128	128
eval batch size	512	512
warmup steps	250	250
weight decay	0.02	0.02
adam beta 1	0.91	0.89
adam beta 2	0.997	0.995
metric best model	'eval_loss'	'eval_loss'
eval steps	50	50
esp	25	25
fp16	True	True

Table 3: (Changed) Hyperparameters for (LLM3). esp, os = early stopping patience and oversampling.

guidelines as to how the models might perform had it been done on all the training examples. After these experiments, the best-performing model was further tested using both the full test data and a further randomly sampled 5000 examples. The 5000 experiments aims to reveal the tradeoff between achieving accurate results and minimizing the number of tests using the paid OpenAI API, while the full dataset tests produce results that can be compared to the other models.

Since the (OA1) model is only a text completion model, its prompt has to be structured in a special way to be used for sentiment analysis. The original GPT-3 paper's (Brown et al., 2020) authors set up and test this type of problem by simply giving context in text form and then asking the model to answer a question. Therefore, the prompt used in this paper is set up very similarly. An example input to the (OA1) model can be seen in Table 4. As the example shows, this type of setup also allows for the 0-, 1-, and 3-shot prompting to be conducted by adding already known questions and answers to the text. Since the answers that the model procure is just plain text, one cannot be sure that it always answers with a sentiment. Therefore, the answer is considered "positive", "negative" or "neutral", if the response contains that word, but if another answer is given such as "I am not sure" or "The sentiment analysis for this task could be ..." the answer is defaulted to "neutral".

The interface for interacting with the (OA2) model is different from (OA1). Even though the model at its core is still trained for text completion, it is built and interacted with as a chatbot. Therefore, its input instead consists of 3 roles: "system", "user", and "assistant". The "system" role describes the environment, while the "user" is the questioner

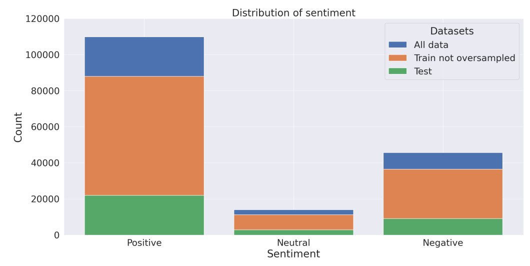

Figure 1: The sentiment distribution for the full dataset, and for train/test splits.

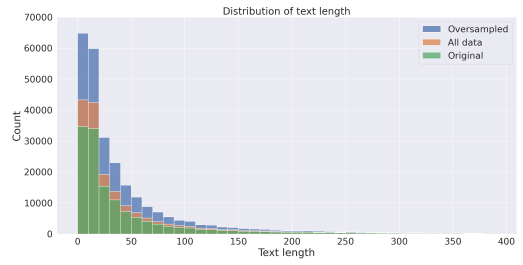

Figure 2: The text length distribution of the full dataset, and for train/test splits.

in this case, and "assistant" is the model answers (chat guide, 2023). Prompting the model is therefore done by following the structure shown in Table 5. The answers back from the model is handled in the same way as for (OA1).

When performing the 1-shot prompting for both (OA1) and (OA2), one of the three *italic* texts in Table 5 was randomly inserted. For the 3-shot prompts, all three *italic* texts were used in a randomly shuffled order.

5 Results

This section presents the analysis and experiment results.

5.1 dataset analyzys

Figure 1 shows the sentiment distribution of the dataset both as a whole and when divided into train/test sets. The figure shows that the original dataset is highly imbalanced, and that this remains the case even after splitting the data. It is important to note that after oversampling, the training (orange) bars will be the same height as the currently highest orange bar.

Figure 2 displays the length of the text for the dataset as a whole and for the train/test splits. The figure shows that the majority of the reviews are short, with almost half of them being less than 30 words. Additionally, the distribution remains consistent for both the training and test sets.

Individual counts of words for the train datasets can be seen in Figure 3. The figure reveals that

"This is a product review sentiment analysis. You can choose your response from the following options: ["positive", "neutral", "negative"]
Review: "Works great, a little expensive but worth it."
Sentiment: positive
Review: "It has a very high price but seems to be good quality."
Sentiment: neutral
Review: "Very low quality, I would not recommend this product."
Sentiment: negative
Review: **This is an example review, I love this product!**
Sentiment: "

Table 4: Example prompt for (OA1) model, where the **bold** text represents the specific review to be classified. *Italic* fields are only inserted for 1 and 3-shot prompting.

Role	Input
system	"Predict the sentiment of a product review as either positive, neutral, or negative. You provide only ONE word as your answer, without any additional information, such as notes, comments, numbers, or explanations."
user	*"Works great, a little expensive but worth it."*
assistant	*"positive"*
user	*"Works great, a little expensive but worth it."*
assistant	*"neutral"*
user	*"Very low quality, I would not recommend this product."*
assistant	*"negative"*
user	"**This is an example review, I love this product!**"

Table 5: Example prompt for (OA2) model, where the **bold** text represents the specific review to be classified. *Italic* fields are only inserted for 1 and 3-shot prompting.

the words "good" and "product" have very high counts in both the oversampling and not oversampled cases. The same figure also shows that although some words, such as "quality", "nice", and "bad", have changed position, the overall distribution across words has stayed relatively the same.

5.2 Fine-tuning

As for the fine-tuning of (LLM3), Figure 4 shows the training and validation losses, both with and without oversampling the training data. In both cases, we can see how fast the model learns initially, then keeps learning but at a much slower pace after only a couple of steps. After about 1000-1500 training steps, the losses start to deviate from each other.

5.3 Model performances

This section contains the quantitative results for all experiments.

Small experiments

The results for the small experiments, using only 1000 randomly sampled data points, can be seen in Table 7. They show how the 0-shot (OA2) model generally performed the best, but a few other models got higher scores for individual categories. The results for testing the best overall performing model on both 5000 and the entire dataset, consisting of 33,931 samples, are shown in Table 8. The table indicates that the 5000 samples test produce higher scores than the full dataset was tests, but we need to keep in mind that these results are not directly comparable.

Large experiments

The results for the large experiments, conducted on all the test data, are shown in Table 6. The table shows that the two custom fine-tuned models outperformed all other models in all evaluated metrics, except for precision in the positive sentiment for (LLM1). The table also shows that oversampling the training data did not improve general results such as accuracy, macro and weighted avg F1-scores. Although the oversampled version of (LLM3) did achieve the highest F1-score for the neutral sentiment. The highest accuracy, macro, and weighted average F1-scores are all held by (LLM3) without oversampling.

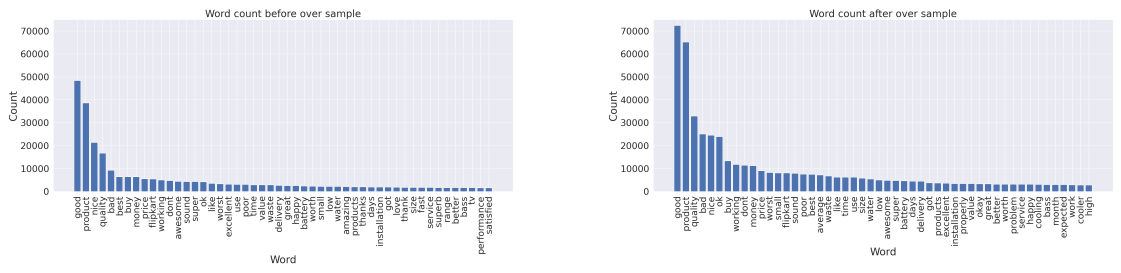

Figure 3: The word count distribution before (left) and after (right) oversampling the training data.

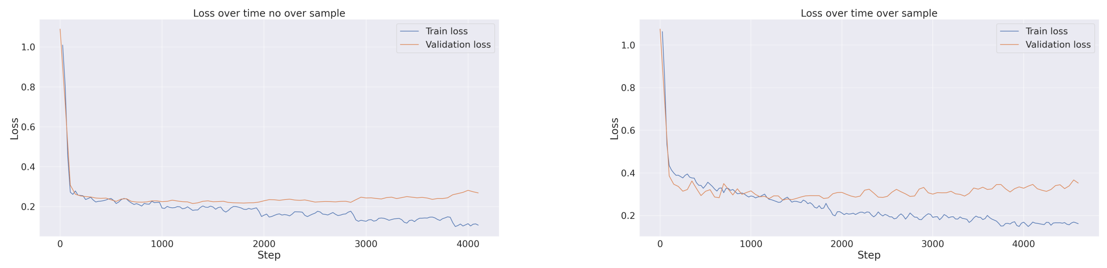

Figure 4: The loss over time without (left) and with (right) oversampling the training data.

6 Discussion

Figure 1 not only shows a significant difference in the number of positive sentiment examples compared to negative and neutral sentiments, but also that this stays consistent after the train/test split. These results suggest that the test and train data distributing are quite similar, which is a necessary property for effectively training and evaluating a model on the data. In other words, the train/test split has been done fairly.

Fine-tuning

The results for fine-tuning the (LLM3), shown in Figure 4, indicate that the sentiment analysis problem on this specific dataset is relatively easy for the model to learn. Despite selecting an early stopping patience of 25, the model only trains for approximately 4000-5000 training steps (3-4 epochs) before stopping. The plot also shows that overfitting to the training data begins to occur after around 1000-1500 training steps, where the losses for the training and validation sets begin to diverge.

Best model

The overall best performing model across the majority of tests was (LLM3), although the other models tested are not far behind. It's particularly interesting to see how the Naive Bayes classifiers remain competitive with the LLMs. This could be an indication that the sentiment analysis on this dataset is relatively easy. One reason for this could be that, as shown in Figure 2, the text length is quite short, and that in general, longer texts can be harder to understand. However, more tests with different datasets would be needed to confirm this claim. This could be an interesting area of future research, since no other extensive work specifically on how text length affects NB and LLMs sentiment classification has been found by the author.

The outperformance of the custom fine-tuned models compared to the pre-fine-tuned models is expected since the pre-fine-tuned models are more general and would instead likely produce better overall results if tested on other sentiment datasets. However, it's worth noting that the multilingual model (LLM2) outperformed the English model (LLM1) by almost 2% in accuracy. This could be due to them not being the same base model, with (LLM1) being a RoBERTa model and (LLM2) being a DistilBERT model. This agrees with the findings in (Adoma et al., 2020) which found that for sentiment analysis of emotions, their tested RoBERTa model beat all other models tested, including both normal BERT and DistilBERT models. Although these results cannot be directly compared since in their paper, they fine-tuned their models on the dataset used for comparisons, while in this paper, both models were fine-tuned on their individual datasets, which does not include the data used to produce this paper's results.

Oversampling

The experiment conducted on oversampling produced some interesting results. Typically, oversampling is used to balance out imbalanced datasets,

Mo & Se & Av	Precision	Recall	F1-score
(NB)			
negative	0.8418	0.8413	0.8416
neutral	0.6816	0.4355	0.5314
positive	0.9191	0.9615	0.9398
Accuracy			0.8858
Macro avg	0.8142	0.7461	0.7709
Weighted avg	0.8787	0.8858	0.8797
(NB) OS			
negative	0.8442	0.7786	0.8101
neutral	0.3961	0.6550	0.4937
positive	0.9666	0.9175	0.9414
Accuracy			0.8584
Macro avg	0.7356	0.7837	0.7484
Weighted avg	0.8866	0.8584	0.8691
(LLM1)			
negative	0.8657	0.8478	0.8567
neutral	0.3761	0.5931	0.4603
positive	**0.9715**	0.9085	0.9390
Accuracy			0.8662
Macro avg	0.7378	0.7832	0.7520
Weighted avg	0.8939	0.8662	0.8774
(LLM2)			
negative	0.8626	0.8689	0.8658
neutral	0.5092	0.4852	0.4969
positive	0.9418	0.9446	0.9432
Accuracy			0.8864
Macro avg	0.7712	0.7662	0.7686
Weighted avg	0.8848	0.8864	0.8856
(LLM3)			
negative	0.8898	**0.9428**	**0.9156**
neutral	**0.7424**	0.5091	0.6040
positive	0.9592	**0.9738**	**0.9664**
Accuracy			**0.9271**
Macro avg	**0.8638**	0.8086	**0.8287**
Weighted avg	**0.9226**	**0.9271**	**0.9229**
(LLM3) OS			
negative	**0.9098**	0.8902	0.8999
neutral	0.5737	**0.6750**	**0.6202**
positive	0.9704	0.9573	0.9638
Accuracy			0.9159
Macro avg	0.8180	**0.8408**	0.8280
Weighted avg	0.9214	0.9159	0.9183

Table 6: Results summary for (NB), (LLM1), (LLM2), and (LLM3) models with/without oversampling. OS = oversampled, Mo & Se & Av = model, sentiment, and averages.

Mo & Se & Av	Precision	Recall	F1-score
(OA1) 0-shot			
negative	0.8405	0.9416	0.8881
neutral	0.5500	0.3014	0.3894
positive	0.9464	**0.9434**	0.9449
Accuracy			0.8960
Macro avg	0.7790	0.7288	0.7408
Weighted avg	0.8866	0.8960	0.8878
(OA2) 0-shot			
negative	0.8683	0.9519	0.9082
neutral	0.4535	0.5342	**0.4906**
positive	**0.9798**	0.9167	**0.9472**
Accuracy			**0.8990**
Macro avg	0.7672	**0.8009**	**0.7820**
Weighted avg	**0.9090**	**0.8990**	**0.9025**
(OA1) 1-shot			
negative	0.7623	0.9588	0.8493
neutral	0.1957	0.1233	0.1513
positive	0.9609	0.8884	0.9232
Accuracy			0.8530
Macro avg	0.6396	0.6568	0.6413
Weighted avg	0.8472	0.8530	0.8453
(OA2) 1-shot			
negative	0.8791	0.9244	0.9012
neutral	0.2677	0.5205	0.3535
positive	0.9746	0.8459	0.9057
Accuracy			0.8450
Macro avg	0.7071	0.7636	0.7201
Weighted avg	0.8952	0.840	0.8641
(OA1) 3-shot			
negative	0.7983	**0.9656**	0.8740
neutral	**0.5897**	0.3151	0.4107
positive	0.9672	0.9261	0.9462
Accuracy			0.8930
Macro avg	**0.7850**	0.7356	0.7436
Weighted avg	0.8905	0.8930	0.8861
(OA1) 3-shot			
negative	**0.9088**	0.9244	**0.9165**
neutral	0.2232	**0.6849**	0.3367
positive	0.9771	0.7374	0.8405
Accuracy			0.7880
Macro avg	0.7030	0.7823	0.6979
Weighted avg	0.9022	0.7880	0.8258

Table 7: Results summary for OpenAI models (OA1) and (OA2) for 0, 1, and 3-shot prompting. Mo & Se & Av = model, sentiment, and averages.

Mo & Se & Av	Precision	Recall	F1-score
(OA2) (5000)			
negative	**0.8622**	**0.9426**	**0.9006**
neutral	**0.4632**	**0.5739**	**0.5127**
positive	0.9795	0.9100	0.9437
Accuracy			**0.8918**
Macro avg	**0.7683**	**0.8088**	**0.7856**
Weighted avg	**0.9049**	**0.8918**	**0.8965**
(OA2) (All)			
negative	0.8491	0.9385	0.8916
neutral	0.4591	0.5584	0.5039
positive	**0.9804**	**0.9106**	**0.9442**
Accuracy			0.8891
Macro avg	0.7628	0.8025	0.7799
Weighted avg	0.9021	0.8891	0.8938

Table 8: Results for (OA2) 0-shot, with 5000 randomly sampled testing points (top) and the whole (33931) test set (bottom), Mo & Se & Av stands for model, sentiment, and averages.

and in some cases, specifically yields better results for underrepresented classes (Brownlee, 2021).

The findings in Figure 2 indicate that oversampling the minority sentiments has increased the number of summaries across most text lengths. However, further analysis is required to determine how each sentiment's text length distribution is affected differently or if it stays overall the same. In Figure 3, it is shown that the distribution of word counts stays very consistent when oversampling. Although one can also see a large shift of the lesser-used words in the plot. These are reasonable results if one hypothesizes that the less often used words would be more common in the summaries of the oversampled sentiments. However, this is not a given, and once again, more analysis is needed.

The results shown in Table 6 indicate that the hypothesis for oversampling in this paper does not hold true, as oversampling yielded worse overall results in both tested cases. One possible explanation for this is that randomly oversampling leads to an overall increase of rare examples from underrepresented sentiments in the training data, causing the model to overfit on these rare examples that doesn't occur as frequently in the test data. This would therefore limit its ability to generalize over all sentiments. However, this is not the only explanation, and more experiments is required to confirm this hypothesis.

OpenAI models

The best-performing OpenAI model on the small experiments was (OA2). As mentioned earlier, that model was then also tested on the full dataset, shown in Table 8. These results show comparable performance to both the other language models and the baseline model, as presented in Table 6. The comparisons of these results is very interesting since this model was originally developed as a chatbot. This suggests the potential for very general-purpose language models to be effectively applied to sentiment analysis. Particularly interesting would be to test fine-tuning them for this specific task and dataset to see what improvement could be made. This is yet another fascinating area for future research to explore.

One of the most interesting results from the OpenAI-model experiments is shown in Table 7. The table shows that the 0-shot prompting for both models outperformed both the 1- and 3-shot prompts. This is the opposite to what was shown in (Brown et al., 2020), where the authors find that in majority of cases their LLMs performance scaled directly with the number of shots. One reason for this contradiction could be that their tests are more extensive and general, spanning multiple fields, models, and datasets. Another explanation could be that the particular way of conducting the multi-shot experiments in this paper might not be well-suited for this specific problem or dataset. The question of how to most effectively use multi-shot prompting for this particular setting is something this paper leaves for future research.

7 Conclusion

In conclusion, this paper investigated the effectiveness of various LLMs for sentiment analysis on a dataset of product reviews. The best-performing model was found to be a custom fune-tuned *DistilBERT base (uncased)* model. The paper also shows that for this problem and dataset, random oversampling did not improve any model. The paper further showed that the OpenAI chatbot GPT3.5-turbo model was also competitive with the other fine-tuned LLMs. However, the findings on the effectiveness of multi-shot prompts were contradictory to previous research, suggesting that more research is needed to understand how to best choose multi-shot prompts.

Although sentiment analysis is a much broader subject than discussed in this paper, and the results not being groundbreaking, the author still considers it a success. This project that the paper is based on has greatly inspired the author to continue their research into the fields of text mining and NLP.

References

Acheampong Francisca Adoma, Nunoo-Mensah Henry, and Wenyu Chen. 2020. Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 117–121.

Francesco Barbieri, José Camacho-Collados, Leonardo Neves, and Luis Espinosa Anke. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. *CoRR*, abs/2010.12421.

Marouane Birjali, Mohammed Kasri, and Abderrahim Beni-Hssane. 2021. A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems*, 226:107134.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Jason Brownlee. 2021. Random oversampling and undersampling for imbalanced classification.

OpenAI API chat guide. 2023. Openai api chat guide.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyu Gao. 2021. Prompting: Better ways of using language models for nlp tasks.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heinz, and Dan Roth. 2021. Recent advances in natural language processing via large pre-trained language models: A survey.

OpenAI API models. 2023. Openai api models.

philschmid. 2023. Philschmid/distilbert-base-multilingual-cased-sentiment-2 · hugging face.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Sebastian Ruder. 2021. Recent advances in language model fine-tuning.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Nirali vaghani. 2023. 171k product review with sentiment dataset.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.