
Exploring Data Cleaning using Language Model for Sentiment Analysis

Puttisan Mukneam
Pitzer College
Claremont, CA 91786
nmukneam@students.pitzer.edu

Abstract

1 In the realm of sentiment analysis, data cleaning is an indispensable but challenging
2 task. This study investigated the potential of applying pre-trained language models
3 to perform data cleaning tasks, specifically in the context of sentiment analysis
4 of Steam reviews. We utilized a suite of models, including gpt2, gpt2-xl, t5-
5 base, t5-xl, and two variations of the recently released flan-t5 model, flan-t5-
6 base and flan-t5-xl. The reviews generated by these pre-trained models were
7 then subjected to sentiment classification using Multinomial Naive Bayes and
8 Support Vector Machines classifiers, and the performance was assessed using Mean
9 Squared Error (MSE), Accuracy, and F1-score. The results varied with the baseline
10 method displaying superior performance. However, the flan-t5-base model showed
11 promising results, with an MSE of 0.96, an accuracy of 75.87%, and an F1-score of
12 0.821 when evaluated using SVM. Our project reveals the potential of pre-trained
13 language models in automating and enhancing data cleaning tasks, while also
14 suggesting that more extensive fine-tuning and larger datasets could lead to further
15 improvement.

16 1 Data Cleaning

17 Data cleaning, also known as data cleansing or data scrubbing, refers to the process of identifying
18 and rectifying or removing errors, inaccuracies, and inconsistencies in datasets. This vital stage in the
19 data preprocessing pipeline enhances the quality of data and significantly influences the outcome of
20 the subsequent analysis.

21 Data cleaning involves several crucial steps. These include removing duplicates, handling missing
22 values, correcting inconsistencies, and dealing with outliers. By ensuring that the data is accurate,
23 complete, consistent, and relevant, data cleaning facilitates the data mining process and helps in
24 making the most out of the collected data.

25 In the context of sentiment analysis, data cleaning plays a pivotal role. Sentiment analysis, also known
26 as opinion mining, involves interpreting and classifying emotions (positive, negative, and neutral)
27 within text data using text analysis techniques. This process allows organizations to understand the
28 social sentiment of their brand, product, or service while monitoring online conversations.

29 However, raw data collected from various sources, like social media platforms, reviews, forums,
30 and more, is typically unstructured and often contains noise in the form of irrelevant information,
31 typos, slang, emojis, and so forth. Data cleaning in sentiment analysis is therefore necessary for the
32 following reasons:

- 33 • **Noise Reduction** Raw text data contains a lot of noise such as irrelevant symbols, numbers,
34 punctuation marks, and special characters that may not contribute to sentiment analysis. Data

cleaning can help remove this noise, making the data more manageable and the subsequent analysis more accurate.

- **Standardization** Data from different sources can be in different formats or languages, and the same sentiment can be expressed in various ways. Data cleaning aids in standardizing the data, making it possible for the sentiment analysis algorithm to correctly interpret the sentiments.
- **Reducing Dimensionality** Text data tends to be high-dimensional due to the large number of unique words (known as the vocabulary). This can make sentiment analysis computationally expensive. Data cleaning techniques such as stemming, lemmatization, and stop-word removal can reduce the dimensionality of the text data, making the analysis process more efficient.
- **Handling Missing or Incomplete Data** Sometimes, the collected data may have missing or incomplete information, which can lead to incorrect analysis if not handled properly. Data cleaning helps in dealing with such issues either by filling the missing values with appropriate techniques or by excluding such instances after careful consideration.
- **Improving Model Performance** Clean and high-quality data is key to the performance of sentiment analysis models. Unprocessed or dirty data can mislead the training process of the models, leading to poor performance. Through data cleaning, the data fed into the models is more representative, leading to more reliable and robust models.

1.1 Data Cleaning in ML task

The paper "CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks" by Peng Li, and others (2021) makes several important observations that underscore the importance of cleaning data before it is utilized for machine learning tasks [9]. Here are their findings:

- **Missing Values:** The paper shows that the imputation of missing values, rather than their deletion, is more likely to improve or maintain the performance of machine learning models. This is a significant observation because missing data is a common problem in datasets. The authors further emphasize that the choice of imputation method and model can increase the positive impact and reduce any negative consequences of this data cleaning step.
 - **Outliers:** Cleaning outliers seems to have an insignificant impact on model performance. However, the authors note that the choice of model and cleaning algorithm can reduce the probability of negative impacts, emphasizing the importance of strategic decision-making in data cleaning processes.
 - **Mislabels:** The paper's findings suggest that cleaning mislabels is likely to have a positive or, at worst, insignificant impact on ML. Once again, the choice of model can increase the likelihood of positive impacts, underscoring the need for careful model selection.
 - **Inconsistencies:** Cleaning inconsistencies, according to the findings, is more likely to have an insignificant impact and is unlikely to negatively impact ML. This suggests that tackling inconsistencies in the data, while not always hugely impactful, is generally a safe cleaning step to undertake.
 - **Duplicates:** Cleaning duplicates is likely to have an insignificant or even negative impact on ML. This impact varies across detection methods and datasets, pointing to the importance of understanding the specific dataset and detection method used when cleaning duplicates.
- The study reveals that data cleaning - when done strategically and methodically - can be beneficial for machine learning models. Different types of data errors require different cleaning methods, and the choice of model and cleaning algorithm can significantly influence the outcomes. It also demonstrates that while some cleaning steps may not always significantly improve model performance, they rarely harm it, suggesting that a thorough data cleaning process is an important part of the machine learning pipeline. This paper thus provides strong evidence in favor of data cleaning as a good practice in ML tasks.

85 2 Backgrounds

86 2.1 Language Model

87 A Language Model (LM) is a type of statistical model that is used for predicting the next word in
88 a sentence given the previous words. It is a fundamental concept in the field of natural language
89 processing and has applications in various tasks such as machine translation, speech recognition,
90 part-of-speech tagging, and sentiment analysis, among others.

91 Formally, given a sequence of words $w = w_1, w_2, \dots, w_n$, a language model assigns a probability
92 $P(w)$ to the entire sequence, which can be factorized as:

$$P(w) = \prod_{i=1}^n P(w_i | w_1, \dots, w_{i-1}) \quad (1)$$

93 More recent language models use neural networks to predict the next word in a sequence. These
94 models, which include Recurrent Neural Networks (RNNs), Long Short-Term Memory networks
95 (LSTMs), and Transformers, can capture complex patterns and long-range dependencies in text data.
96 They have achieved state-of-the-art results on many NLP tasks.

97 In particular, the Transformer model uses a mechanism called self-attention to weigh the relevance
98 of words in the input sequence when predicting the next word. This architecture forms the basis of
99 many modern language models, including OpenAI's GPT series and Google's BERT.

100 2.1.1 GPT-2^[10]

101 Generative Pretrained Transformer 2 (GPT-2) is a large-scale language model developed by OpenAI.
102 As an autoregressive model, GPT-2 generates text by predicting the next word in a sequence, given the
103 previous words. The model was trained on a diverse range of internet text and achieved state-of-the-art
104 results on many natural language processing tasks, demonstrating its ability to generate coherent and
105 contextually relevant sentences.

106 GPT-2 extends the Transformer model architecture introduced in the paper "Attention is All You
107 Need", published by research team at Google in 2017. It uses a variant of the transformer decoder,
108 which consists solely of masked self-attention. However, GPT-2 is significantly larger than its
109 predecessor, GPT, in terms of both the number of parameters and the size of the training data.

110 GPT-2 consists of 1.5 billion parameters in its biggest model and was trained on a dataset of 8 million
111 web pages. It is designed to generate text by predicting the next word in a sequence, given all of the
112 previous words within some text. This is formally expressed as:

113 One of the key innovations in GPT-2 is the use of unsupervised learning. Rather than being trained
114 on a specific task, GPT-2 learns to generate text by predicting the next word in a sequence from a
115 large corpus of text. This enables it to generate coherent and contextually relevant sentences, while
116 also allowing it to perform well on a wide range of tasks without task-specific training data.

117 Despite its success, GPT-2 has also raised concerns about the potential misuse of large language
118 models, particularly in generating deceptive, biased, or offensive content. These challenges highlight
119 the importance of responsible AI use and the need for research into mitigating potential risks.

120 2.1.2 T5^[11]

121 Google's Text-to-Text Transfer Transformer (T5) is a highly flexible and versatile Transformer model,
122 trained to convert text inputs into text outputs. Unlike traditional models that are fine-tuned for each
123 specific task, T5 approaches all tasks as a text-to-text problem.

124 T5 views every NLP problem as a text generation task, which includes tasks like translation (text
125 in one language to text in another), summarization (long text to short text), sentiment analysis (text
126 to sentiment), and more. The advantage of this approach is that it allows the model to use the same
127 objective function (the likelihood of output text given input text) for pretraining and fine-tuning.

128 The T5 model was trained using a new variant of the denoising autoencoder objective, called "Causal
129 Language Modeling" (CLM). In CLM, the model is trained to recover the original text when provided

a corrupted version as input. The corruption process involves randomly masking out spans of text from the input and the model must predict the masked spans, similar to the BERT training process but with spans of text rather than individual tokens.

The model is pretrained on a large corpus of publicly available text from the internet, called Causal Language Modeling 1T (C4). After pretraining, T5 is fine-tuned on specific tasks using supervised training on individual task datasets.

2.2 Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a subfield of natural language processing (NLP) that focuses on extracting subjective information from text data. The primary goal of sentiment analysis is to determine the sentiment or emotional tone behind a series of words, which can be used to understand the attitudes, opinions, and emotions expressed by the author. Sentiment analysis is widely applied in various domains, including social media monitoring, product reviews, customer feedback analysis, and market research, among others.

There are various approaches to sentiment analysis, such as Lexicon-Based Methods, Machine Learning Methods, and Hybrid Methods. In this project, we will focus on Machine Learning Methods for sentiment analysis.

2.2.1 TF-IDF

There are various approaches to converting text data into numerical representations, such as Bag of Words, Word Embeddings, and Term Frequency-Inverse Document Frequency (TF-IDF). In this project, we use the TF-IDF method to represent the text data in the Steam reviews.

TF-IDF is a widely used technique in information retrieval and text mining that captures the importance of words in a document relative to a collection of documents. It assigns a weight to each word based on its frequency in a document and its rarity across the entire document collection.

TF-IDF consists of two components:

- **Term Frequency (TF):** It measures the frequency of a word in a document. A higher term frequency indicates the word is more important within the document. Mathematically, it can be expressed as:

$$TF(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

where $f_{t,d}$ is the frequency of term t in document d and the denominator represents the total frequency of all terms in the document.

- **Inverse Document Frequency (IDF):** It measures the rarity of a word across the entire collection of documents. Words that are more unique and rare receive higher IDF values. Mathematically, it can be expressed as:

$$IDF(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

where N is the total number of documents in the collection D , and the denominator represents the number of documents containing the term t .

The TF-IDF weight for a word in a document is the product of its TF and IDF values:

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

TF-IDF is chosen for this project due to its ability to balance the importance of words based on their frequency and rarity. It reduces the influence of common words, which appear in many documents but may not carry significant meaning for sentiment analysis. By representing the text data using TF-IDF, our machine learning models can better capture the relationships between the words in the reviews and their corresponding sentiment labels.

170 2.2.2 Naive Bayes

171 The Naive Bayes classifier is a probabilistic machine learning model based on Bayes' theorem,
 172 which is commonly used for text classification tasks such as sentiment analysis. It assumes that the
 173 features (words in our case) are conditionally independent given the class label, which simplifies the
 174 computation of probabilities.

175 Given a document d and a class label c , Bayes' theorem can be expressed as:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \quad (2)$$

176 The goal is to compute the probability $P(c|d)$ for each class and choose the class with the highest
 177 probability. Since the denominator $P(d)$ remains constant for all classes, we can ignore it and focus
 178 on the numerator:

$$P(d|c)P(c) = P(w_1, w_2, \dots, w_n|c)P(c) \quad (3)$$

179 where w_1, w_2, \dots, w_n are the words in the document.

180 Applying the conditional independence assumption, we can rewrite the probability as:

$$P(w_1, w_2, \dots, w_n|c)P(c) = P(c) \prod_{i=1}^n P(w_i|c) \quad (4)$$

181 To classify a document, we choose the class c^* with the highest probability:

$$c^* = \arg \max_c P(c) \prod_{i=1}^n P(w_i|c) \quad (5)$$

182 **2.2.2.1 Laplacian Smoothing** In practice, some words in the test data may not be present in the
 183 training data, leading to a zero probability for $P(w_i|c)$, which can cause issues in classification. To
 184 address this, Laplacian smoothing (also known as additive smoothing) is applied, which assigns a
 185 small non-zero probability to unseen words.

186 Given a word w_i and a class c , the smoothed probability is calculated as:

$$P(w_i|c) = \frac{f_{w_i,c} + \alpha}{\sum_{w \in V} (f_{w,c} + \alpha)} \quad (6)$$

187 where $f_{w_i,c}$ is the frequency of word w_i in class c , V is the vocabulary, and α is the smoothing
 188 parameter. A common choice for α is 1 (Laplace smoothing) or values between 0 and 1 (Lidstone
 189 smoothing).

190 2.2.3 Linear Support Vector Machine (SVM)

191 Support Vector Machines (SVM) are a class of supervised machine learning models used for classifi-
 192 cation and regression tasks. The main objective of an SVM is to find the optimal hyperplane that
 193 separates the data points of different classes with the maximum margin.

194 In the case of a linear SVM, the decision function is a linear combination of the input features:

$$f(x) = w^T x + b \quad (7)$$

195 where w is the weight vector, x is the input feature vector, and b is the bias term.

196 The goal is to minimize the following objective function:

$$\frac{1}{2}|w|^2 + C \sum_{i=1}^n \xi_i \quad (8)$$

subject to the constraints:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, \dots, n \quad (9)$$

where n is the number of training examples, y_i are the class labels, ξ_i are the slack variables that allow some margin violations, and C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification errors.

3 Implementation

3.1 Steam Reviews Dataset [1]

Steam is a digital distribution platform developed by Valve Corporation, which provides video game developers with a platform to distribute their games and allows gamers to purchase and play those games. It also offers various additional features, such as game updates, friend lists, achievements, and cloud storage for game progress. As one of the largest digital distribution platforms for PC gaming, Steam has a massive user base and hosts thousands of games ranging from indie titles to AAA releases.

Steam allows users to write and submit reviews for the games they have played. These reviews are valuable sources of information for potential buyers, as they provide insights into the gameplay experience, performance, and overall satisfaction of the players. The dataset used in this project was obtained from Kaggle [1] and contains Steam game reviews submitted by users.

The dataset consists of several columns, including:

- **app_id**: A unique Game ID on the Steam store.
- **app_name**: The name of the game being reviewed.
- **review_text**: The text of the review written by the user.
- **review_score**: Review sentiment, indicating whether the review recommends the game (1) or not (-1).
- **review_votes**: The count of whether the review was voted useful by another user or not.

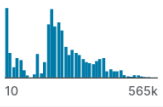


app_id	app_name	review_text	review_score	review_votes
Game ID	Game Name	Review text	Review Sentiment: whether the game the review recommends the game or not.	Review vote: whether the review was recommended by another user or not.
	[null] 3% PAYDAY 2 1% Other (6144899) 96%	Early Access Review 16% Early Access Review 0% Other (5392421) 84%		
10	Counter-Strike	Ruined my life.	1	0
10	Counter-Strike	This will be more of a 'my experience with this game' type of review, because saying things like '...	1	1
10	Counter-Strike	This game saved my virginity.	1	0
10	Counter-Strike	• Do you like original games? • Do you like games that don't lag? • Do you like games you can run on...	1	0

Figure 1: Steam Reviews [1]

In this project, we focus on the 'review_text' and 'review_score' columns to perform sentiment analysis. The 'review_text' column provides the text data for analysis, while the 'review_score' column serves as the sentiment label (positive if the game is recommended, negative otherwise).

3.2 Justification

Based on the challenges identified in the paper "Data Cleaning: Overview and Emerging Challenges" by Xu Chu and others (2016), the use of language models can potentially offer solutions to these data cleaning problems[8]. Here are some of the challenges identified in the paper and our proposed solutions using language models:

- **Scalability:** Language models, when properly trained and fine-tuned, can handle data of any size and perform data cleaning tasks effectively and efficiently. This can help to scale data cleaning techniques for large and rapidly growing datasets in the Big Data era.
- **User Engagement:** With an optimized language model, the need for human intervention in data cleaning tasks can be significantly reduced. This not only increases the efficiency of the process but also mitigates the risks of human errors.
- **Semi-structured and unstructured data:** The use of machine learning tools can help detect the type of data and structure it appropriately before passing it into the cleaning model. This can help address data quality problems for semi-structured and unstructured data formats.
- **Privacy and Security Concerns:** Due to the automated nature of the data cleaning process using language models, human intervention is minimized, thereby addressing privacy and security concerns. An optimized language model doesn't require user to access the raw data, which can help to ensure privacy.

Given the challenges and findings from the above studies, we propose using language models such as GPT or T5 for data cleaning. The reasons include:

- **Automation:** Language models can automate the data cleaning process, reducing manual effort and the need for exhaustive pattern matching or rule creation.
- **Versatility:** With the right prompt, a language model can perform various cleaning tasks, such as removing stop words, special characters, and unrelated words.
- **Adaptability:** Language models can keep up with evolving language use, including slangs and new semantics, by fine-tuning them on recent data.
- **Handling Unknowns:** Language models, especially those using attention mechanisms, can handle unknown or unexpected data elements better due to their ability to understand long-term dependencies in the data.
- **All-in-one tool:** With the right prompt, a language model can perform multiple cleaning tasks at once, such as removing special characters, stop words, and non-sentimental words, detecting type error and fixing them.

This approach addresses the challenges identified in the previous studies, offering a scalable, automated, and adaptable solution for data cleaning in the context of machine learning and sentiment analysis.

3.3 Methodology

3.3.1 Data Processing

The first step in our data processing pipeline involves importing our raw review data into a Pandas DataFrame for easier manipulation. We leverage a dataset of over six million Steam reviews sourced from Kaggle [1]. To make the pre-training process more manageable, we randomly select a subset of 1000 reviews, ensuring that 65% of them are positive, and the remaining 35% are negative. This decision was motivated by our preliminary tests that indicated significant results could be achieved with this sample size.

Number of Reviews	Mean Squared Error	Accuracy	F-1 Score
1,000	1.08	73.0	0.791
10,000	0.824	79.4	0.843
100,000	0.629	84.285	0.883

Table 1: Midterm’s model performance on testing dataset of various data size using the baseline filtered review evaluated using SVM with $C = 10$ as regularization parameters.

3.3.2 Creating the Baseline Reviews

With our selected subset of 1000 reviews, we create a new column in our DataFrame named `filtered_review_text`. This column contains a filtered version of the raw review text, which is stored in the `review_text` column. We also retain the sentiment labels of the original raw reviews in a column named `review_score`.

The `filtered_review_text` column serves as our baseline for comparison with the reviews generated by the pre-trained language model. We generate this column using standard Python libraries, including pandas, re, nltk, and nltk.corpus.stopwords.

Our data cleaning process follows standard practices in sentiment analysis:

- Conversion of reviews into lowercase to ensure uniformity and avoid duplication based on case differences.
- Removal of all special characters using regular expressions, reducing noise in the text data.
- Elimination of stopwords (commonly used words that do not contribute to the sentiment of a sentence, e.g., 'the', 'is', 'in') to focus on words that carry sentiment.
- Stripping trailing spaces to maintain consistency in the data.
- Dropping duplicate and NAs strings value review.

The result of this process is a clean and consistent set of reviews that we can use as a baseline for evaluating our language model’s performance.

3.3.3 Generating Filtered Review using Pre-trained Models

A challenge we faced during this project was finding an accessible, capable model. Although GPT-3/4 possess impressive capabilities, they are not open source and require payment based on the number of tokens used via the API. This becomes cost-prohibitive especially with long or numerous reviews. Consequently, we decided to explore alternative models that, while not as powerful, are open source and allow for reasonable experimentation.

We elected to use GPT-2, Google’s T5, and Google’s Flan [12]. Flan is an optimized fine-tuned version of the original T5 model on various NLP task, making it a genuine competitor to GPT-2 and other proprietary models. Importantly, Flan is an open-source pre-trained model, making it accessible to all for individual experimentation.

We utilized the following pre-trained models:

Model	Number of Parameters
gpt2	117M
gpt2-xl	1558M
t5-base	250M
t5-xl	3000M
flan-t5-base	250M
flan-t5-xl	3000M

Table 2: Pre-trained models

295 Given the computational demands of deep learning models, we used Google Colab for this project, as
296 it provides access to powerful GPUs, which are not typically available to average individuals or on
297 consumer-grade PCs.

298 In Google Colab, we primarily used the Hugging Face transformers library to implement these
299 models. Both the tokenizer and the pre-trained models were obtained from this source.

300 For every model, the input prompt was:

301 "Keeps only sentimental words: *review_text*"

302 We used this prompt to achieve multiple filters, such as removing special characters and non-
303 sentimental words (i.e., stopwords, etc.). Additionally, we aimed to use fewer tokens to save on
304 execution time. More details on run-time can be found in the results section.

305 The models' outputs were stored in a new column titled `model_name`, adjacent to the original
306 `review_text` column.

307 For GPT-2, here are the parameters we used for the tokenizer and model:

```
308 tokenizer(input,  
309           max_length=50,  
310           truncation=True,  
311           return_tensors='tf')  
312  
313 model.generate(**encoded_input,  
314               early_stopping=True,  
315               max_length=25,  
316               repetition_penalty = 5.2,  
317               top_k = 80)  
318  
319 tokenizer.batch_decode(output, skip_special_tokens=True)[0]
```

320 For T5 and Flan-T5:

```
321 encoded_input = tokenizer(input,  
322                           max_length=50,  
323                           truncation=True,  
324                           return_tensors='pt').input_ids.to("cuda")  
325  
326 model.generate(encoded_input,  
327               early_stopping=True,  
328               max_length=64)  
329  
330 tokenizer.batch_decode(output, skip_special_tokens=True)[0]
```

331 Many of these parameters were chosen to save execution time.

332 3.4 Validating the Filtered Review

333 The objective of this project is to utilize pre-trained language models for data cleaning in sentiment
334 analysis tasks. Consequently, our validation metric involves evaluating the performance of sentiment
335 classification on the cleaned reviews.

336 To validate the effectiveness of our approach, we will compare the results of sentiment classification
337 on the baseline filtered reviews and the cleaned reviews generated by the pre-trained models.

338 We will employ the same machine learning algorithms as outlined in "Exploring the Sentiment of
339 Steam Reviews with Naive Bayes and Support Vector Machine" by Puttisan Mukneam (2023) [2].
340 These algorithms include Multinomial Naive Bayes with Laplacian Smoothing and Support Vector
341 Machines (SVM) with various smoothing parameters and regularization parameters.

342 The performance of these algorithms will be evaluated using Mean Squared Error (MSE), Accuracy,
343 and F1-score. These metrics will provide a comprehensive view of the model's performance, from

overall accuracy to the balance between precision and recall (F1-score), and the average squared differences between predicted and actual outcomes (MSE).

In both MVB and SVM models, we drop reviews where their values were NA to combat TypeError during training and testing process. We also have split the filtered reviews. We decided to use 80% as training data for the model, and 20% for testing. Lastly, out of the training data, we extract 20% to be a local testing data during the training process, in order to determine optimal parameters to be used during testing phase.

3.4.1 Multinomial Naive Bayes

We trained the Multinomial Naive Bayes classifier using various parameters for Laplacian smoothing. Then we evaluate model's performance on the testing data set using a fixed smoothing parameter of 0.1, and fixed the seed for reproducibility. The MultinomialNB class from the scikit-learn library was utilized for this purpose.

3.4.2 Linear Support Vector Machine (SVM)

We trained a Linear Support Vector Machine (SVM) classifier with various regularization parameter C values. We also fixed the seed, and tested the model using a fixed regularization parameters of C=1. The implementation was performed using the LinearSVC function from the scikit-learn library.

4 Result

Here are the results and discussion of our finding.

4.1 Testing

Model	Mean Squared Error	Accuracy	F-1 Score
Baseline	0.7	82.5	0.865
gpt2	1.18	70.5	0.784
gpt2-xl	1.2	70.0	0.781
t5-base	1.2	70.0	0.776
t5-xl	1.24	69.0	0.780
flan-t5-base	0.96	75.87	0.821
flan-t5-xl	1.02	74.371	0.80

Table 3: Models performance on testing dataset evaluated using SVM with $C = 1$ as regularization parameters (fixed seed).

Model	Mean Squared Error	Accuracy	F-1 Score
Baseline	0.74	81.5	0.864
gpt2	1.0	75.0	0.821
gpt2-xl	1.1	72.5	0.808
t5-base	1.3	67.5	0.758
t5-xl	1.08	73.0	0.80
flan-t5-base	1.25	74.37	0.815
flan-t5-xl	1.14	71.356	0.78

Table 4: Models performance on testing dataset evaluated using Naive Bayes with $a = 0.1$ as smoothing parameter (fixed seed).

4.2 Assessing Results

The performance of the pre-trained models varied in comparison to the baseline, with both SVM and Multinomial Naive Bayes classifiers. Although the baseline model demonstrated superior performance

across all metrics, the pre-trained models showed promising results despite some limitations in our experiment.

Among the pre-trained models, flan-t5-base displayed the most promising performance, with relatively low MSE, high accuracy, and a reasonably balanced F1-Score. This suggests that with the right fine-tuning and optimization, this model could potentially improve and possibly outperform traditional data cleaning methods.

The gpt2, gpt2-xl, t5-base, t5-xl, and flan-t5-xl models showed varied performance, with some performing better in certain metrics than others. While their overall performance did not exceed that of the baseline, it's essential to consider that the parameters used in this experiment were chosen to expedite run times. With more comprehensive fine-tuning, the use of optimal prompts, and a larger dataset, these models could potentially yield improved results.

It's worth noting that the size of the dataset used in this experiment was relatively small. We used only 1000 reviews, which might not fully harness the power of these models. Larger datasets could potentially provide the models with more context and variability, leading to more accurate and robust performance.

In conclusion, while the baseline model outperformed the pre-trained models in this experiment, the latter hold significant potential for improvement and application in the data cleaning domain. Future work involving optimized parameters, fine-tuning, and larger datasets could yield more promising results.

4.3 Drawbacks of Using Pre-trained Models

Although pre-trained models offer several advantages, there are notable drawbacks that need to be considered when implementing them for data cleaning tasks. Some of these drawbacks are as follows:

1. **Computational expense:** Fine-tuning these pre-trained models is computationally expensive. It necessitates advanced resources and significant computational power, which may not be readily available to every researcher or developer. Additionally, an optimized pre-trained model for data cleaning is currently lacking, which adds to the computational overhead.
2. **Time consumption:** Generating cleaned reviews using these models can be time-consuming.

Model	Generating time
gpt2	20m
gpt2-xl	1hr
t5-base	4m
t5-xl	25m
flan-t5-base	5m
flan-t5-xl	12m

Table 5: Approximate generating time of 1000 reviews on a free Google Colab account

393

3. **Accessibility:** Many state-of-the-art models such as GPT-3/4 are not open-source but are accessible via paid APIs. This can be prohibitive for small businesses or individual developers who cannot afford the associated costs.

4. **Prompt design:** It can be challenging to design optimal prompts when the model has not been fine-tuned, as the training details of the original model may not be available or clear. This lack of transparency can lead to inefficiencies and inaccuracies in the data cleaning process.

5. **Knowledge requirement:** Fine-tuning these models requires extensive knowledge and understanding of their architecture and parameters. Finding optimal parameters can be challenging and may require a large amount of data and computational resources. This again can be expensive and beyond the means of many users.

- 405 6. **Data Bias:** Pre-trained models can carry biases from their training data. This can lead
406 to skewed results when these models are used for data cleaning, particularly in sentiment
407 analysis tasks.
- 408 7. **Black Box Models:** Understanding why a model made a certain prediction or decision can
409 be challenging, making it hard to diagnose issues or improve the model.
- 410 8. **Lack of Flexibility:** Pre-trained models may not perfectly fit the data cleaning task at hand,
411 requiring additional work in fine-tuning or adapting the model for the specific task.

412 5 Future Work

413 Based on the challenges and limitations identified in this project, several areas of future work are
414 suggested:

- 415 1. **Fine-tuning FLAN-T5 for data cleaning:** We found that fine-tuning the FLAN-T5-base
416 model on 300K reviews with a high-end GPU (40GB memory) and 70 compute units using
417 Google Colab Pro took more than 9 hours. However, given the flexibility of the FLAN-T5
418 architecture in handling various NLP tasks, we believe it holds potential for being effectively
419 fine-tuned to perform data cleaning tasks.
- 420 2. **Exploring specialized deep learning models:** We plan to investigate more optimized deep
421 learning models that are specifically designed for data cleaning tasks. This might help in
422 enhancing the performance and reducing the computational overhead associated with current
423 models.
- 424 3. **Fine-tuning parameters:** There is scope for exploring different fine-tuning strategies,
425 tokenizers, and model parameters to improve the efficacy of the pre-trained models in
426 cleaning tasks.
- 427 4. **Exploring optimal prompts:** Further work can be done to investigate and design optimal
428 prompts for the pre-trained models to maximize their performance in data cleaning tasks.

429 6 References

- 430 [1] Sobkowicz, A. (2017). Steam Reviews Dataset. Retrieved from [https://www.kaggle.com/datasets/](https://www.kaggle.com/datasets/andrewmvd/steam-reviews)
431 [andrewmvd/steam-reviews](https://www.kaggle.com/datasets/andrewmvd/steam-reviews)
- 432 [2] Mukneam, P. (2023). sentiment_steam. Retrieved from [https://github.com/pmukneam/sentiment_](https://github.com/pmukneam/sentiment_steam)
433 [steam](https://github.com/pmukneam/sentiment_steam)
- 434 [3] Konduru, Y. J. (2021). Game Review Classifier. Retrieved from [https://yaswanth3277.github.io/](https://yaswanth3277.github.io/Portfolio/gamereviewclassifier.html)
435 [Portfolio/gamereviewclassifier.html](https://yaswanth3277.github.io/Portfolio/gamereviewclassifier.html)
- 436 [4] Kumar, S. (2018). IMDB movie review polarity using Naive
437 Bayes Classifier. Retrieved from [https://satyam-kumar.medium.com/](https://satyam-kumar.medium.com/imdb-movie-review-polarity-using-naive-bayes-classifier-9f92c13efa2d)
438 [imdb-movie-review-polarity-using-naive-bayes-classifier-9f92c13efa2d](https://satyam-kumar.medium.com/imdb-movie-review-polarity-using-naive-bayes-classifier-9f92c13efa2d)
- 439 [5] Reddy, V. (2018). Sentiment Analysis using SVM. Retrieved from [https://medium.com/@vasista/](https://medium.com/@vasista/sentiment-analysis-using-svm-338d418e3ff1)
440 [sentiment-analysis-using-svm-338d418e3ff1](https://medium.com/@vasista/sentiment-analysis-using-svm-338d418e3ff1)
- 441 [6] Narayanan, V., Aror, I., & Bhatia, A. (2013). Fast and accurate sentiment classification using an enhanced
442 Naive Bayes model. Indian Institute of Technology.
- 443 [7] Zuo, Z. (2018). Sentiment Analysis of Steam Review Datasets using Naive Bayes and Decision Tree
444 Classifier. Retrieved from <http://hdl.handle.net/2142/100126>
- 445 [8] Chu, X., Ilyas, I. F., Krishnan, S., & Wang, J. (2016). Data Cleaning: Overview and Emerging Chal-
446 lenges. SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA. DOI: [http://dx.doi.org/10.1145/](http://dx.doi.org/10.1145/2882903.2912574)
447 [2882903.2912574](http://dx.doi.org/10.1145/2882903.2912574)
- 448 [9] Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X., & Zhang, C. (2021). CleanML: A Study for Evaluating
449 the Impact of Data Cleaning on ML Classification Tasks. 2021 IEEE 37th International Conference on Data
450 Engineering (ICDE), Chania, Greece. DOI: 10.1109/ICDE51399.2021.00009.
- 451 [10] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., & others. (2019). Language models are
452 unsupervised multitask learners. OpenAI blog, 1(8), 9.

- 453 [11] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020).
454 Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning*
455 *Research*, 21(140), 1-67.
- 456 [12] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma,
457 S., & others. (2022). Scaling Instruction-Finetuned Language Models. *arXiv*. [https://arxiv.org/abs/](https://arxiv.org/abs/2210.11416)
458 [2210.11416](https://arxiv.org/abs/2210.11416)
- 459 [13] Mukneam, P. (2023). sentiment-deep-cleaning. Retrieved from [https://github.com/pmukneam/](https://github.com/pmukneam/sentiment-deep-cleaning)
460 [sentiment-deep-cleaning](https://github.com/pmukneam/sentiment-deep-cleaning)