# Deep Learning vs. Traditional Machine Learning for Sentiment Analysis: A Performance Comparison

Shlomit Finegold

# Abstract

In the past years, the hype around deep learning methods has become increasingly prominent, especially when compared to traditional machine learning approaches.
The field of natural language processing (NLP) has undergone significant development and advancements due to the emergence of deep learning (DL) techniques.
DL has revolutionized NLP by enabling models to automatically learn and extract complex features from raw text data, leading to improved performance in various NLP tasks. One of the most popular NLP tasks is sentiment analysis, which aims to determine the sentiment expressed in textual data. This project investigates and compares the performance of state-of-the-art deep learning methods with traditional machine learning techniques for sentiment analysis on the IMDb reviews dataset. The results show that the traditional ML approach managed to achieve learning, though far from being compatible with the DL approach in terms of accuracy. As for precision-recall trade off, there was a clear preference for DL models for precision optimization and for traditional ML models for recall optimization.

# Introduction and literature review

In the last two decades, there has been a significant rise in text data (Berger et al., 2020), often referred to as "big data" (Borgman, 2017). It is estimated that 80-90% of today's data is unstructured (Harbert, 2021), with a significant portion consisting of text data. According to Tan et al. (2023), the increase in text data is largely due to the widespread use of e-commerce platforms, consumer review sites and travel aggregators. These platforms allow consumers to provide numerical ratings and text-based explanations of their overall product or service experiences. One of those sites is IMDb that allows users to rate and write movie or TV shows reviews and provides a publicly available database for sentiment analysis. (Tan et al., 2023).
This project uses this database for comparison between advanced DL techniques - pre-trained and fine tuned “DistilBERT” model and state of the art ML methods with pre-processing of feature engineering. Despite the abundance of research in these domains, the majority of studies concentrate solely on DL methods or ML methods combined with frequency based encoders. (Tan et al., 2023). In contrast, this project employs feature extraction techniques to extract NLP based features for the ML approach, extracting numerical features such as word count, percentage of punctuation, and the ratio of positive to negative common words.

**What is sentiment analysis?**

Sentiment analysis is categorizing text into positive, negative, or neutral sentiments. Understanding sentiments on online platforms is crucial for informed decision-making. It helps companies enhance satisfaction, boost brand reputation, and increase revenue by grasping customer sentiments. Sentiment analysis is a valuable tool for political analysis, as it can be used to assess public opinion on parties, candidates, and policies. In the financial industry, it enables stock price prediction and identification of investment opportunities by analyzing news articles and social media posts. (Tan et al., 2023). As extracting meaningful information from the vast amounts of data available has become increasingly crucial, sentiment analysis has experienced significant advancements within the field of NLP. (Liaqat et al., 2022). The following section will dive into some of those advanced NLP methods and their preprocessing steps as well as traditional ML methods for classification.

**Text preprocessing**

Feature engineering is crucial for text classification as it converts unstructured textual data into numerical features that classification models can utilize, enabling the leveraging of machine learning methods. (Garla, Brandt, 2012). The success of machine learning frequently hinges on the effectiveness of feature engineering that can involve various methodologies. Some techniques rely heavily on domain knowledge, while others are derived from intuition or data analysis. (Verdonck et al., 2021). The choice of text pre-processing technique relies heavily on the specific model we intend to utilize. When working with traditional ML models such as SVM (Support Vector Machine), naive Bayes, RF (random forest), and others, which typically take a feature matrix as input with features represented as columns, there are two main feature extraction approaches available. The first approach involves extracting NLP-based features such as word count, noun count, and more using feature extraction techniques. The alternative option is to employ encoders that will represent the text as a numeric vector. (Başarslan, 2023, Tan et al., 2023). The first encoders that have been introduced were frequency-based such as TF-IDF (Term Frequency-Inverse Document Frequency) and BoW (Bag of Words). Frequency-based encoders transform documents into numeric vectors based on word counts, but they lack correlations between terms and fail to capture word context. (Xu et al., 2013). With the advancements in NLP, researchers have redirected their attention towards DL techniques. In DL, a similar process is applied , text is first encoded into pre-trained word embeddings, which then utilized as input for DL classifiers. Word embeddings for DL have a variety of advanced techniques, all of which consider both the word's position within the text and its associations with other words, yielding feature vectors that reflect the contextual nuances of the words. (Wang et al., 2019). Mikolov et al. (2013) introduced Word2Vec as one of the early and most popular static embedding methods, which marked the beginning of a rapid evolution in the field of word embeddings. Since then, embedding techniques have progressed rapidly, with the introduction of more advanced models, Pennington et al. (2014) introduced GloVe (Global Vectors for Word Representation) and Joulin et al. (2016) introduced FastText, among others.

Since text is often “messy”, before applying any embedding technique, preprocessing should be applied on the text itself. As Berger et al. (2020) describe, there are several steps that can be taken in order to improve the text embedding and analysis:

1. Cleaning: HTML tags and non-textual information, like images, are eliminated or removed from the dataset. The extent of cleaning required depends on the data format in which it was obtained or extracted. Data sourced from the web often requires more extensive cleaning due to the presence of HTML tags.
2. Stop word removal: Stop words refer to common words like "a" and "the" that appear frequently in documents but typically contribute little to the overall meaning. It is recommended to add domain-specific common words (e.g., "Amazon" in a corpus of Amazon reviews) to this list. However, depending on research objectives, retaining stop words can be valuable for analyzing writing style.
3. Spelling correction: Text-mining packages often offer built-in spell checkers, but researchers should consider domain-specific language that may not be recognized or incorrectly corrected. Counting spelling mistakes can provide valuable insights into the author's text characteristics.
4. Stemming and lemmatization: Stemming reduces words to their stems, while lemmatization returns the proper lemma, preserving meaning. If capturing writing style, skipping stemming is preferable as it may alter text tense.
5. Tokenization: Tokenization breaks text into units (words or sentences), with researchers defining delimiters like spaces, periods, or semicolons, though special cases may need attention to ensure sensible tokens. (Berger et al., 2020).

Once the preprocessing and feature extraction steps have been completed, the processed text can be utilized as input for a model.

**ML methods for text classification**

Naïve Bayes: One of the most ancient methods to create classification since it’s based on Bayes theorem which was published in 1763. The probabilistic classifier, used to generate possibilities of a group to provide prediction that group of properties belongs to one particular label. (Mehta, Pandya, 2020)

SVM: SVM (Support Vector Machines) is a popular ML method for classification and regression analysis, constructing hyperplanes to separate data points and maximize the margin between the hyperplane and support vectors (closest points to the classification margin) (Gove, Faytong, 2012)

Extensive research has been conducted in text classification, by combining frequency-based encoders with traditional machine learning classifiers such as NB and SVM. Omar et al. (2013) employed the TF-IDF preprocessing method and compared various ML classifiers, including NB and SVM for sentiment analysis of Arabic customer reviews, while Sriram et al. (2010) employed BoW and NB for text classification of Twitter data, exemplifying the diverse approaches explored in this field.

XGBoost: the advancement of computational capabilities has paved the way for the creation of more powerful models. Notably, Chen and Guestrin (2016) have introduced XGBoost, a classification model that has demonstrated exceptional efficiency and competitiveness across various challenges. XGBoost utilizes decision trees as its fundamental learning model, employing the ensemble strategy gradient boosting to address overfitting. By combining multiple weak base learning models iteratively and refining predictors using residuals, XGBoost optimizes the loss function to create a more powerful learner. (Chen, Guestrin, 2016)

While traditional models widely used for text classification combined with frequency-based encoders, the introduction of XGBoost came at a time when researchers had already shifted their focus towards working with neural network architectures and embedding preprocessing techniques such as word2text (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). As a result, there has been limited research on text classification using XGBoost. This project aims to explore the capabilities of XGBoost, an advanced machine learning model, in conjunction with a more traditional preprocessing method - extracting NLP based features. To provide a baseline for this preprocessing method, it is being compared to the more traditional models - NB and SVM and the recent development of DL.

**DL methods for text classification**

RNN: Rumelhart et al. (1986) has introduced RNNs (Recurrent Neural Networks ), but it wasn't until advancements in computing power and the availability of large-scale datasets that RNNs became practical for real-world applications and gained popularity for text processing and NLP tasks. Unlike traditional neural networks, RNNs have feedback connections and hidden layers that allow information transfer across time steps. In RNNs, each neuron processes input and produces an output, and the output of one hidden layer becomes the input for the next layer. This recursive function-like behavior enables RNNs to model explicit time dependencies. (Wei et al., 2017)

LSTM: Hochreiter and Schmidhuber (1997) has introduced LSTM (Long Short-Term Memory). The LSTM model is an enhanced version of the traditional RNN model. It incorporates a control mechanism to overcome the RNN's long-term dependence and overcome gradient vanishing/explosion issues caused by lengthy sequences. By introducing a specialized cell structure, the LSTM model enables the RNN to memorize long-term information effectively. Furthermore, the design of three "gate" structures-the forget gate layer, the input gate layer, and the output gate layer allows for selective addition and removal of information within the cell structure. (Hochreiter, Schmidhuber ,1997)

In order to utilize RNN and LSTM, an embedding phase is typically required. With the emergence of DL-based embedding methods, RNN and LSTM have become popular choices for text classification. One of many research works is Liu et al. (2016) who employed word2vec embeddings and RNN-based architectures for modeling text sequences, leveraging multi-task learning.

Vaswani et al. (2017) suggested a new approach and architecture that named Transformer and created a revolution in text analysis. The new architecture integrates the embedding phase and the classification phase. Unlike traditional approaches involving recurrence and convolutions, the Transformer model is solely based on attention mechanisms. Unlike recurrent networks that process words sequentially, with each word relying on information from the previous word, the Transformer processes the entire input sequence as a whole. Another notable feature of the Transformer is the introduction of positional embedding, which captures word order information. (Vaswani et al., 2017)

Since transformers architecture was introduced, there have been several attempts to create pre-trained models for text analysis tasks. Devlin et al. (2018) introduced BERT (Bidirectional Encoder Representations from Transformers) which was the first model to provide compatible results across several tasks and became state of the art for text analysis. BERT is a transformer-based language model designed for pre-training deep bidirectional representations from unlabeled text, achieving outstanding performance for tasks like question answering and language inference with minimal modifications. Unlike previous models, BERT overcomes unidirectionality by introducing a masked language model objective, allowing it to capture bidirectional context. It also employs the next sentence prediction task for further pre training enhancements.(Devlin et al., 2018)

After BERT was introduced, numerous models emerged that adopted its architecture, making minor adjustments while retaining the core principles of BERT. Sanh et al. (2019) proposed a method to pre-train a smaller general-purpose language representation model, called DistilBERT, that is faster by 60% from BERT and smaller by 40% than BERT while retaining 97% of its language understanding capabilities. DistilBERT was trained on the same corpus as the original BERT model and has the same general architecture as BERT, however, DistilBERT focuses on reducing the number of layers - which was reduced by a factor of 2 and optimizing the operations in the transformer architecture. (Sanh et al.,2019)

Until now, the transformers architecture remains widely regarded as the most effective method for conducting text analysis tasks, with no new architecture surpassing its performance. BERT and its latest variations remained one of the most common models of text classification. Therefore, for this project, I used DistilBERT, both the pre-trained off the shelf solution and a fine-tuned version trained on the IMDb dataset.

# Method

Numerous methodologies are available for sentiment analysis, in this project, two main groups are used. The traditional “non-deep” machine learning models - Naive Bayes (NB), Support Vector Machine (SVM) and Extreme Gradient Boosting (XGBoost) are compared with one of the most advanced DL models - DistilBERT. For each method a different preprocessing is conducted:

Preprocessing for traditional ML:

1. Text basic cleaning - removal of html tags and url
2. Extracting NLP based numerical features
    - Count of words
    - Percent of name entities (count of name entities / count of words)
    - Percent of punctuation (count of punctuation / count of words)
    - Percent of uppercase words (count of uppercase words / count of words) - Percent of stop words (count of stop words / count of words)
    - Percent of spelling mistakes (count of spelling mistakes / count of words) - Percent of nouns (count of nouns / count of words)
    - Percent of verbs (count of verbs / count of words)
    - Percent of adjectives (count of adjectives / count of words)
    - Ratio between common positive and negative words (extraction top 200 words per label, clean from shared words, counting the number of words for each label and dividing positive word counter by negative word counter. for avoiding deviation by zero, zeros are being replaced with a small epsilon)
3. Standardization and scaling
4. Feature selection - removing features that have correlation with other features and are less correlated with the label according to correlation table (appendix 1).

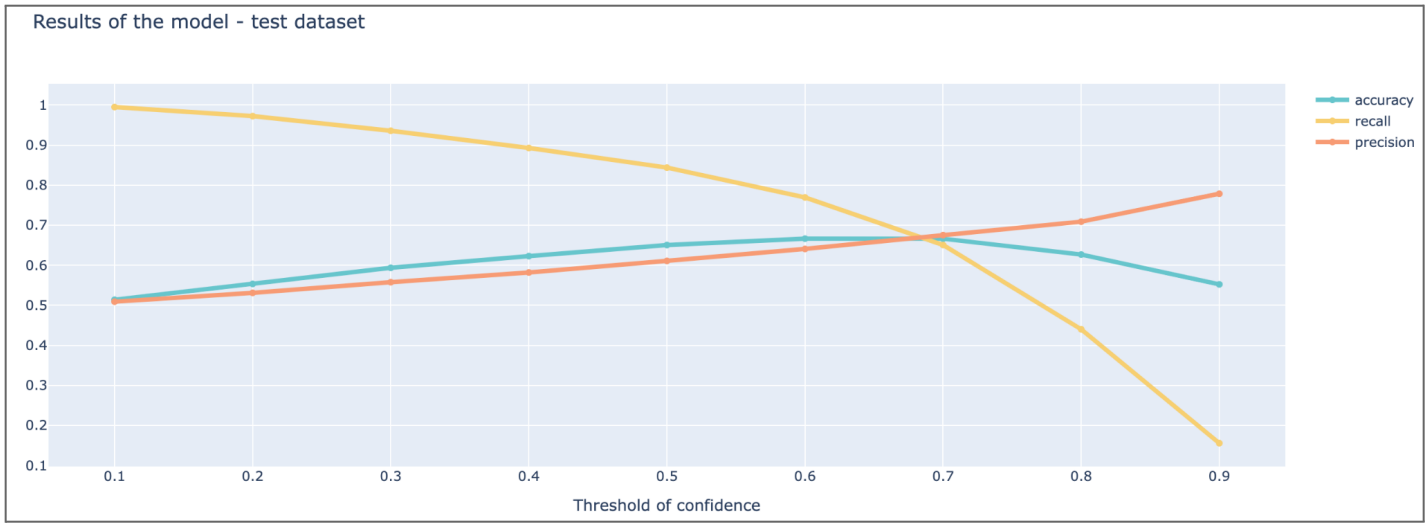
Preprocessing for DL:

During the exploratory data analysis (EDA) phase, certain insights were gained regarding how the text should be processed.

- Repetitive text containing HTML tags (such as <br/ br>) and URLs was removed from the text.
- Since approximately 98% of the reviews contain named entities (see appendix 2), all keywords were converted to lowercase and punctuation was removed, except for the named entities to preserve their contextual information.
- Although stop words constituted nearly 50% of the text in the reviews (see appendix 3), the reviews were sufficiently lengthy (with a median of 170 words and a third quartile of 278 words) to still contain significant relevant information. Hence, stop words were eliminated.
- To enhance the tokenizer's performance, lemmatization was applied to the text. In order to use the DistilBERT model, tokenization is being applied on the final clean text; the used tokenizer is also by DistilBERT.

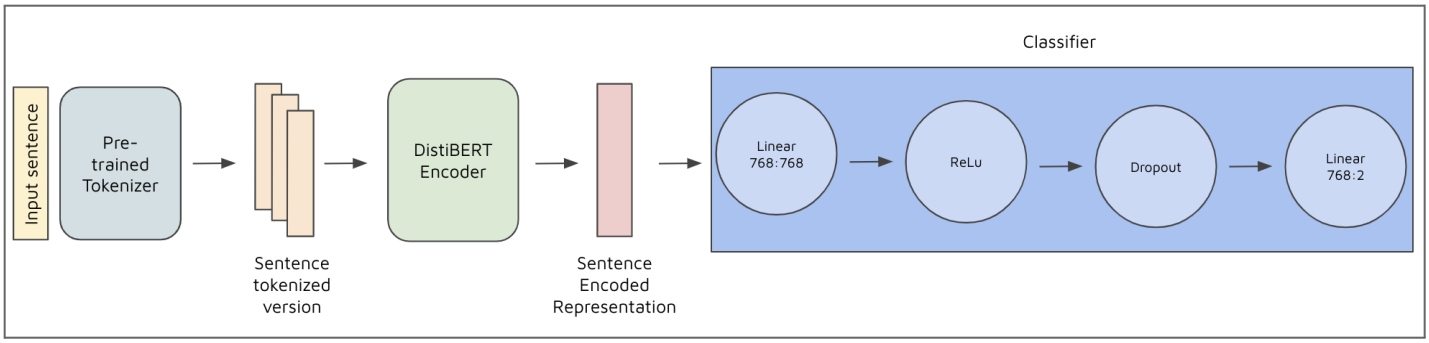
The main examined ML model is XGBoost, while SVM and NB are utilized as baseline models for comparison purposes. The XGBoost model underwent parameter optimization using grid search, resulting in the final parameter values of {'n_estimators': 200, 'min_samples_split': 5, 'max_depth': 3, 'learning_rate': 0.1}. In addition to standard prediction, threshold optimization was implemented. Figure 1 demonstrates the optimization of the model using various thresholds to improve different performance metrics.

Figure 1: XGBoost results per threshold of confidence



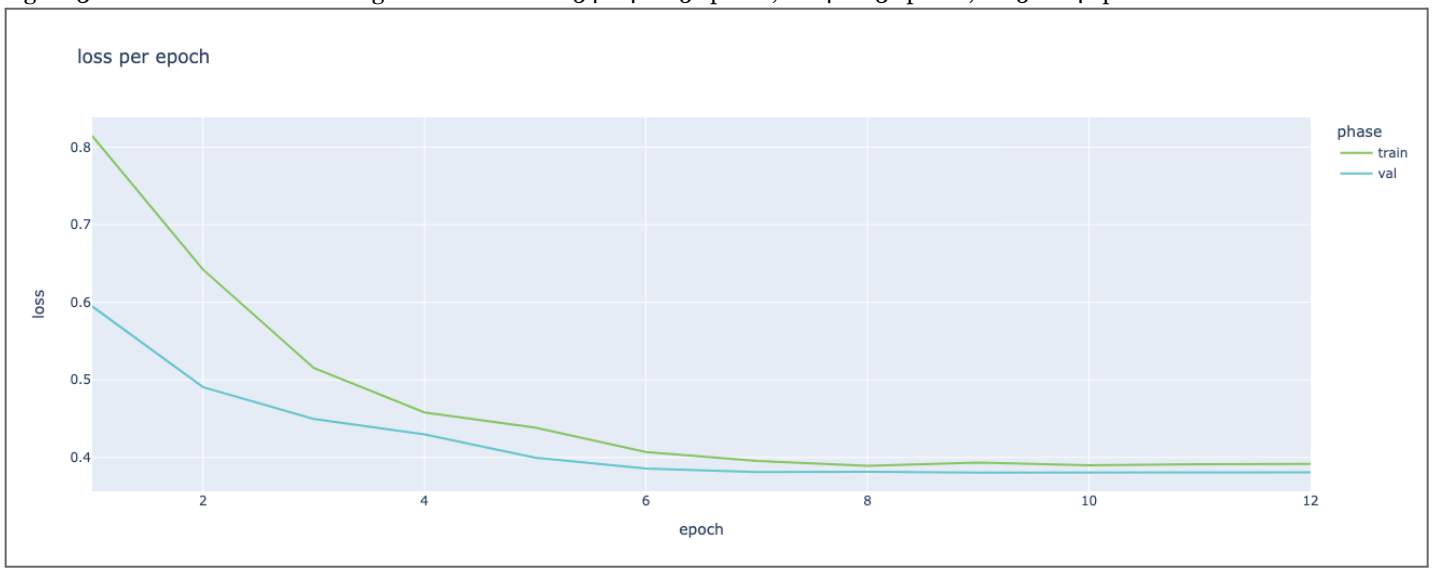

The DL baseline is set by implementing DistilBERT pretrained off the shelf model using huggingface, without any training on the dataset. The compared model is also DistilBERT, but with fine-tuning the classification layers on the dataset. Figure 2 presents the model's architecture, with the marked classifier layers that were fine tuned.

Figure 2: DistilBERT classification architecture




During the training process, the hyperparameters such as the number of epochs, learning rate with step decay, and optimizer-loss were fine-tuned. The loss curves depicted in Figure 3 were analyzed to determine the optimal values for these hyperparameters. Specifically, 6 epochs were chosen as the ideal number of epochs, with a learning rate set to 5e^-4 for 5 epochs, e^-4 for the last epoch. The selected optimizer for this study was Adam optimizer. After fine-tuning the hyperparameters, the training and validation sets were merged to create a final training set for re-training the model before prediction is made on the test set.

Figure 3: Loss curves for fine tuning classification: lr=54^-4 for 5 epochs, e^-4 for 3 epochs, e^-5 for 4 epochs




# Results

Table 1 presents the results obtained from the experiments conducted. In light blue - the traditional ML models: the SVM model achieved an accuracy of 0.649, a recall of 0.697, and a precision of 0.639. Naive Bayes exhibited an accuracy of 0.504, a recall of 0.99, and a precision of 0.504. XGBoost, with a threshold of 0.6 optimized for accuracy, attained an accuracy of 0.666, a recall of 0.769, and a precision of 0.64. XGBoost, with a threshold of 0.4 optimized for recall while ensuring a minimum accuracy greater than 0.6, demonstrated an accuracy of 0.622, a recall of 0.892, and a precision of 0.581.

In pink - the DL models: the pre-trained DistilBERT model achieved an accuracy of 0.786, a recall of 0.633, and a precision of 0.909. Fine-tuning the DistilBERT model further improved its performance, resulting in an accuracy of 0.818, a recall of 0.748, and a precision of 0.868. On the recall column we can see in bold that the best score belongs to XGBoost optimized for recall, compared with the best DL score, there's a 14% gap. On the precision column pre-trained model achieved the best score (while ignoring the un-learned model, NB) compared with the best ML score, there's a 27% gap. On the accuracy column, the best performance is with the fine tuned model, with 3% improvement from the pre-trained model and with a gap of 15% from the best ML score.

The Naive Bayes (NB) model exhibits a unique behavior where it predicts nearly all reviews as positives, leading to exceptionally high recall and random accuracy scores, as the dataset is perfectly balanced. On the other hand, the remaining traditional ML models achieved performance above random chance, indicating some level of learning from the features. Nevertheless, these models did not demonstrate strong compatibility with the DL models, suggesting differences in their learning capabilities and approaches.

Table 1: Comparison of all models' results, accuracy, recall and precision

| Method | Model | Accuracy | Recall | Precision |
|---|---|---|---|---|
| Traditional ML | SVM | 0.649 | 0.697 | 0.639 |
| | Naive Bayes | 0.504 | 0.99 | 0.504 |
| | XGBoost (ths=0.6 optimizing accuracy) | 0.666 | 0.769 | 0.64 |
| | XGBoost (ths=0.4 optimizing recall with min accuracy > 0.6) | 0.622 | **0.892, 14%** | 0.581 |
| SOTA DL | Pre-trained DistilBERT | 0.786 | 0.633 | **0.909, 27%** |
| | Fine Tuned DistilBERT | **0.818, 15%** | 0.748 | 0.868 |

# Conclusions

The results of this study lead to several significant conclusions:
First and foremost, the possibility of achieving learning outcomes using the outdated method of extracting NLP-based numeric features was demonstrated. Surprisingly, even with this approach, some level of learning was observed. However, it should be noted that employing one of the most advanced traditional ML models, XGBoost, did not yield significant improvements beyond the baseline models. This highlights the limitations of relying solely on advanced ML techniques without incorporating more sophisticated feature extraction methods such as word embedding.

The Naive Bayes (NB) model, on the other hand, failed to effectively predict and learn any meaningful features. While previous research has shown its success in text classification using frequency-based embedding, the approach of extracting NLP-based features proved to be unsuitable for preprocessing for NB.

Furthermore, in general, the DL models utilizing embedding pre-processing showcased superior performance compared to the ML models relying on NLP-based extracted features. The substantial gap between these approaches reveals a clear disparity in their ability to learn tasks and make accurate predictions.

Additionally, there were noticeable differences between the ML and DL models. The ML models, such as SVM and XGBoost, demonstrated a focus on recall, whereas the DL models prioritized precision. High recall and low precision indicate a tendency to predict more instances as positive, capturing a higher proportion of actual positive instances but potentially including false positives. High precision and low recall suggest a more selective prediction approach, where instances labeled as positive are done so with high confidence, but the model may miss several

actual positive instances. Although the DL approach performed better overall, there are specific use cases where the ML approach may offer advantages. The ML approach demonstrates superior performance in scenarios where maximizing true positives is desirable, even at the cost of some false positives. This is particularly relevant in domains such as advertising or reminders. On the other hand, the DL approach excels in situations where minimizing false positives is crucial, as in the healthcare domain where misclassification must be minimized to a great extent.

In conclusion, this study highlights the importance of incorporating advanced DL techniques and fine-tuning DL models for achieving superior performance in sentiment classification tasks. While the traditional ML approach showcased some learning capabilities, it fell short of the DL models' effectiveness. However, there may still be situations where the ML approach proves beneficial, particularly when the goal is to cover as many true positives as possible, even at the expense of some false positives. Conversely, the DL approach is more advantageous in domains where minimizing false positives is critical.

# Discussion

Sentiment analysis is facing many challenges such as understanding sarcasms, slang or culture references, usually the challenge is bigger for products that interact in real time with humans and need to classify the sentiment correctly and reply a proper response. This challenge becomes even more complex in products that heavily rely on communication. A domain greatly affected by misclassification in sentiment analysis is the realm of care bots for elders, often referred to as sidekick bots. These bots aim to foster collaboration and empathy with their users, but achieving this goal becomes arduous when the user's sentiment is misunderstood. Another issue of mis-classification sentiment is the bot’s understanding of when it is a good (or bad) time to intercat - especially for proactive bots. The issue is even bigger when bots are trained on the responses they receive as misunderstanding the user's sentiment can lead to highly inappropriate or unwelcome responses. Consider, for instance, an educational bot shared by grown and young siblings. If the bot is trained on sarcastic responses from the grown sibling, it may inadvertently provide inappropriate responses to the younger sibling. As bots become increasingly integrated into various aspects of our lives and human-bot interactions become commonplace, the ability to process text and comprehend sentiment becomes a vital skill for bot developers. While sentiment analysis is a powerful tool, the development teams should approach it’s application cautiously to prevent awkward situations or negative impacts on the product. With limited current performance, the product teams should make smarter decisions when and how to use it, highlighting it’s advantages and hiding it’s challenges.

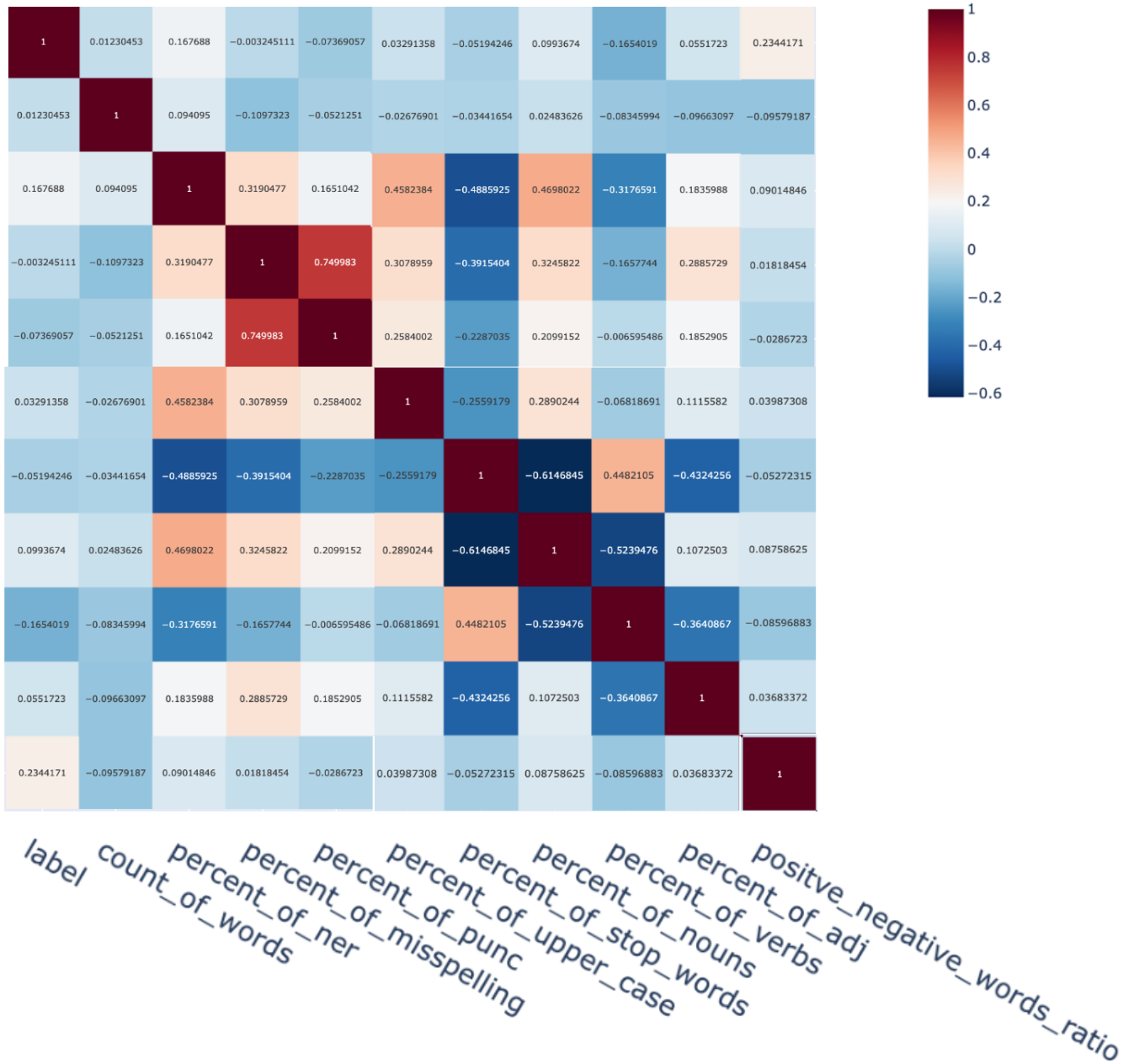
# References


1. Başarslan, M. S., & Kayaalp, F. (2023). MBi-GRUMCONV: A novel Multi Bi-GRU and Multi CNN-Based deep learning model for social media sentiment analysis. Journal of Cloud Computing, 12(1), 5.
2. Berger, J., Humphreys, A., Ludwig, S., Moe, W. W., Netzer, O., & Schweidel, D. A. (2020). Uniting the tribes: Using text for marketing insight. Journal of marketing, 84(1), 1-25.
3. Borgman, C. L. (2017). Big data, little data, no data: Scholarship in the networked world. MIT press.
4. Chen, M., Weinberger, K. Q., & Sha, F. (2013). An alternative text representation to tf-idf and bag-of-words. arXiv preprint arXiv:1301.6770.
5. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
6. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. 7. Garla, V. N., & Brandt, C. (2012). Ontology-guided feature engineering for clinical text classification. Journal of biomedical informatics, 45(5), 992-998.
8. Gove, R., & Faytong, J. (2012). Machine learning and event-based software testing: classifiers for identifying infeasible GUI event sequences. In Advances in computers (Vol. 86, pp. 109-135). Elsevier.
9. Harbert, T. (2021). Tapping the power of unstructured data. MIT Sloan. Feb, 1, 3. 10. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
11. Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.
12. Liaqat, M. I., Hassan, M. A., Shoaib, M., Khurshid, S. K., & Shamseldin, M. A. (2022). Sentiment analysis techniques, challenges, and opportunities: Urdu language-based analytical study. PeerJ Computer Science, 8, e1032.
13. Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101.
14. Mehta, P., & Pandya, S. (2020). A review on sentiment analysis methodologies, practices and applications. International Journal of Scientific and Technology Research, 9(2), 601-609.
15. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
16. Omar, N., Albared, M., Al-Shabi, A. Q., & Al-Moslmi, T. (2013). Ensemble of classification algorithms for subjectivity and sentiment analysis of Arabic customers' reviews. International Journal of Advancements in Computing Technology, 5(14), 77.
17. Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

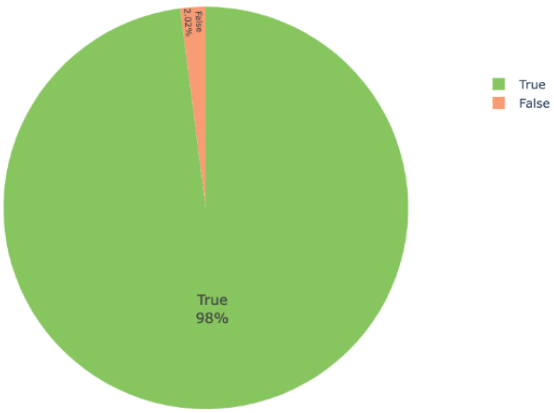
18. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.
19. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108. 20. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., & Demirbas, M. (2010, July). Short text classification in twitter to improve information filtering. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (pp. 841-842).
21. Tan, K. L., Lee, C. P., & Lim, K. M. (2023). A survey of sentiment analysis: Approaches, datasets, and future research. Applied Sciences, 13(7), 4550.
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
23. Verdonck, T., Baesens, B., Óskarsdóttir, M., & vanden Broucke, S. (2021). Special issue on feature engineering editorial. Machine Learning, 1-12.
24. Wang, B., Wang, A., Chen, F., Wang, Y., & Kuo, C. C. J. (2019). Evaluating word embedding models: Methods and experimental results. APSIPA transactions on signal and information processing, 8, e19.
25. Wei, D., Wang, B., Lin, G., Liu, D., Dong, Z., Liu, H., & Liu, Y. (2017). Research on unstructured text data mining and fault classification based on RNN-LSTM with malfunction inspection report. Energies, 10(3), 406.

# Appendices

Appendix 1 - correlation matrix after normalization and scaling, used for feature selection for the traditional ML models



Appendix 2 - percent of reviews with at least one name entity, used to decide on a strategy of text pre-processing




Appendix 3 - percent of stop words per sentiment label, used to decide on a strategy of text pre-processing